VERİ GRİD SİSTEMLERİNDE GERÇEK-ZAMANLI VERİ YÖNETİMİ

Mustafa Müjdat Atanak

Doktora Tezi

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Haziran 2012

ANADOLU ÜNİVERSİTESİ

JÜRİ VE ENSTİTÜ ONAYI

Mustafa Müjdat Atanak'ın **"Veri Grid Sistemlerinde Gerçek-zamanlı Veri Yönetimi"** başlıklı **Elektrik-Elektronik Mühendisliği** Anabilim Dalındaki Doktora Tezi 02.02.2012 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı Soyadı	**İmza**
Üye (Tez Danışmanı)	**: Doç. Dr. ATAKAN DOĞAN**	**.....**
Üye	**: Yard. Doç. Dr. CÜNEYT AKINLAR**	**.....**
Üye	**: Yard. Doç. Dr. EMİN GERMEN**	**.....**
Üye	**: Yard. Doç. Dr. EROL SEKE**	**.....**
Üye	**: Yard. Doç. Dr. NİHAT ADAR**	**.....**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Doktora Tezi

VERİ GRİD SİSTEMLERİNDE GERÇEK-ZAMANLI VERİ YÖNETİMİ

Mustafa Müjdat ATANAK

Anadolu Üniversitesi

Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Atakan DOĞAN

2012, 154 sayfa

Bilimsel ve ticari uygulamaların ihtiyaç duydukları işlem gücü, veri depolama alanı ve ağ bant genişliği gereksinimi, gün geçtikçe artmaktadır. Bu gereksinimlerin karşılanması için, binlerce işlemciyi ortak bir uygulama çatısı altında çalıştırabilen Veri Grid sistemleri geliştirilmektedir. Yüksek miktarlarda verinin işlenmesini gerektiren veri yoğun uygulamalar, artan sıklıkta gerçek-zaman kriterine gereksinim duymaktadırlar. Gerçek-zamanlı uygulamalar adı verilen bu uygulamaların, belirli bir son zamandan önce tamamlanmaları gerekmektedir. Gerçek-zamanlı uygulamaları çalıştıran Veri Grid sistemlerinin performansları, iş çizelgeleme, veri dağıtımı, veri kopyalama ve ön rezervasyon sistemi gibi mekanizmalardan etkilenmektedir. Bu çalışmada, yukarıdaki dört unsuru da barındıran bir Veri Grid sistemi modeli sunulmuştur. Önerilen modelde, Veri Grid sistemini oluşturan servisler ve servisler arası etkileşimler tanımlanmıştır. Önerilen model, hiyerarşik iş çizelgeleme, hiyerarşik veri dağıtımı, çekme tabanlı, itme tabanlı dağıtık ve itme tabanlı merkezi veri kopyalama modellerini desteklemektedir. Ayrıca, önerilen modelde, sistem üzerinde bulunan kaynakların önceden rezerve edilebilmesi için bir sistem geliştirilmiştir. Simülasyon çalışmaları için üç veri erişim düzeni (rassal, geometrik ve zipf) ve iki veri organizasyon modeli (federatif ve hiyerarşik) tanımlanmıştır. Rand, EDF, MCTF, MCwDP, MMwDP iş çizelgeleme algoritmaları önerilmiş ve performans ölçümleri yapılmıştır. Gerçek-zamanlı veri transferi isteklerinin bir rotadan (RTU/DDP) veya birden fazla rotadan (RTS/DDP) karşılanması problemi tanıtılmış, her iki problemin çözümü için de keşifsel yaklaşımlar önerilmiştir. Veri dağıtımı problemi için önerilen SP_MinHop, SP_MinDelay, SP_MinMin, SP_MinCon, MinHop/FPF, MinDelay/FPF, MinMin/FPF, MinCon/FPF, IO_UFF, MNOFF, MOFF, GA, kSP, kDP, ESMP, BSMP algoritmalarının performans ölçümleri yapılmıştır. Farklı veri kopyalama modelleri için algoritmalar geliştirilmiş ve önerilen MRD, RT_DIW ve RT_CENT algoritmalarının performans ölçümleri yapılmıştır.

Anahtar Kelimeler: Gerçek-Zaman, Veri Grid, Modelleme, Çizelgeleme, Ön Rezervasyon, Veri Dağıtımı, Veri Kopyalama, Bilgisayar Ağları, Algoritma Analizi, Kaynak Yönetimi, Simülasyon

ABSTRACT

Ph.D. Dissertation

REAL-TIME DATA MANAGEMENT IN DATA GRID SYSTEMS

Mustafa Müjdat ATANAK

Anadolu University

Graduate School of Sciences

Electrical and Electronics Engineering Program

Supervisor: Assoc. Prof. Dr. Atakan DOĞAN

2012, 154 pages

The computing power, storage space and network bandwidth requirements of scientific and commercial applications are increasing. In order to meet these requirements, Data Grid systems that orchestrates thousands of processors under a common application framework are developed. Data intensive applications that require large datasets to be processed, comes with real-time requirements with increasing frequency. These applications are called real-time applications and they come with a predefined deadline before which the applications should be completed. The performances of Data Grid systems that runs real-time applications are affected by the underlying job scheduling, data dissemination, data replication and advance reservation mechanisms. A Data Grid system model that incorporates all these components is presented in this study. In the proposed model, services that constitutes the Data Grid system and inter service interations are defined. In the proposed model, hierarchical job scheduling, hierarchical data dissemination, pull based, push based distributed and push based centralized data replication models are supported. Furthermore, in the proposed model, an advance reservation system is devised to enable advance reservation of the system resources. Three data access profiles (random, geometric, and zipf) and two data organization models (federative and hierarchical) are defined. Rand, EDF, MCTF, MCwDP, MMwDP job scheduling algorithms are proposed and performances are measured. The problems of satisfying real-time data transfer requests via one (RTU/DDP) or more paths (RTS/DDP) are introduced. SP_MinHop, SP_MinDelay, SP_MinMin, SP_MinCon, MinHop/FPF, MinDelay/FPF, MinMin/FPF, MinCon/FPF, IO_UFF, MNOFF, MOFF, GA, kSP, kDP, ESMP, BSMP heuristic algorithms are proposed for data dissemination problem and performances are measured. Algorithms for various data replication models are developed and performance measurements of the proposed MRD, RT_DIW and RT_CENT algorithms are shown.

Keywords: Real-Time, Data Grid, Modelling, Scheduling, Advance Reservation, Data Dissemination, Data Replication, Computer Networks, Algorithm Analysis, Resource Management, Simulation

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

BGP : Border Gateway Protocol (Sınır Geçit Protokolü)
BSMP : Balanced Share Multi Path (Dengeli Bölüşümlü Çoklu Rota)
CE : Computing Element (İşlem Elemanı)
CMS : Compact Selenoidal Detector (Kompak Selenoid Algılayıcı)
DDOA : Data Dissemination Optimization Algorithm (Veri Dağıtımı En İyileme Algoritması)
DM : Data Manager (Veri Yöneticisi)
DMS : Data Management Service (Veri Yönetim Servisi)
DRS : Data Replication Service (Veri Kopyalama Servisi)
DTS : Data Transfer Service (Veri Transfer Servisi)
EDF : Earliest Deadline First (En Erken Son Zaman İlk)
ESMP : Equal Share Multi Path (Eşit Bölüşümlü Çoklu Rota)
FPF : Feasible Path First (Uygun Rota İlk)
GA : Genetic Algorithm (Genetik Algoritma)
GarQ : Grid Advance Reservation Queue (Grid Önceden Rezervasyon Sırası)
GCL : Grid Component Library (Grid Bileşen Kütüphanesi)
GDMS : Grid Data Management Service (Grid VeriYönetimi Servisi)
Giggle : Giga-Scale Global Location Engine (Giga-Ölçekli Küresel Konumlama Motoru)
GIS : Global Information Service (Global Bilgi Servisi)
GJDM : Grid Job Dispatch Manager (Grid İş Dağıtım Yöneticisi)
GJSS : Grid Job Submission Service (Grid İş Arz Servisi)
GMS : Global Management Site (Global Yönetim Sitesi)
GRAM : Globus Resource Allocation Manager (Globus Kaynak Paylaştırma Yöneticisi)
GSI : Grid Security Infrastructure (Grid Güvenlik Altyapısı)
GSS : Grid Scheduling Service (Grid Çizelgeleme Servisi)
GUI : Graphical User Interface (Grafik Kullanıcı Arayüzü)
GZAP : Gerçek-Zamanlı Akış Paylaşımı

GZİS : Gerçek-Zamanlı İş Seççimi
GZRS : Gerçek-Zamanlı Rota Seçimi
IETF : İnternet Mühendislik İş Gücü (Internet Engineering Task Force)
IntServ : Integrated Services (Entegre Servisler)
IO_UFF : In Order Unsatisfiable Flow First (Sıradan Karşılanamayan Akış İlk)
JDL : Job Description Language (İş Tanımlama Dili)
JIM : Job Invoke Manager (İş Başlatma Servisi)
*k*DP : *k*-Shortest Disjoint Path (*k*-En Kısa Ayrık Yol)
*k*GZRS : *k*-Adet Gerçek-Zamanlı Rota Seçimi
*k*SP : *k*-Shortest Path (*k*-En Kısa Yol)
LHC : Large Hadron Collider (Büyük Hadron Çarpıştırıcısı)
LDM : Local Data Manager (Yerel Veri Yöneticisi)
LDMS : Local Data Management Service (Yerel Veri Yönetimi Servisi)
LDRS : Local Data Replication Service (Yerel Veri Kopyalama Servisi)
LFN : Logical File Name (Mantıksal Veri İsmi)
LFU : Least Frequently Used (En Az Sıklıkta Kullanılan)
LIS : Local Information Service (Yerel Bilgi Servisi)
LMCE : Local Management CE (Yerel Yönetim İşlem Elemanı)
LRC : Local Replica Catalogue (Yerel Kopya Kataloğu)
LRLS : Local Replica Location Service (Yerel Kopya Konum Servisi)
LRS : Local Reservation Service (Yerel Rezervasyon Servisi)
LRU : Least Recently Used (En Önce Kullanılan)
MCTF : Minimum Completion Time First (En Erken Bitiş Zamanlı İlk)
MCTFwDP : Minimum Completion Time First with Data Present (Veriyi Barındıran En Erken Bitiş Zamanı İlk)
MDS : Monitoring and Discovery Service (İzleme ve Bulma Servisi)
MI : Millions of Instructions (Milyon Komut)
MinCon : Minimum Contention (En Az Çakışmalı)
MinDelay : Minimum Delay (En Az Gecikmeli)
MinHop : Minimum Hop (En Az Atlamalı)
MIPS : Milllions of Instructions Per Second (Saniyedeki Milyon İşlem)

MKP : Multidimensional 0-1 Knapsack Problem (Çok boyutlu 0-1 Sırt Çantası Problemi)
MMwDP : MinMin with Data Present (Veriyi Barındıran MinMin)
MNOFF : Maximum Number of Outgoing Flows First (En Fazla Sayıda Çıkan Akış İlk)
MOFF : Maximum Outgoing Flows First (En Fazla Çıkan Akış İlk)
MRD : Most Requested Data (En Fazla İstek Alan Veri)
NRM : Network Resource Manager (Ağ Kaynak Yöneticisi)
NWS : Network Weather Service (Ağ Durumu Servisi)
OGF : Open Grid Forum (Açık Grid Forumu)
OGSA : Open Grid Services Architecture (Açık Grid Servisleri Mimarisi)
OGSI : Open Grid Services Infrastructure (Açık Grid Servisleri Altyapısı)
OSPF : Open Shortest Path First (En Kısa Açık Yol İlk)
PFN : Physical File Name (Fiziksel Veri İsmi)
QoS : Quality of Service (Hizmet Nitelikleri)
Rand : Random (Rassal)
RB : Resource Broker (Kaynak Simsarı)
RC : Replica Catalogue (Kopya Kataloğu)
RIP : Routing Information Protocol (Yönlendirme Bilgi Protokolü)
RLI : Replica Location Index (Kopya Yer Göstergesi)
RLS : Replica Location Service (Kopya Konum Servisi)
RM : Replica Manager (Kopya Yöneticisi)
RO : Replica Optimizer (Kopya Eniyileştiricisi)
RS : Reservation Service (Rezervasyon Servisi)
RSVP : Resource Reservation Protocol (Kaynak Rezervasyon Protokolü)
RT-CENT : Real-Time Centralized (Gerçek-Zamanlı Merkezi)
RT-DIW : Real-Time Distributed Weighted (Gerçek-Zamanlı Dağıtık Ağırlıklı)
RT-MaxMax : Real-Time MaxMax (Gerçek-zamanlı MaxMax)
RTU/DDP : Real-Time Unsplittable Data Dissemination Problem (Gerçek-Zamanlı Bölünemez Veri Dağıtımı Problemi)
RTS/DDP : Real-Time Splittable Data Dissemination Problem (Gerçek-

Zamanlı Bölünebilir Veri Dağıtımı Problemi)

SE : Storage Element (Depolama Elemanı)

SDMS : Site Data Management Service (Site Veri Yönetimi Servisi)

SJDM : Site Job Dispatch Manager (Site İş Dağıtım Yöneticisi)

SJSS : Site Job Submission Service (Site İş Arz Servisi)

SOAP : Simple Object Access Protocol (Basit Nesne Erişim Protokolü)

SP_MinCon : Single Pass MinCon (Tek Aşamalı MinCon)

SP_MinDelay : Single Pass Minimum Delay(Tek Aşamalı En Az Gecikmeli)

SP_MinHop : Single Pass Minimum Hop (Tek Aşamalı En Az Atlamalı)

SP_MinMin : Single Pass MinMin (Tek Aşamalı MinMin)

SSS : Site Scheduling Service (Site Çizelgeleme Servisi)

UFP : Unsplittable Flow Problem (Bölünemez Akış Problemi)

WLCG : Worldwide LHC Computing Grid (Evrensel LHC İşlem Gridi)

WMS : Workload Management System (İş Yükü Yönetim Sistemi)

XML : Extensible Markup Language (Genişletilebilir İşaretleme Dili)

TERİMLER DİZİNİ

ad uzayı	: namespace
ağ	: network
ağ elemanı	: network element
ağaç	: tree
alıt	: domain
bağımsız işler	: independent tasks
benzetim yapmak	: simulate
benzetimli tavlama	: simulated annealing
bilgi servisleri	: information services
birli	: monadic
çekirdek	: core
çizelgeleme servisleri	: scheduling services
dağıtık	: distributed
data manager	: veri yöneticisi
depolama elemanı	: storage element
Veri transferi servisleri	: data transfer services
en-iyi-girişim	: best-effort
eşzamanlı	: synchronous
eşzamansız	: asynchronous
geçit	: gateway
gerçek-zamanlı	: real-time
gevşek birliktelikli	: loosely coupled
hizmet nitelikleri	: Quality of Service
istek paketleme	: request packing
iş akışları	: workflows
iş başlatma servisleri	: job invoke managers
iş dağıtım yöneticileri	: job dispatch managers
iş parçacığı	: thread
iş sunum servisleri	: job submission services
iş torbası	: bag of tasks

işbirlikçi	: collaborative
işlem elemanı	: computing element
işlemci	: processor
kabul kontrolü	: admission control
karınca kolonisi	: ant colony
karmaşıklık	: complexity
kimlik denetlenmiş	: authenticated
kopya kataloğu	: replica catalogue
kopya konum göstergesi	: replica location index
kopya konum servisleri	: replica location services
kullanma durumları	: use cases
küme	: cluster
mekansal	: spatial
melez	: hybrid
noktadan noktaya	: peer-to-peer
olay	: event
olay güdümlü	: event-driven
ortanca	: *p*-median
özel	: custom
rezervasyon servisleri	: reservation services
rulet tekerleği	: roulette wheel
sanal organizasyon	: virtual organization
seçme	: selection
son zaman	: deadline
sunucu	: server
süreç	: process
süreç tabanlı	: process-oriented
şeffaflık	: transparency
taşınabilirlik	: portability
yararlanılırlık	: availability
yasaklı arama	: tabu search
yerellik	: locality

yönetilmiş	: managed
yönlendirme	: routing
tanelilik	: granularity
taşma	: flooding
uzay-paylaşımlı	: space-shared
vektör dönüştürücü	: vector converting
veriler hakkında bilgiler	: metadata
veri yönetim servisleri	: data management services
veri kopyalama servisleri	: data replication services
zaman-paylaşımlı	: time-shared
zamansal	: temporal

1. GİRİŞ

Tüm bilim dallarında önemli gelişmelerin kaydedildiği bir zaman aralığındayız. Bu gelişmelerin önemli bir bölümü de bilgi ve iletişim teknolojileri alanında yaşanmaktadır: Kişisel bilgisayarlar, piyasaya ilk sürülmelerinden daha 50 yıl bile geçmeden, hayatımızın en önemli parçası oldular. Veri iletişim altyapısının geniş coğrafi alanlara yayılması ile Internet'in kişisel bilgisayarlar ve mobil iletişim araçları arasında kullanımı olabildiğince yaygınlaştı. Günümüzde, bu ve benzeri gelişmelerin ışığında, işlemciler, bilgisayarlar ve bilgisayar kümeleri arası işbirlikleri kurarak; binlerce işlemciyi ortak bir uygulama çatısı altında çalıştırabilen sistemler geliştirilebilmektedir.

Pahalı olmayan kişisel bilgisayarların ve Internet'in yaygınlaşması ile dikkatler, Dünya'nın farklı noktalarında yer alan işlem ve veri depolama elemanlarının ağ yapıları üzerinden paylaşılmasını sağlayan Grid sistemleri üzerine yoğunlaşmıştır. Grid sistemleri, kendilerine arz edilen işleri, coğrafik olarak farklı bölgelerde yer alan kaynakları koordineli bir şekilde kullanarak çalıştırmak amacı ile oluşturulmuş olan ve yazılım/donanım unsurları içeren sistemlerdir. Grid sistemlerinin kendine has özelliklerden bazıları şunlardır [1]:

1. Grid sisteminde yer alan kaynaklar ve kullanıcılar, farklı kontrol alanları içerisinde yer alırlar.
2. Sistem üzerinde bulunan kaynakların paylaşımı için, merkezi bir kontrol ve koordinasyon merkezi zorunlu değildir.
3. Kullanıcılar, sistem bünyesindeki kaynaklara, dağıtık protokoller ve mekanizmalar aracılığı ile erişirler.
4. Söz konusu protokoller ve yazılımlar, genel kullanıma açıktır.
5. Dünya'nın farklı bölgelerinden kaynaklar barındıran Grid sisteminin sürdürülebilir performans gösterebilmesi için, Grid altyapısı yeterli hizmet kalitesini sunacak şekilde gerçeklenir.
6. Sistem bileşenleri, kullanıcılara, güvenli ve basit arayüzler ile sunulurlar.
7. Kullanıcıların, sistemin detayları hakkında bilgi sahibi olmasına gerek yoktur. İşler, Grid sistemine, sanal bir süperbilgisayara arz edilir gibi

sunulur. İşlerin hangi kaynakları kullandığı da kullanıcı tarafından bilinmez.

8. Kaynaklar sisteme dinamik olarak dâhil olup çıkabilirler. Kaynaklar, bu özelliğe sahip olabilmek için, sistem tarafından tanımlanmış olan bir protokolü kullanırlar.
9. Grid altyapısının kaynakların kontrolünü sağlayabilmesi için, kaynaklar, durumlarındaki değişiklikleri Grid kontrol mekanizmasına bildirirler.

Grid sistemlerinin sağladığı altyapı sayesinde, büyük ölçekli işler için çok pahalı olan süperbilgisayar sistemlerinin kullanılması yerine, farklı bölgelerde atıl durumda olan bilgisayar sistemleri üzerinde işlerin bölüştürülmesi ve paralel çalıştırılması tercih edilir olmuştur [1]. Bu nedenle, geleneksel olarak Grid sistemlerinde işlem yoğun uygulamaların çalıştırılmasına ağırlık verilmiştir. Legion [2], Globus [3], Nimrod-G [4], Apples [5], SETI@home [6] ve Condor-G [7] gibi projeler, işlem yoğun problemlerin çözümü için tasarlanmışlardır.

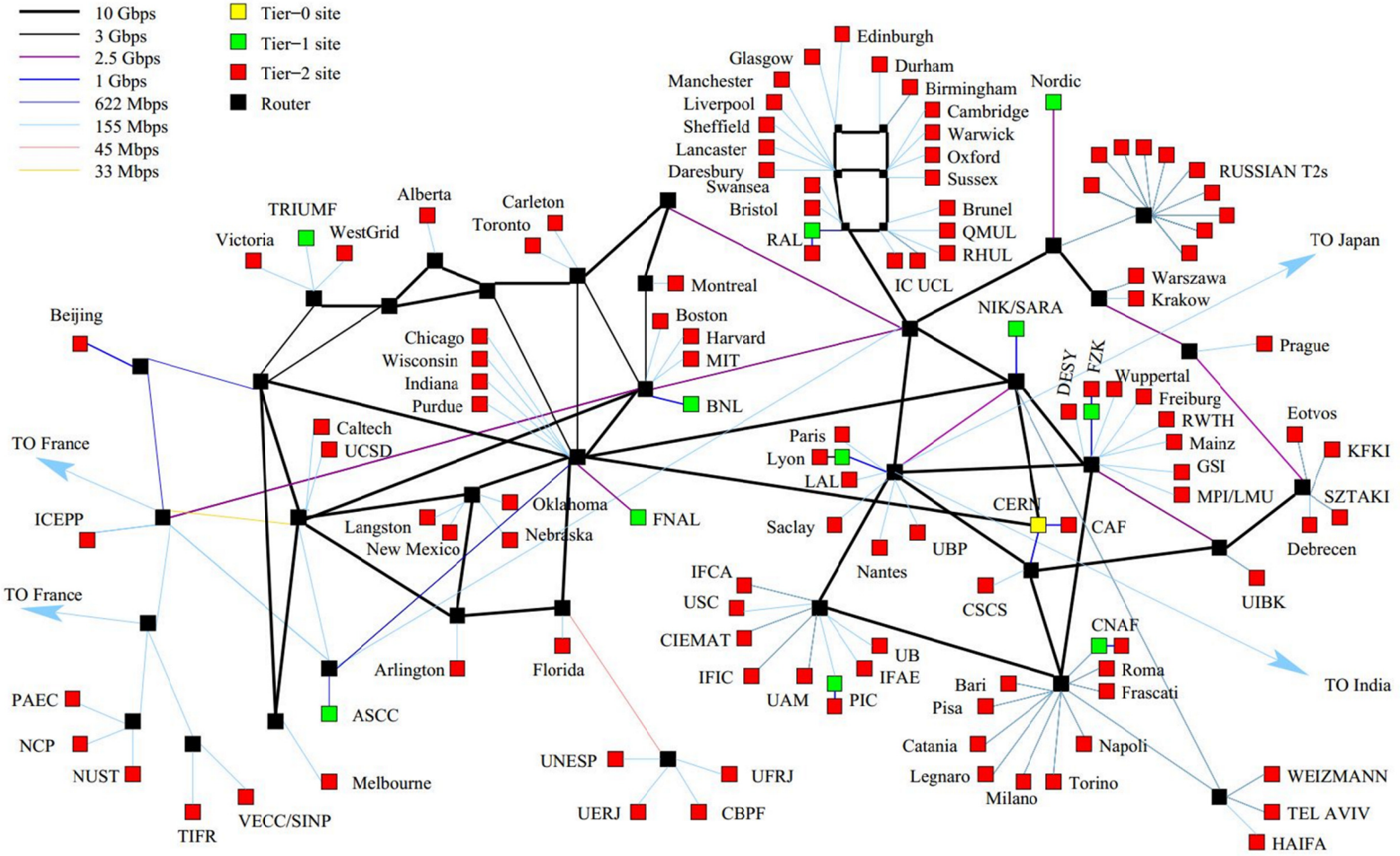
Günümüzde yüksek enerji fiziği [8, 9], klimatoloji [10], jeoloji [11], astronomi [12], protein simülasyonu [13] ve benzeri bilim dallarındaki uygulamalar, terabayt/petabayt ölçeğindeki veri kümelerine coğrafi konumları farklı kullanıcıların erişimini; verinin coğrafi konumları farklı ağ, hesaplama, veri depolama ve görselleştirme kaynakları kullanılarak yüksek başarımlı bir şekilde analizini gerektirmektedir. Benzeri uygulamaların çoğalması ile yüksek miktardaki verinin saklanmasını, transferini ve yönetimini gerektiren Grid sistemleri, önemli bir araştırma alanı olmuştur [14].

Bu gereksinimler temel alınarak Veri Grid (Data Grid) kavramı ve bileşenleri ilk olarak [15]'te tanımlanmıştır. Günümüzde, çeşitli ülkelerde Veri Grid projeleri yürütülmektedir. Veri Grid sistemleri üzerine tamamlanmış veya devam etmekte olan projelerden bazıları Çizelge 1.1'de sunulmuştur. Çizelgede de görüldüğü üzere, Veri Grid sistemleri, farklı bilim dallarında çalışma yapan bilim insanlarına, araştırmalarında ihtiyaç duydukları yüksek performanslı hesaplama, veri depolama, iletişim, vb. kaynaklara ulaşmalarını sağlayan bir donanım ve yazılım altyapısı sunmayı hedeflemektedir.

Çizelge 1.1. Dünya üzerindeki Veri Grid projeleri

Projenin İsmi	**İlgili Alanlar**	**Kapsamı**	**Destekleyen Kuruluş**
EU-DataGrid [9]	Yüksek enerji fiziği, biyoloji, tıp, astronomi	Avrupa Birliği	Avrupa Birliği
GridPP [16]	Yüksek enerji fiziği	İngiltere	İngiltere Hükümeti
Earth System Grid [17]	İklim modelleme	ABD	DOE Bilim Ofisi, ABD
Worldwide LCG Computing Grid (WLCG) [18]	Yüksek Enerji Fiziği	Global	CERN
iVDGL [19]	Yüksek enerji fiziği, astronomi, bioenformatik	Global	NSF, ABD
EGEE [20]	Yüksek enerji fiziği, biomedikal, astrofizik, hesaplamalı kimya	Global	Avrupa Birliği
BioGrid [13]	Tıp, biyoloji	Japonya	Japonya Hükümeti
NEESit [21]	Deprem	ABD	NSF, ABD
eDiaMoND [22]	Meme kanseri	İngiltere	İngiltere Hükümeti

Avrupa Birliği fonları ile kurulan ve CERN öncülüğünde devam eden Büyük Hadron Çarpıştırıcısı projesi, özellikle yüksek enerji fiziği, Dünya bilimleri ve bioenformatik alanlarında yüksek işlem gücü ve veri depolama/paylaşma kapasitesi gerektiren işlerin çalıştırıldığı Evrensel LHC İşlem Gridi (Worldwide LHC Computing Grid - WLCG) içermektedir. WLCG sisteminin altyapısı, örnek bir Veri Grid sistemi olarak, Şekil 1.1'de gösterilmiştir.



Şekil 1.1. WLCG sistemi [18]

Büyük Hadron Çarpıştırıcısı üzerinde dört farklı deney için yerleştirilmiş olan dört adet algılayıcı bulunmaktadır: ALICE, ATLAS, CMS ve LHCb. Bu algılayıcılardan sadece Kompak Selenoid Algılayıcı (Compact Selenoidal Detector - CMS) yılda yaklaşık 1 petabayt veri üretmektedir [23]. Bu veriler, farklı bölgelerdeki uygulamalarda kullanılmak üzere CERN'de ve bazı yerel kopya bölgelerinde saklanmakta ve Dünya'nın farklı bölgelerinde bulunan bilimsel çalışma grupları tarafından analiz edilmektedir. WLCG sisteminde verilerin verimli, hızlı ve güvenli bir şekilde kullanıcılara ulaştırılması, sistemin birincil önceliğidir.

WLCG benzeri bir Veri Grid sisteminin kullanıcılarına sağlamış olduğu servislerden bazıları şunlardır:

1. Yüksek miktarda verinin depolanması için veri depolama elemanları sağlamak.
2. Verilerin, sistem üzerinde birden fazla kopyasının bulundurulmasına olanak tanıyacak veri kopyalama mekanizmaları çalıştırmak.
3. Sistem üzerindeki verilerin adreslenebileceği global bir isim uzayı tanımlamak.
4. Veri depolama elemanlarının, istekte bulunulan veri için sorgulanmasına olanak tanımak.
5. Veriye erişim için gerekli olan kullanıcı erişim izinlerinin olup/olmadığını denetlemek.
6. Veri transferlerinde sistem kaynaklarını verimli kullanabilmek için veri dağıtımı mekanizmaları çalıştırmak.
7. Yüksek miktarda veri transferi ve işlem gücü gerektiren işleri sisteme arz etmek, işlem elemanları üzerinde çizelgelemek ve çalıştırmak.

Bazı Grid uygulamalarında, işlerin tamamlanmış olmaları yeterli görülmez. Kullanıcının istediği bazı servis kalitesi şartları da sağlanmalıdır. En çok kullanılan servis kalitesi şartlarından birisi uygulamalar için son zaman tanımlanmasıdır. Bu çalışmada, son zaman servis kalitesi içeren işlere *gerçek-zamanlı iş*; en az bir adet *gerçek-zamanlı iş* barındıran uygulamalara da *gerçek-zamanlı uygulama* adı verilecektir. Gerçek-zamanlı işlerin veya uygulamaların

başarı ile tamamlanmış sayılabilmeleri için, son zamanlarından önce sonlanmış olmaları gerekmektedir.

Gerçek-zamanlı işlerin çalıştırıldığı Veri Grid sistemlerinde, çizelgeleyici algoritmaların hedefi, mümkün olan en fazla sayıda gerçek-zamanlı işin *başarılı* biçimde tamamlanmasıdır. Bu durumda, Veri Grid sisteminin *gerçek-zaman performansı*, başarılı şekilde tamamlanmış işlerin, toplam işlere oranı olarak tanımlanır. Veri Grid sistemlerinin gerçek-zaman performansını eniyilemek için, iş çizelgeleme, veri dağıtımı ve veri kopyalama gibi operasyonların ve işlem, ağ ve veri depolama elemanlarını kontrol eden yöneticilerin, servis kalitesi gereksinimlerini sağlayacak şekilde tasarlanmış olması gerekmektedir.

Bu çalışmanın 2. Bölüm'ünde, gerçek-zamanlı uygulamaların çalıştırılmasını ve gerçek-zamanlı veri transferi isteklerinin karşılanmasını destekleyen bir Veri Grid sistemi modeli tanıtılacaktır. Önerilen Veri Grid sistemi modeli, Grid sistemleri için standart servisler tanımlayan Açık Grid Servisleri Mimarisi (Open Grid Services Architecture - OGSA)'nin ve Grid sistemlerinde dağıtık kaynak paylaşımında kullanılan açık kaynak kodlu yazılım platformu Globus'un incelenmesi neticesinde ortaya çıkarılmıştır. Önerilen modelde, servislerin rol ve sorumlulukları ile etkileşimleri detaylı bir şekilde tanımlanmıştır. Model, literatürdeki hiyerarşik iş çizelgeleme, hiyerarşik veri dağıtımı ile farklı veri kopyalama yaklaşımlarını desteklemekte olup, Veri Grid sistemi üzerindeki tüm kaynakların önceden rezerve edilebildiği bir ön rezervasyon sistemini de içermektedir.

2. Bölüm'de tanıtılan Veri Grid sistemi modeli için, nesne tabanlı programlama tekniklerine uygun olarak tasarlanmış olan ve DGridSim adı verilen bir Veri Grid simülatörü geliştirilmiştir. Bu çalışmanın 3. Bölüm'ünde tanıtılan DGridSim, tüm işlem, veri depolama ve ağ kaynakları için önceden rezervasyon özelliğini desteklemektedir. Dolayısıyla, DGridSim'de tüm iş çizelgeleme, veri dağıtımı ve veri kopyalama algoritmaları bu ön rezervasyon yapısının üzerinde bulunmaktadır. Literatürde ön rezervasyon temelinde iş çizelgeleme, veri dağıtımı ve veri kopyalama algoritmalarının simülasyonunu gerçekleştirebilecek yetenekte diğer bir simülatör bulunmamaktadır. DGridSim son derece modüler ve esnek bir

yapıda tasarlanmış olup, yeni iş çizelgeleme, veri dağıtımı ve veri kopyalama algoritmalarının da eklenmesine uygundur.

Veri Grid sistemine arz edilen gerçek-zamanlı işlerin çalıştırılabilmesi için, işin ihtiyaç duyduğu verilerin, işin başlangıç zamanından önce işin üzerinde çalışacağı işlem elemanına transfer edilmiş olması gerekmektedir. Bu çalışmada, son zaman servis kalitesi içeren veri transferi isteklerine *gerçek-zamanlı veri transferi isteği* adı verilecektir. Gerçek-zamanlı veri transferi isteklerinin başarılı sayılabilmesi için, kendileri için belirlenen son zamanlarından önce tamamlanmaları gerekmektedir. Bu çalışmanın 4. Bölüm'ünde gerçek-zamanlı veri transferi isteklerinin performansını eniyilemek için gerçek-zamanlı veri dağıtımı problemi incelenmiştir. Öncelikle, veri transferlerinin tek rota üzerinden dağıtıldığı varsayılmıştır. Tek rotadan veri dağıtımı probleminin biçimsel tanımlaması yapılmış, keşifsel algoritmalar önerilmiş ve algoritmaların performans ölçüm sonuçları sunulmuştur. Daha sonra, veri transferlerinin birden fazla rota üzerinden gerçekleştirildiği çoklu rotadan veri dağıtımı problemi biçimsel olarak tanımlanmış, keşifsel algoritmalar önerilmiş ve algoritmaların performans ölçümleri yapılmıştır.

Veri kopyalama algoritmaları, Veri Grid sistemlerinin önemli bir bileşenidir. Verilerin, işlerin çalıştırılacağı sitelere yaklaştırılmasını sağlayan veri kopyalama algoritmaları, hem işlerin bitiş zamanlarını hem de sistemde gereksinim duyulan bant genişliğini önemli ölçüde azaltırlar. Veri kopyalama algoritmaları, kopyalanacak veriyi ve verinin kopyalanacağı Grid sitesini seçerken, geçmişteki veri istekleriyle ilişkilendirilmiş istek sıklığı, ortalama istek son zamanı ve benzeri çeşitli istatistiklerden faydalanmaktadır. Bu çalışmanın 5. Bölüm'ünde, Veri Grid sistemlerinin gerçek-zaman performansını eniyilemek için önerilen veri kopyalama algoritmaları ve bu algoritmaların sistemin gerçek-zaman performansı üzerindeki etkilerini gösteren simülasyon sonuçları yer almaktadır.

Çalışmanın son bölümü olan 6. Bölüm'ünde ise, sonuçlar ve gelecekte yapılması planlanan çalışmalar belirtilmiştir.

2. GERÇEK-ZAMANLI VERİ GRİD SİSTEMİ MODELLEMESİ

Veri Grid sistemleri ile ilgili literatürdeki çalışmalarda, sisteme arz edilen işlerin mümkün olan en kısa zamanda tamamlanmasının amaçlandığı *en-iyi-girişim* (*best-effort*) sistemler üzerinde yoğunlaşılmıştır. Ancak, son yıllarda, gerçek-zamanlı uygulamaların sayısı artmaktadır. *Gerçek-zamanlı uygulamalar*, genellikle, bir veya birden fazla gerçek-zamanlı iş ve işlerle ilişkilendirilmiş veri transferlerinden oluşmaktadır. Gerçek-zamanlı uygulamaların başarılı sayılabilmeleri için belirlenmiş bir son zamandan önce tamamlanmış olmaları gerekmektedir. Dolayısıyla, gerçek-zamanlı bir uygulamanın başarılı olabilmesi için; uygulamayı oluşturan gerçek-zamanlı işlerin ilgili son zamanlarından önce tamamlanmaları; işlerin başarılı olabilmesi içinse veri transferlerinin belirli son zamanlarından önce bitirilmeleri gerekmektedir.

Bu çalışmada, gerçek-zamanlı uygulamaların üzerinde çalıştığı ve başarı ile tamamlanan gerçek-zamanlı uygulama sayısınını eniyilemek için tasarlanmış sistemlere *gerçek-zamanlı Veri Grid sistemleri* adı verilmektedir. Literatürde gerçek-zamanlı Veri Grid sistemleri için önerilmiş bir model bulunmamaktadır. Bu çalışma kapsamında, literatürdeki bu eksiklik ilk olarak doldurulmaya çalışılmıştır.

Önerilen gerçek-zamanlı Veri Grid sistemi modeli, dört temel perspektifi içerir: 1) *İş çizelgeleme*: Veri Grid sistemine arz edilen gerçek-zamanlı işleri, siteler üzerinde bulunan işlem elemanları üzerinde çizelgeleyebilecek servisler bulunmalıdır. 2) *Veri dağıtımı*: İş çizelgeleme algoritmaları tarafından işlem elemanları üzerine çizelgelenen işlerin ihtiyaç duyduğu verilerin, işin başlangıç zamanından önce işlem elemanına transfer edilebilmesi için gerekli servisler tanımlanmış olmalıdır. 3) *Veri kopyalama*: İş çizelgeleme ve veri dağıtımı algoritmalarının performanslarının artması için, verileri, farklı Veri Grid sitelerinde kopyalayacak mekanizmalar bulunmalıdır. 4) *Ön rezervasyon*: Gerçek-zamanlı işlere servis kalitesi sunabilmek için iş çizelgeleme, veri dağıtımı ve veri kopyalama algoritmalarının kontrolünde sistem kaynaklarını önceden rezerve edebilecek bir yapı olmalıdır.

Önerilen model bünyesinde, söz konusu perspektifleri gerçekleştirecek biçimde iş çizelgeleme, veri dağıtımı, veri kopyalama ve rezervasyon servisleri tanımlanmış ve servisler arası etkileşimler organize edimiştir. Önerilen modelde, iş çizelgeleme için hiyerarşik iş çizelgeleme ve veri dağıtımı için hiyerarşik veri dağıtımı yaklaşımları benimsenmiştir. Ayrıca, önerilen model, literatürde bulunan çekme ve itme tabanlı veri kopyalama modellerinin gerçeklenmesi için uygundur. Önerilen modelde, Veri Grid sistemi üzerindeki kaynakların önceden rezerve edilebilmesini sağlayan ön rezervasyon sistemi de bulunmaktadır.

2.1. İlgili Çalışmalar

Açık Grid Forumu (Open Grid Forum – OGF), Grid sistemleri için standartlar tanımlamak amacıyla kullanıcılar, geliştiriciler ve üreticilerin bir araya gelmesi ile oluşturulmuş bir platformdur. Bu platformun çalışmaları neticesinde, Grid sistemleri için kullanılabilecek GridFTP, JSDL gibi standartlar geliştirilmiştir.

OGF ve benzeri grupların çalışmaları neticesinde, Grid sistemlerinin altyapılarını tanımlayan birçok standart oluşturulmuştur. Bunlardan en çok bilineni Açık Grid Servisleri Mimarisi (Open Grid Services Architecture - OGSA) standardıdır [24]. Açık Grid Forumu tarafından [25]'te temelleri atılan OGSA, heterojen sistemlerde bulunan farklı özellikteki kaynakların haberleşmesini ve bilgi paylaşımını sağlayan, servis tabanlı bir mimaridir.

OGSA, Grid üzerinde kullanıcı ve uygulamaların kullanabilecekleri standart servisler tanımlamaktır. Bu standart genel olarak aşağıdaki bileşenleri içerir:

1. Programlar arası arayüz ve mesajlaşma için Basit Nesne Erişim Protokolü (Simple Object Access Protocol - SOAP).
2. Veri paylaşımı için Genişletilebilir İşaretleme Dili (Extensible Markup Language - XML).
3. İş yükü yönetimi için WS-Management.
4. Diğer ilgili tanımlamalar.

OGSA standartlarını kullanan ara yazılımlar geliştirebilmek için Açık Grid Servisleri Altyapısı (Open Grid Services Infrastructure - OGSI) tanımlanmıştır. OGSI'nın çok kullanılan bir gerçeklemesi de Globus yazılım setidir [3]. Globus

seti ile sunulan yazılım araçları, yeni Grid sistemleri ve bu sistemlerde çalışacak uygulamaların geliştirilmesi için kullanılmaktadır. Globus yazılım setinde üç ana bileşen vardır: kaynak yönetimi, bilgi servisleri ve veri yönetimi. Globus bünyesinde; kaynak yönetimi için Globus Kaynak Paylaştırma Yöneticisi (Globus Resource Allocation Manager - GRAM) [3], bilgi servisleri için İzleme ve Bulma Servisi (Metacomputing Directory Service - MDS) [26] ve veri transferleri için de GridFTP [27] bileşenleri bulunmaktadır. Bu bileşenler, haberleşme için Grid Güvenlik Altyapısı (Grid Security Infrastructure - GSI) güvenlik protokolünden istifade ederler.

2.1.1. İş çizelgeleme

Gerçek-zamanlı veri yoğun işlerin Veri Grid kaynakları üzerine, son zamanlarından önce tamamlanacak şekilde çizelgelenmeleri zor bir problemdir. Bunun sebebi şöyle özetlenebilir:

1. Yeterli işlem kapasitesi olan Veri Grid sitesi veya siteleri belirlenmelidir.
2. Yeterli okuma/yazma bant genişliği ve depolama alanı bulunan veri depolama kaynakları bulunmalıdır.
3. Terabaytlar mertebesinde verilerin transferi için ağ elemanları tahsis edilmelidir.

İş çizelgeleme yaklaşımları üç ana gruba ayrılır: Hiyerarşik, merkezi ve dağıtık. Hiyerarşik iş çizelgeleme de, Grid sistemine gelen işleri sitelere çizelgeleyen merkezi bir servis bulunur. Sitelerde ise işleri, site içindeki işlem elemanlarına çizelgeleyen yerel servisler bulunur. Merkezi ve yerel çizelgeleme servislerinde, farklı çizelgeleme algoritmaları çalışabilir. Darwin [28], Nimrod-G [4], Calana [29], VIOLA [30], Gallop [31] ve GridFlow[32] hiyerarşik iş çizelgeleme örnekleri arasında sayılabilir.

Merkezi iş çizelgeleme de ise, Grid sistemi işlem, veri depolama ve ağ kaynakları için gerekli olan tüm çizelgelemeleri yapan merkezi bir servis bulunur. Bu sistemlerde çizelgeleyici, tüm Grid sisteminin anlık durumunu göz önünde bulundurarak, işleri işlem elemanlarına çizelgeler. Dolayısıyla, merkezi iş çizelgelemenin, hiyerarşik ve dağıtık iş çizelgeleme yaklaşımlarıyla

karşılaştırıldığında, sistem performansını eniyilemek açısından daha iyi sonuçlar vermesi beklenir. Ancak merkezi iş çizelgeleme ölçeklenebilir değildir. Ayrıca, merkezi çizelgeleyici servisinin çalışmasını durdurması, tüm çizelgeleme sisteminin çökmesi ile sonuçlanır. Merkezi iş çizelgelemeye örnek olarak Condor [33], GridWay [34], UNICORE [35] ve GrADS [36] gösterilebilir.

Dağıtık çizelgeleme de ise, merkezi bir çizelgeleme servisi bulunmaz. Sitelerde bulunan tüm yerel çizelgeleyiciler, birbirleri ile haberleşerek, işlerin çizelgelenmesini gerçekleştirilirler. Bu sistemlerde, yerel çizelgeleyicilerin çalışmalarını durdurmalarının etkileri, sadece site ile sınırlı kalır. Ancak çizelgelemeler, sadece sınırlı bilgi ile yapılırlar. AppleS [5], Ninf [37], NetSolve [38] ve DRMAA [39] dağıtık iş çizelgeleme örnekleri arasında sayılabilir.

2.1.2. Veri dağıtımı

Çok sayıdaki kullanıcı Veri Grid sistemi üzerinde gerçek-zamanlı uygulamalarını çalıştırırken; uygulamaların başarılı sayılabilmesi için sistemin bu uygulamalar ile ilişkili veri transferi isteklerini son zamanlarından önce tamamlaması gerekmektedir. Mümkün olan en fazla sayıda veri transferi isteğini son zamanından önce tamamlayabilmek için, veri transferlerinin bir veri dağıtımı algoritması kullanılarak çizelgelenmesi gerekmektedir.

Veri dağıtımı algoritmaları, tekli rotadan veri dağıtımı algoritmaları ve çoklu rotadan veri dağıtımı algoritmaları olarak ikiye ayrılabilirler. Tekli rotadan veri dağıtımı algoritmaları, veri transferi isteklerinin tek bir rota üzerinden karşılandığını varsayarlar. Literatürdeki tekli rotadan veri dağıtımı algoritmaları, en-iyi-girişim sistemler üzerine yoğunlaşmıştır. Bu algoritmalarda hedef, veri transferi isteklerinin, mümkün olan en kısa zamanda bitirilmesidir. Bu algoritmalara örnek olarak, Dijkstra [40] ve Bellman-Ford [41] en kısa yol algoritmaları tabanlı yaklaşımlar olan Yönlendirme Bilgi Protokolü (Routing Information Protocol - RIP) [42], OSPF [43] ve Sınır Geçit Protokolü (Border Gateway Protocol – BGP) [44] sayılabilir. Literatürde, veri transferlerine son zaman ve en fazla rota gecikmesi gibi servis kalitesi destekleri sunan çalışmalar da mevcuttur [45, 46].

Çoklu rotadan veri dağıtımı algoritmaları ise, veri transferi isteklerinin birden fazla rota üzerinden karşılandığını varsayar. [47]'de, birden fazla rota üzerinden veri dağıtımı algoritması önerilmiştir. Bu alandaki çalışmalar [48] ile hız kazanmıştır. Ancak, gerçek-zamanlı veri transferi isteklerinin birden fazla rota üzerinde dağıtılması ile ilgili literatürde çalışma bulunmamaktadır.

2.1.3. Veri kopyalama

Verileri Grid sistemi üzerindeki farklı yerlerde bulunan veri depolama elemanlarına kopyalamanın işlerin tamamlanma zamanlarını ve veri transferleri sırasında ihtiyaç duyulan bant genişliği/depolama alanı gereksinimlerini azalttığını, yapılmış olan birçok çalışma göstermiştir [49, 50-52]. Son zamanlarda yapılan bir çalışma [53], veri kopyalama algoritmalarının Veri Grid sisteminin gerçek-zaman performansı üzerinde oldukça etkili olduğunu göstermiştir.

Veri kopyalama algoritmaları temel olarak iki gruba ayrılırlar: Çekme tabanlı ve itme tabanlı. Çekme tabanlı veri kopyalama algoritmalarında, veriye ihtiyacı olan site, ne zaman ve hangi siteden veriyi kopyalayacağına karar verir. İtme tabanlı veri kopyalama algoritmalarında ise, veriye sahip olan site veriyi ne zaman ve hangi siteye kopyalayacağına karar verir. Çekme tabanlı algoritmalara örnek çalışmalar [50, 51, 54]'de verilmiştir.

İtme tabanlı algoritmalar ise temel olarak iki alt gruba ayrılırlar: Dağıtık ve merkezi. İtme tabanlı dağıtık veri kopyalama yaklaşımında, tüm sitelerde bir veri kopyalama servisi bulunur. Bu servisler, çevrim içi veya çevrim dışı algoritmaları çalıştırarak, site içerisinde bulunan verilere ait veri erişimi istatistiki bilgilerini de kullanarak, site içerisindeki verilerin, Veri Grid sistemi içerisindeki diğer sitelere kopyalanmasını sağlarlar. İtme tabanlı dağıtık veri kopyalama algoritmalarına örnek olarak [49, 55, 56] gösterilebilir.

İtme tabanlı merkezi veri kopyalama yaklaşımında ise, merkezi bir veri kopyalama servisi, Veri Grid sistemi içerisindeki tüm veri erişimi istatistiklerini tutar. Çevrim dışı çalışan merkezi veri kopyalama servisi, veri erişim istatistiklerini de kullanarak, Veri Grid sistemi içerisindeki sitelerde bulunan verilerin, diğer sitelerde kopyalanmasını kontrol eder. Bu yaklaşıma örnek olarak [52, 57, 58] çalışmaları verilebilir.

2.1.4. Ön rezervasyon

Veri Grid sistemlerinde öngörülebilir bir performans sunabilmek için, uygulamaların çalışmaları esnasında kullanılacak olan sistem üzerindeki işlemci, veri depolama ve ağ kaynakları önceden rezerve edilebilir olmalıdırlar. Kaynakların önceden rezerve edilmesi sayesinde, gerektiği zaman uygulamaların kaynaklara erişimi garanti altına alınmış olmaktadır. Bu durum, son zaman gibi servis kalitesi gereksinimlerinin karşılanması için önemlidir. Ancak ön rezervasyon sistemleri iyi tasarlanmadığında, sistem verimliliğinde önemli kayıplara neden olabilmektedirler.

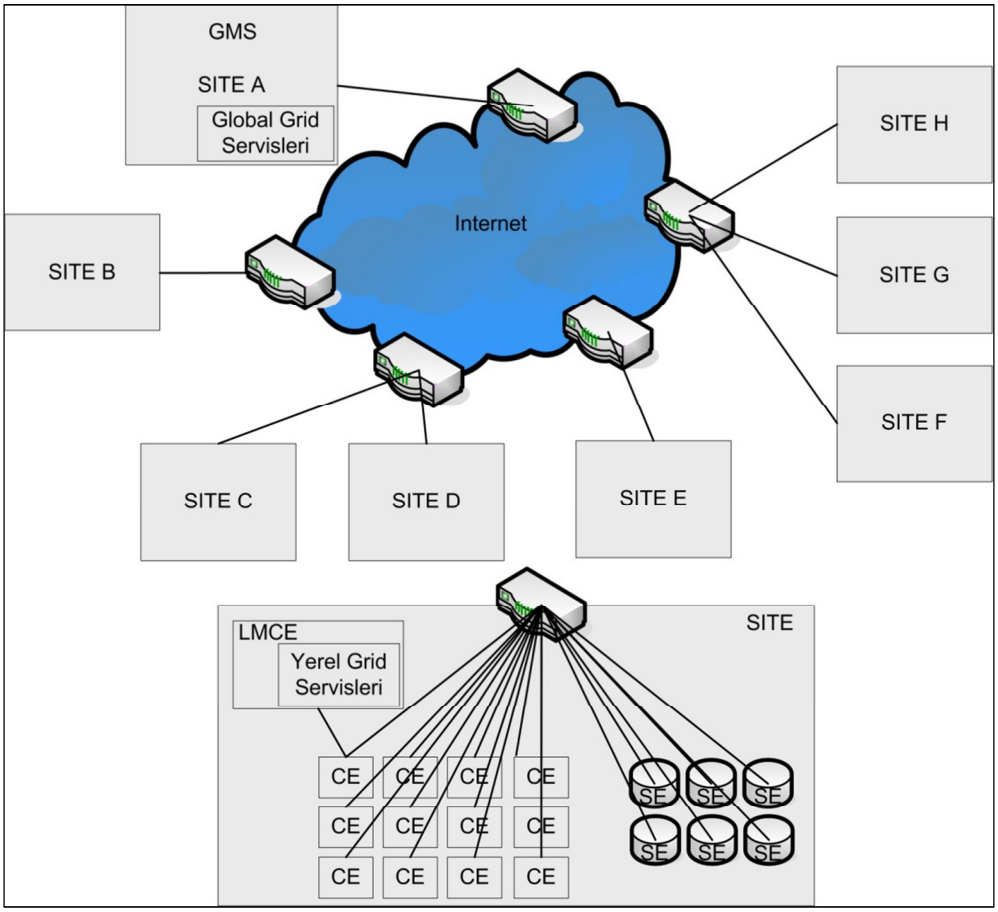
Ön rezervasyon sisteminde, her sistem kaynağı ile ilgili rezervasyonların tutulduğu bir rezervasyon tablosu bulunur. Rezervasyon tablolarında, kaynağın sağladığı kapasitenin tamamı veya bir kısmı belirli zaman aralıklarında rezerve edilir. Eğer kaynağın sağladığı kapasitenin belirli bir oranı rezerve edilebiliyorsa, kaynak *zaman-paylaşımlı* (time-shared), bir kerede sadece tamamı rezerve edilebiliyor ise *uzay-paylaşımlı* (space-shared) adı verilir.

Literatürden ön rezervasyon sistemine örnek olarak GARA [59] sistemi gösterilebilir. Bu sistemde tüm kaynak çeşitleri için ortak mekanizmalar oluşturulmuştur. İşlemci demetleri ve süperbilgisayarlar tarafından kullanılan Maui [60] iş çizelgeleyicisinde de ön rezervasyon desteği bulunmaktadır. Son zamanlarda önerilmiş olan Grid Önceden Rezervasyon Sırası (Grid Advance Reservation Queue - GarQ) [61] ön rezervasyon sisteminde ise her kaynak için ayrı bir rezervasyon ünitesi tanımlanmıştır. Kaynak için oluşturulmuş olan rezervasyon ünitesi, rezervasyon durumu bilgi isteklerini ve rezervasyon oluşturma isteklerini karşılar.

2.2. Gerçek-Zamanlı Veri Grid Sistemi Modeli

Şekil 2.1'de gösterildiği gibi, önerilen model, Veri Grid sistemini, internet aracılığı ile bağlanmış olan siteler olarak modellemektedir. Tüm siteler, Grid sistemi üzerindeki kaynakları yerel Grid servisleri aracılığı ile kullanırlar. Bu modele göre sitelerden bir tanesi Global Yönetim Sitesi (Global Management Site - GMS) olarak belirlenir ve üzerinde Global Grid Servisleri çalıştırır.

Sitelerde, Grid kullanıcılarına ait işlerin üzerlerinde çalışacağı İşlem Elemanı (Computing Element – CE) ve üzerlerinde işlerin gerektirdiği verilerin saklanacağı Depolama Elemanı (Storage Element – SE) bulunur. CE'lerden bir tanesi Yerel Yönetim İşlem Elemanı (Local Management CE - LMCE) olarak belirlenir ve üzerinde Yerel Grid Servisleri çalıştırır.



Şekil 2.1. Veri Grid sistemi modeli

Bu modele göre bir site, ağa bağlı veri depolama sitesi (hiç bir işlem elemanı bulunmayan site), ağa bağlı işlem elemanları sitesi (hiç bir veri depolama elemanı bulunmayan site) ve ağa bağlı işlem elemanları ve veri depolama sitesi gibi farklı yapıları destekleyecek biçimde modellenebilmektedir. Tüm siteler site içindeki tüm işlem ve veri depolama elemanlarının direk linkler ile bağlandığı bir adet geçit (gateway) yönlendirici ile sistemin geri kalanına bağlanırlar. Her bir sitedeki işlem ve veri depolama elemanları geçit yönlendirici aracılığıyla

birbirleriyle ve diğer Grid sistemi bileşenleriyle haberleşebilmektedir. Şekil 2.2'de farklı site örnekleri gösterilmiştir.

(a) **(b)** **(c)**

Şekil 2.2. Farklı site örnekleri (a) bir ağ veri depolama birimi (b) bir işlemci ve bir veri depolama birimi (c) bir işlemci öbeği ve ortak veri depolama birimi

2.2.1. Temel Grid Altyapısı

Önerilen modeldeki temel Grid yapısı, üç temel donanımsal elemandan oluşmaktadır: depolama elemanı, işlem elemanı ve ağ elemanı. Elemanların arayüzleri, OGSA [24] standardında belirtilen talimatlara uygun olarak oluşturulmuştur.

2.2.1.1. Veri depolama elemanı

Depolama elemanı, Grid sistemi üzerinde verilerin depolanmasını sağlayan temel birimdir. Grid sistemi üzerinde kullanılan depolama elemanları, tek bir sabit disk olabileceği gibi, bir sabit diskler kümesi de olabilmektedir. Depolama elemanları, veriler ile birlikte veri hakkındaki bilgileri de saklarlar. Depolama elemanları, depolama kapasitesi ve okuma/yazma hızları ile modellenirler.

Veri depolama elemanlarının okuma ve yazma bant genişlikleri, rezervasyon servisleri tarafından rezerve edilebilmektedir. Söz konusu rezervasyonların gerçekleştirilmesi için rezervasyon servislerine *REZERVASYON_OLUŞTUR* := [*Kaynak*, *RezervasyonBaşlangıçZamanı*, *RezervasyonBitişZamanı*, *RezervasyonBüyüklüğü*] biçiminde mesajlar gönderilir. Bu mesajdaki *Kaynak*, *VeriDepolamaElemanı*:*Okuma* veya *VeriDepolamaElemanı*:*Yazma* olur.

Veri depolama elemanları, veri depolama elemanı yöneticisi tarafından yönetilirler. Veri depolama elemanı yöneticileri, yeni gelen verilerin kopyalanmaları, verilerin depolama alanından silinmeleri, yeni kopyalanacak veri için yeterli alan bulunmadığı zaman silinecek verilerin karar verildiği *veri yenileme algoritmaları*nın (Least Recently Used ve benzeri) çalıştırılması gibi fonksiyonları gerçekleştirirler.

Veri depolama elemanı yöneticileri veri yenileme algoritmaları tarafından silinmesi uygun olmayan verilerin kilitlenmesi için *VERİ_KİLİTLE* := [*Veri*, *KilitBaşlangıçZamanı*, *KilitBitişZamanı*] biçiminde gönderilen mesajları alırlar ve *Veri*'nin *KilitBaşlangıçZamanı* ile *KilitBitişZamanı* zamanları arasında silinmelerini engellemek için, verileri ilgili zaman aralıklarında kilitlerler. Eğer işlem başarılı ise, *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi*], başarısız ise de *VERİ_KİLİT_HATA* mesajlarını geri iletirler. Veri depolama elemanı yöneticisine *VERİ_KİLİT_SİL* := [*VeriKilitBilgisi*] mesajı gelmesi durumunda, veri kilit bilgisi verilmiş olan kilit kaldırılır.

2.2.1.2. İşlem elemanı

İşlem elemanı, Grid sistemi üzerinde işlerin çalıştırılmasını sağlayan temel birimdir. İşlem elemanları, gelen işleri çalıştırırlar. İşlem elemanları tek işlemcili olabileceği gibi, işlemciler kümesi şeklinde de yapılanmış olabilirler. Ayrıca uzay-paylaşımlı (space-shared) veya zaman-paylaşımlı (time-shared) olabilirler. Uzay paylaşımlı işlem elemanlarında, işlemci işi bir sıradan çeker ve iş bitene kadar çalıştırır. Zaman-paylaşımlı işlem elemanlarında ise işlemci sıradan bir iş çeker, önceden belirlenmiş süre kadar çalıştırır, eğer iş tamamlanmamış ise tekrar sıraya yerleştirir ve sıradaki işi alır.

İşlem elemanlarının işlem bant genişlikleri, rezervasyon servisleri tarafından rezerve edilebilmektedir. Söz konusu rezervasyonların gerçekleştirilmesi için rezervasyon servislerine *REZERVASYON_OLUŞTUR* := [*Kaynak*, *RezervasyonBaşlangıçZamanı*, *RezervasyonBitişZamanı*, *RezervasyonBüyüklüğü*] biçiminde mesajlar gönderilir. Bu mesajdaki *Kaynak*, *İşlemElemanı* olur.

Ayrıca, işlem elemanlarının, çalıştıracakları iş ile ilgili verileri işlem elemanında bulunan geçici veri depolama elemanına yazması için gerekli yazma

bant genişliği de, rezervasyon servisleri tarafından rezerve edilebilmektedir. Bu rezervasyonun gerçekleştirilmesi için rezervasyon servislerine *REZERVASYON_OLUŞTUR* := [*Kaynak*, *RezervasyonBaşlangıçZamanı*, *RezervasyonBitişZamanı*, *RezervasyonBüyüklüğü*] biçiminde mesaj gönderilir. Bu mesajdaki *Kaynak*, *İşlemDepolamaElemanı*:*Yazma* olur.

Önerilen modelde, işlemciler, uzay-paylaşımlı çalışırlar ve işlem hızı ile modellenir. Rezerve edilmiş işlem bant genişliğinin kullanılmasından, Bölüm 2.2.3.5'te belirtilen İş Başlatma Servisi (Job Invoke Manager – JIM) sorumludur.

2.2.1.3. Ağ elemanı

Veri transferleri, Grid sisteminde ağ elemanları aracılığı ile gerçekleşmektedir. Linkler pasif elemanlardır. Ancak bu çalışmada ağ elemanları, pasif olan linkleri kullanan aktif yöneticiler olarak modellenmiştir. Bu yöneticiler, Şekil 2.2'deki modelde, yönlendiriciler üzerinde çalışmaktadırlar. Bu çalışmada ağ elemanı dendiğinde, aktif olan yöneticiler ve pasif olan link bileşenlerinden oluşan tek bir eleman olarak anlaşılmalıdır. Çalışmanın geri kalan kısmında ağ elemanı ve link bu eleman anlamında kullanılacaktır. Ağ elemanları, bant genişliği ve gecikme ile modellenirler.

Ağ elemanlarının bant genişliklerinin rezerve edilebilmesi için rezervasyon servislerine *REZERVASYON_OLUŞTUR* := [*Kaynak* = *RotaÜzerindekiLink*, *RezervasyonBaşlangıçZamanı*, *RezervasyonBitişZamanı*, *RezervasyonBüyüklüğü*] biçiminde mesajlar gönderilir. Rezerve edilmiş ağ elemanı bant genişliğinin kullanılmasından, Bölüm 2.2.3.10'da belirtilen Veri Tranfer Servisi (Data Transfer Service – DTS) sorumludur.

2.2.2. İletişim

Veri Grid sistemi üzerindeki siteler birbirlerine, ağ elemanları ile bağlanmışlardır. Site içindeki geçit yönlendiriciler, internetteki köşe yönlendiricilere çift taraflı direk linkler ile bağlanırlar. Köşe yönlendiriciler de birbirlerine çekirdek (core) yönlendiriciler aracılığı ile bağlanırlar. Tanımlanan bu

yapı, günümüz Internet topolojisi ile ve literatürdeki [51] ve benzeri çalışmalar ile de uyum içerisindedir.

Veri Grid sisteminin gerçek-zamanlı işleri destekleyebilmesi için, ağ altyapısı uçtan uca gecikme garantisi verebilmelidir. Uçtan uca gecikme sınırlaması gibi servis kalitesi garantileri sunma problemi, literatürde birçok çalışmaya konu olmuştur. Internet üzerinde uçtan uca gecikme garantisi sunabilmek için, İnternet Mühendislik İş Gücü (Internet Engineering Task Force – IETF), RFC 1633 Entegre Servisler (Integrated Services – IntServ) mimarisini tanımlamıştır [63]. Daha sonra [26]'da, Internet üzerinde gecikme ve bant genişliği garantileri sunmak için gerekli yönlendirici davranışları belirtilmiştir. Ayrıca, RFC 2205 Kaynak Rezervasyon Protokolü (Resource Reservation Protocol - RSVP) [64], IntServ mimarisini tamamlayarak, rota üzerindeki yönlendiricilerde kaynak rezervasyonu yapılabilmesine olanak tanımıştır.

[26, 63, 64] ve ilgili çalışmalar baz alınarak, bu çalışmada, ağ yapısındaki herhangi bir linkin bant genişliğinin, verinin transferi boyunca rezerve edilebildiği ve önceden rezerve edilen ağ elemanlarının bir veri transfer servisi tarafından kullanılabildiği varsayılacaktır.

2.2.3. Veri Grid Servisleri

Şekil 2.1'de de gösterildiği gibi, önerilen modelde, Grid sistemi üzerinde Global Grid Servisleri'ni çalıştıran bir adet Global Yönetim Sitesi (Global Management Site – GMS) mevcuttur. Ayrıca, Grid sisteminin üzerindeki her sitede yerel Grid servislerini çalıştıran bir Yerel Yönetim İşlem Elemanı (Local Management Computing Element – LMCE) bulunmaktadır. Bu bölümde, GMS'de ve LMCE'lerde çalışan servisler anlatılacaktır.

2.2.3.1. Bilgi servisleri

Grid sistemine ait bilgileri barındıran Global Bilgi Servisi (Grid Information Service - GIS) ve bir siteye ait bilgileri barındıran Yerel Bilgi Servisi (Local Information Service - LIS) bileşenlerinden oluşur. Bu bilgiler gerek statik (ağ/depolama/işlem elemanı kapasiteleri gibi) veya gerekse dinamik (kaynakların

kullanılan bant genişlikleri/depolama alanları gibi) bilgiler olabilir. Örnekleri GRASP [63] paketindeki Information Service ve MDS [26] olarak sayılabilir.

2.2.3.2. İş sunum servisleri

Kullanıcılar tarafından Grid sistemine gönderilen işlerin kabul edildiği Grid İş Arz Servisi (Grid Job Submission Service - GJSS) ve Grid İş Dağıtım Yöneticisi (Grid Job Dispatch Manager - GJDM) tarafından bir siteye gönderilen işlerin kabul edildiği Site İş Arz Servisi (Site Job Submission Service - SJSS) bileşenlerinden oluşur. İş sunum servisleri, gelen her bir iş isteğini uygun olan global/yerel iş sırasına ekler ve gelen iş konusunda ilgili çizelgeleme servisini bilgilendirir. GRASP paketinde yer alan Job Submission Service [65] bir örnek olarak sayılabilir.

2.2.3.3. Çizelgeleme servisleri

GJSS tarafından bildirilen işleri Grid Çizelgeleme Servisi (Grid Scheduling Service - GSS) ve SJSS tarafından bildirilen işleri Site Çizelgeleme Servisi (Site Scheduling Service - SSS) kabul eder. Çizelgeleme servisleri yerel/global Grid servisleri tarafından sağlanan bilgileri de kullanarak, iş için gerekli kaynakları çizelgelemeye çalışırlar. Örnekleri GRASP paketindeki çizelgeleme servisi [65], K-Grid bünyesindeki RAEMS [66] ve EU DataGrid bünyesindeki İş yükü Yönetim Sistemi (Workload Management System - WMS) [67] olarak sayılabilir. Çizelgeleme servisleri üç aşamalı olarak çalışırlar: 1) Kaynakları bulma: tüm sistem üzerindeki uygun olan kaynakların bulunması; 2) Seçme: Uygun kaynaklar arasından, işlem, veri depolama, ve ağ kaynaklarını verimli kullanacak ve ön koşulları da sağlayacak biçimde bir listenin oluşturulması; 3) İşlerin yürütülmesi: Veri transferlerinin ve uygulamaların çalıştırılması.

2.2.3.4. İş dağıtım yöneticisi

Grid İş Dağıtım Yöneticisi (Grid Job Dispatch Manager - GJDM), GSS tarafından çizelgelenen işleri SJSS birimlerine yönlendiren servistir.

2.2.3.5. İş başlatma servisi

İş Başlatma Servisi (Job Invoke Manager - JIM) site içinde çalışacak olan ve başlama saati gelmiş olan işleri ilgili işlem elemanına gönderen bir servistir.

2.2.3.6. Kopya kataloğu

Kopya Kataloğu (Replica Catalogue - RC), tüm Veri Grid sistemlerinin temel taşlarındandır. RC, Veri Grid sisteminde, sitelerde bulunan Yerel Kopya Kataloğu (Local Replica Catalogue - LRC) bileşenleriyle irtibatlı çalışır. LRC'ler sitedeki veri depolama elemanlarında bulunan veriler için Mantıksal Veri İsmi (Logical File Name - LFN) ⇔ Fiziksel Veri İsmi (Physical File Name - PFN) eşlemelerini tutan veritabanlarıdır. LRC birimleri, ayrıca Kopya Konum Göstergesi (Replica Location Index – RLI) veritabanlarını periyodik olarak güncellerler. EU DataGrid [9] ve Globus [3] sistemleri, kopya kataloğu servislerini kullanırlar.

2.2.3.7. Kopya konum göstergesi

Kopya Konum Göstergesi (Replica Location Index – RLI), tüm Grid sistemi üzerinde bulunan veriler için LFN ⇔ LRC eşlemelerini tutan bir veritabanıdır. RLI, Grid sistemindeki tüm LFN'leri barındıran merkezi bir servis olabileceği gibi LFN kümelerini barındıran bir dağıtık servisler kümesi de olabilir.

2.2.3.8. Kopya konum servisleri

Kopya Konum Servisi (Replica Location Service - RLS), Yerel Kopya Kataloğu (Local Replica Catalogue - LRC) ve Kopya Konum Göstergesi (Replica Location Index – RLI) servisleri ile haberleşerek, istenen verilerin Grid sistemi içindeki konumlarını bulan servistir. Yerel Kopya Konum Servisi (Local Replica Location Service - LRLS) ise istenen verilerin sitede olup/olmadığını sorgulayan bir servistir. Globus [3] bünyesindeki Replica Location Service ve Giga-Ölçekli

Küresel Konumlama Motoru (Giga-scale Global Location Engine - Giggle) [68] örnek olarak sayılabilir.

2.2.3.9. Rezervasyon servisleri

Grid sistemi üzerindeki kaynakların önceden rezervasyonundan sorumlu olan Rezervasyon Servisi (Reservation Service - RS) ve site üzerindeki kaynakların önceden rezervasyonlarından sorumlu olan Yerel Rezervasyon Servisi (Local Reservation Service - LRS) bileşenlerinden oluşur. RS, gelen rezervasyon isteklerini site içindeki LRS'lere iletebilir veya LRS'lerden uygun kaynakları sorgulayabilir. Rezervasyon isteklerinin alınması, değiştirilmesi, karşılanması gibi fonksiyonları yerine getirir. Ayrıca, rezerve edilmiş olan kaynakların rezervasyon durumlarını da kontrol eder. PluS [69] örnek olarak sayılabilir.

2.2.3.10. Veri transfer servisi

Veri Transfer Servisi (Data Transfer Service - DTS) ilgili veri transfer protokolünü kullanarak verilerin kaynak sitelerden hedef sitelere transfer edilmesini sağlayan bir merkezi servistir. Örnek olarak Globus bünyesindeki GridFTP [27] ve Kangaroo [70] bünyesindeki Nexus [71] sayılabilir.

2.2.3.11. Veri yönetim servisleri

Veri Yönetim Servisi (Data Management Service - DMS) Grid sisteminde bir siteden diğer bir siteye veri transferleri isteklerini yöneten ve çizelgeleyen; Yerel Veri Yönetimi Servisi (Local Data Management Service - LDMS) ise site içindeki veri transferi isteklerinin çizelgelenmesinden sorumlu servistir. DMS, gerçek-zamanlı veri transferi istekleri için verilerin verimli bir şekilde dağıtılmasını sağlayan algoritmalarının da gerçeklendiği yerdir. Ayrıca, verilerin ilgili kopya kataloglarına kayıt yaptırmalarını da kontrol eder. Reptor [72], örnek olarak sayılabilir.

2.2.3.12. Veri yöneticisi servisleri

Veri Grid sistemine gelen veri transferi isteklerinin kabul edildiği Veri Yöneticisi (Data Manager - DM) ve site içi veri transferi isteklerinin kabul edildiği Yerel Veri Yöneticisi (Local Data Manager - LDM) bileşenlerinden oluşur. Veri yöneticisi servisleri, gelen veri transferi isteklerini uygun veri isteği sırasına ekler ve gelen istek konusunda DMS/LDMS'yi bilgilendirir.

2.2.3.13. Veri kopyalama servisleri

Global istatistikleri tutan Veri Kopyalama Servisi (Data Replication Service - DRS) ve site içi verilerin istatistikleri tutan Yerel Veri Kopyalama Servisi (Local Data Replication Service - LDRS) bileşenlerinden oluşur. Kopyalama servisleri, tutulan bu istatistikleri kullanarak, çekme veya itme tabanlı veri kopyalama algoritmalarını çalıştırır ve algoritmalar tarafından oluşturulan veri transferi isteklerini ilgili verinin siteden siteye transferi için Veri Yöneticisi (Data Manager - DM) servisine iletirler. Örnek olarak GDMP [73] gösterilebilir.

2.2.4. Uygulama

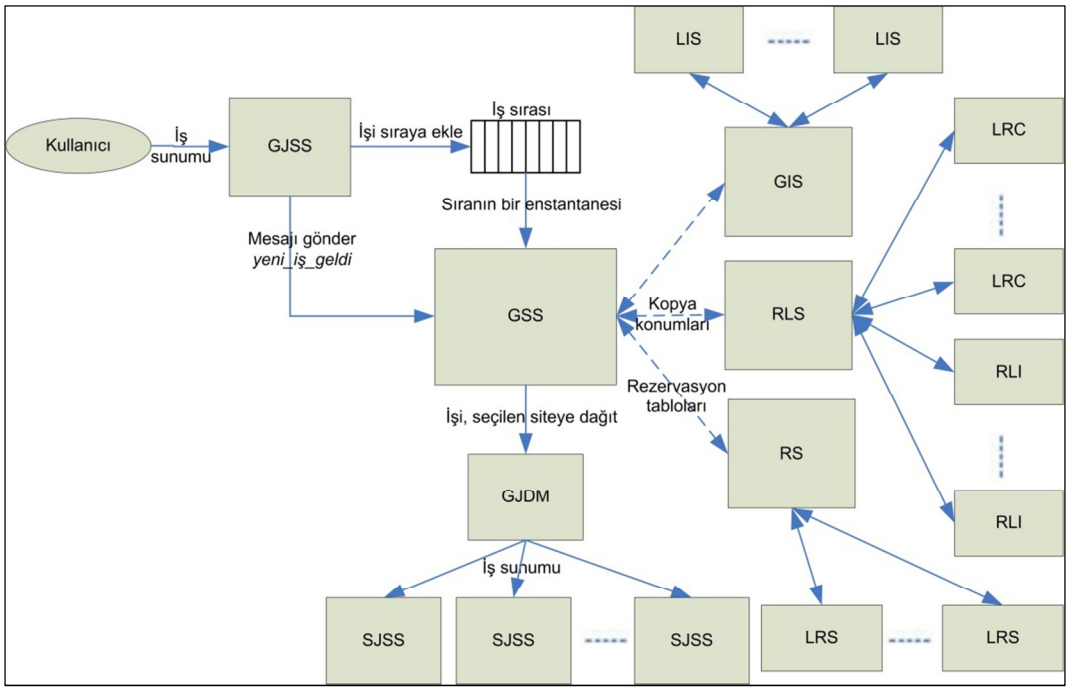
Veri Grid sistemleri, [74]'te anlatıldığı üzere, süreç-tabanlı (process oriented), bağımsız işler (independent tasks), iş torbası (bag of tasks) ve iş akışları (workflows) iş modellerini çalıştırırlar. Bu çalışmada, bağımsız işler uygulama modeli varsayılmıştır. Bu modele göre, farklı amaçları gerçekleştiren işler ve bu işler için gerekli olan sistem kaynakları, bağımsız olarak çizelgelenirler. Bu çalışmada belirtilen modele, diğer uygulama modellerinin uyarlanması, ilerisi için olası bir araştırma konusudur.

2.3. Hiyerarşik İş Çizelgeleme

Bu çalışmada önerilen model, iki seviyeli bir hiyerarşi içerir: GSS global çizelgeleme kararları verirken, sitelerde bulunan SSS ise yerel çizelgeleme kararlarını verir.

2.3.1. Grid Çizelgelemesi

Şekil 2.3'te işlerin çizelgelenmesi için, GSS'in diğer Grid servisleri ile girdiği etkileşimler gösterilmiştir. Şekilde gösterilen kesikli çizgi, iş çizelgeleme algoritmalarına bağlı olarak oluşabilecek servis etkileşimini gösterir. GSS, *en yakın bitiş zamanı öncelikli site*, *veriye sahip en az bitiş zamanı öncelikli site*, *veriye sahip min-min site* gibi farklı çizelgeleme algoritmalarını çalıştırabilir. Adı verilen algoritmalar, önerilen iş çizelgeleme modelinin bünyesine farklı çevrim içi ve çevrim dışı iş çizelgeleme algoritmalarının dâhil edilmesine örnek teşkil etmesi için anlatılacaktır. Literatürdeki, farklı iş çizelgeleme algoritmaları da, benzer biçimde modele dâhil edilebilir.



Şekil 2.3. Hiyerarşik iş çizelgeleme modelinde Grid sistemine iş arzı

1. Şekil 2.3'te gösterildiği üzere GJSS kullanıcıdan gerçek-zamanlı işleri alır. Kullanıcılar Grid sistemine işleri, çeşitli uygun arayüzler aracılığı ile arz ederler. İşler arz edilirken, gerekli ön koşullar ve kabullenmeler de, İş Tanımlama Dili (Job Description Language - JDL) [75] veya benzeri bir tanımlama dili ile belirtilirler.

2. GJSS, gelen işi sıraya ekler ve GSS'yi haberdar eder. Benzeri iş sıraları, GRASP [63] veya Globus [3] sistemlerinde de kullanılmıştır.
3. GSS, çevrim içi veya çevrim dışı bir çizelgeleme algoritması çalıştırarak işin hangi siteye gönderileceğine karar verir.
    a. *Çevrim içi iş çizelgeleme algoritması*: GSS gelen işleri hemen çizelgelemeye çalışır.
    b. *Çevrim dışı iş çizelgeleme algoritması*: GSS, belli aralıklar ile global iş sırasını kontrol eder, çizelgelenmemiş işleri alır ve çizelgeler.

    Grid çizelgeleme algoritması işleri çizelgelerken, rezervasyon tabloları için RS'yi veya iş için gerekli veriler için RLS'yi veya diğer bazı bilgiler için de GIS'yi sorgulayabilir. Bu servisler, sorgulanan bilgiler için yerel servisler ile haberleşebilirler.
4. İşin gönderileceği site seçildikten sonra, GSS işi GJDM'ye gönderir ve işi global sıradan çıkartır. İşin ilgili SJSS'ye gönderilmesi, GJDM'nin sorumluluğundadır.

Bu çalışma kapsamında geliştirilen örnek bazı iş çizelgeleme algoritmaları aşağıda listelenmiştir.

2.3.1.1. Rassal

Rassal (Random - Rand) algoritması GSS tarafından çevrim içi olarak şu şekilde gerçeklenir: İş çizelgeleme kuyruğuna giren her bir iş için, beklemeksizin bir site rassal olarak seçilir.

2.3.1.2. En erken son zaman ilk

En erken son zaman ilk (Earliest Deadline First – EDF) algoritması GSS tarafından çevrim dışı olarak aşağıdaki şekilde çalıştırılır:

1. İş çizelgeleme olayı periyodik olarak oluştuğunda, kuyrukta bekleyen işler son zamanlarına göre küçükten büyüğe dizilir.
2. Bu sıraya göre, her bir iş için bir site rastgele olarak seçilir.

2.3.1.3. En erken bitiş zamanı ilk

En erken bitiş zamanı ilk (Minimum Completion Time First – MCTF) algoritması, GSS tarafından çevrim içi olarak aşağıdaki şekilde çalıştırılır:

1. GSS, sıradaki işleri birer birer çizelgeler. Hâlihazırdaki iş *j* için GSS, GIS'e, tüm site *i*'ler için ayrı ayrı MIPS (Millions of Instructions Per Second) cinsinden toplam işlem gücü (p_i) ve saniye cinsinden ortalama anlık iş yükü (l_i) bilgileri için istekte bulunur.
2. GSS, birinci aşamada, GIS'den bilgisi dönülen tüm siteler için beklenen iş bitiş zamanı (c_i) hesaplar. İşin, ilgili siteye gönderilmesi durumunda, beklenen iş bitiş zamanı, $c_i = l_i + t_j/p_i$ eşitliği ile bulunur. Bu eşitlikte t_j, işin MI (Millions of Instructions) cinsinden toplam büyüklüğünü ifade eder.
3. GSS, en az beklenen iş bitiş zamanını sağlayan siteyi seçer ve GJDM'yi haberdar eder.
4. GJDM işi seçilen siteye gönderir.

2.3.1.4. Veriyi barındıran en erken bitiş zamanı ilk

Veriyi barındıran en erken bitiş zamanı ilk (Minimum Completion Time First with Data Present – MCTFwDP) algoritması, GSS tarafından çevrim içi olarak aşağıdaki belirtildiği şekilde gerçeklenir:

1. GSS, sıradaki işleri birer birer çizelgeler. Hâlihazırdaki iş *j* için GSS, MCTF algoritmasındakine benzer olarak GIS'den tüm site *i*'ler için p_i ve l_i değerlerini sorgular.
2. GSS, RLS'den, iş için gerekli olan verilerin bulunduğu siteleri sorgular.
3. İşin, ilgili siteye gönderilmesi durumunda, beklenen iş bitiş zamanı, c_i, aşağıdaki şekilde bulunur:
   a. Eğer iş *j* için gerekli en az bir veri site *i*'de mevcut ise $c_i = l_i + t_j/p_i$; eğer mevcut değil ise c_i = *sonsuz* olarak bulunur.

b. Eğer tüm siteler için c_i = *sonsuz* ise; tüm site *i*'ler için $c_i = l_i + t_j/p_i$ olarak bulunur.

4. GSS, en az beklenen iş bitiş zamanını (c_i) sağlayan siteyi seçer ve GJDM'yi haberdar eder.
5. GJDM işi seçilen siteye gönderir.

2.3.1.5. Veriyi barındıran MinMin

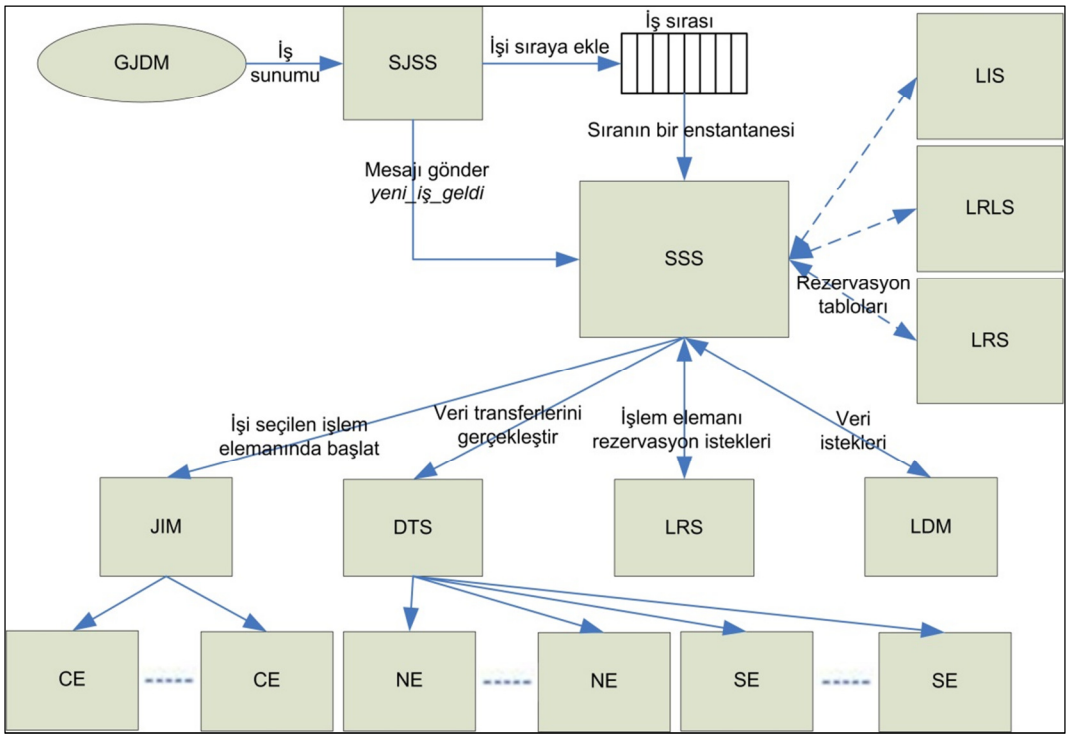
Veriyi barındıran MinMin (MinMin with Data Present – MMwDP) algoritması, GSS tarafından çevrim dışı bir algoritma olarak gerçeklenir ve aşağıdaki şekilde çalıştırılır:

1. GSS, sıradaki tüm iş *j*'leri aynı anda çizelgeler. GSS, GIS'den tüm site *i*'ler için p_i ve l_i değerlerini sorgular.
2. GSS, RLS'den, sıradaki her bir iş için gerekli olan verilerin bulunduğu siteleri sorgular.
3. GSS, beklenen iş bitiş zamanları matrisi olan *C*'yi hesaplar. *C* matrisinin elemanları olan c_{ij}, iş *j*'in site *i* üzerinde beklenen iş bitiş zamanını ifade eder ve aşağıdaki şekilde hesaplanır:
   a. İş *j* için gerekli verilerden bir veya birden fazla sayıda barındıran her bir site *i* için $c_{ij} = l_i + t_j/p_i$ şeklinde hesaplanır. Diğer site *i*'ler için c_{ij} = *sonsuz* atanır.
   b. Eğer tüm siteler için c_{ij} = *sonsuz* ise; tüm site *i*'ler için $c_{ij} = l_i + t_j/p_i$ olarak hesaplanır.
4. GSS, işi ve atanacağı siteyi, min-min stratejisinde [76] belirtildiği şekilde seçer ve sıradaki işler bitinceye kadar *C* matrisini günceller.
5. GDJM işleri, seçilen sitelere gönderir.

2.3.2. Site Çizelgelemesi

Şekil 2.4'te, GJDM tarafından siteye gönderilen işlerin, site içindeki işlem elemanlarına çizelgelenmesi için, SSS'in diğer sistem servisleri ile olan

etkileşimleri gösterilmiştir. Literatürdeki iş çizelgeleme algoritmaları, önerilen site çizelgeleme modeline uymaları koşulu ile, sisteme entegre edilebilirler.



Şekil 2.4. Hiyerarşik iş çizelgeleme modelinde Grid sitelerine iş arzı

1. SJSS, GJDM'den işi alır.
2. SJSS, işi yerel iş sırasına ekler ve SSS'yi haberdar eder.
3. SSS, çevrim içi veya dışı iş çizelgeleme algoritmasını çalıştırarak işin son zamanından önce çalışacağı işlem eleman(lar)ı ve zaman aralık(lar)ı bulmaya çalışır.
   a. *Çevrim içi iş çizelgeleme algoritması*: SSS gelen işleri hemen çizelgelemeye çalışır.
   b. *Çevrim dışı iş çizelgeleme algoritması*: SSS, belli aralıklar ile global iş sırasını kontrol eder, çizelgelenmemiş işleri alır ve çizelgeler.

   SSS, çizelgeleme algoritmasına göre LRS, LRLS ve LIS servislerini sorgulayabilir; [77] ve benzeri çalışmalardaki çizelgeleme algoritmalarından herhangi birisini gerçekleyebilir. Çizelgeleme

algoritması, her bir iş için [*SeçilenİşlemElemanı*, *İşBaşlangıçZamanı*, *İşBitişZamanı*] bilgilerini hesaplar. Örnek bir site iş çizelgeleme algoritması olarak RT MaxMax, Bölüm 2.3.2.1'de sunulmuştur.

4. Eğer *SeçilenİşlemElemanı* = *Boş* ise; SSS işi yerel iş sırasından siler. İş *karşılanamaz* olarak işaretlenir ve işlerin çizelgelenmesi prosedürü sonlandırılır.
5. Eğer site iş çizelgeleme algoritması tarafından *SeçilenİşlemElemanı* ≠ *Boş* olarak belirtilirse, SSS, iş için seçilen işlem elemanı üzerindeki işlem bant genişliği rezervasyonu için LRS'ye *REZERVASYON_OLUŞTUR* := [*Kaynak* = *SeçilenİşlemElemanı*, *RezervasyonBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBitişZamanı* = *İşBitişZamanı*, *RezervasyonBüyüklüğü* = *İşBüyüklüğü*] mesajı gönderir. SSS, LRS'den *REZERVASYON_ONAY* := [*RezervasyonBilgisi* = *İşlemElemanıRezervasyonBilgisi*] veya *REZERVASYON_HATA* mesajı bekler. *İşlemElemanıRezervasyonBilgisi*, işin son zamanından önce tamamlanması için *SeçilenİşlemElemanı* üzerinde rezerve edilmiş olan rezervasyon objesinin bilgilerini içerir.
6. LRS'den *REZERVASYON_HATA* mesajı gelirse, iş, yerel iş sırasından silinir. İş *karşılanamaz* olarak işaretlenir ve işlerin çizelgelenmesi prosedürü sonlandırılır.
7. Eğer LRS'den *REZERVASYON_ONAY* := [*RezervasyonBilgisi* = *İşlemElemanıRezervasyonBilgisi*] mesajı gelirse, iş ile ilişkili tüm verileri *İşBaşlangıçZamanı*'dan önce transfer etmesi için, SSS, LDM'ye iş ile ilgili tüm verileri içeren *VERİ_TRANSFERİ_İSTEĞİ* := [*TransferEdilecekVeriListesi* = *İşVeriListesi*, *Hedef* = *SeçilenİşlemElemanı*, *TransferSonZamanı* = *İşBaşlangıçZamanı*] mesajı gönderir. LDM, hiyerarşik veri dağıtımı bölümünde daha detaylı anlatılacağı şekilde, iş ile ilişkili verileri *İşBaşlamaZamanı*'ndan önce site içinden veya dışından sağlayabilir. SSS, LDM'den *VERİ_TRANSFERİ_ONAY* := [*OnayID* = *VeriTransferiOnayID*, *RezervasyonBilgisi* = *VeriTransferiİstekRezervasyonBilgisi*, *KilitBilgisi*

= *VeriTransferiİstekKilitBilgisi*] veya *VERİ_TRANSFERİ_HATA* mesajı bekler. *VeriTransferiİstekRezervasyonBilgisi*, iş ile ilgili verilerin transfer edilebilmesi için gerekli olan tüm ağ elemanı rezervasyonları için rezervasyon objeleri bilgilerini; *VeriTransferiİstekKilitBilgisi* ise, iş ile ilgili verilerin transfer edilmesi sırasında gerekli olan veri kilitleri ile ilgili bilgileri içerir.

8. Eğer LDM'den *VERİ_TRANSFERİ_İSTEĞİ_ONAY* := [*OnayID* = *VeriTransferiOnayID*, *RezervasyonBilgisi* = *VeriTransferiİstekRezervasyonBilgisi*, *KilitBilgisi* = *VeriTransferiİstekKilitBilgisi*] mesajı gelirse:
   a. SSS, işin *SeçilenİşlemElemanı* üzerinde çalıştırılması için gerekli olan rezervasyon bilgilerini içeren *İşlemElemanıRezervasyonBilgisi*'ni JIM'e gönderir. JIM, işin, LRS tarafından rezerve edilmiş olan işlem elemanı *SeçilenİşlemElemanı* üzerinde [*İşBaşlangıçZamanı*, *İşBitişZamanı*] aralığında çalışmasını sağlar.
   b. SSS, iş ile ilgili veri transferlerini yapmak için DTS'ye, LDM'den gelen *VeriTransferiİstekRezervasyonBilgisi*'ni ve *VeriTransferiİstekKilitBilgisi*'ni iletir. DTS, *VeriTransferiİstekRezervasyonBilgisi*'nde belirtilen ağ kaynaklarının ve *VeriTransferiİstekKilitBilgisi*'nde belirtilen veri depolama kaynaklarının kullanılmasını kontrol eder.
   c. SSS'nin JIM ve DTS'ye gerekli mesajları iletmesinden sonra, SSS, işi yerel iş sırasından siler. Bu durumda iş *başarılı* olarak işaretlenir ve işlerin çizelgelenmesi prosedürü tamamlanmış olur.
9. Eğer LDM'den *VERİ_TRANSFERİ_İSTEĞİ_HATA* mesajı gelirse, iş, yerel iş sırasından silinir. İş için yapılmış olan işlem elemanı rezervasyonlarının iptal edilebilmesi için SSS, LRS'ye *REZERVASYON_SİL*:= [*RezervasyonBilgisi* = *İşlemElemanıRezervasyonBilgisi*] mesajı gönderir.

2.3.2.1. Gerçek-zamanlı MaxMax

Gerçek-zamanlı MaxMax (Real-Time MaxMax – RT MaxMax) iş çizelgeleme algoritması, SSS tarafından çevrim içi olarak aşağıdaki belirtildiği şekilde gerçeklenir:

1. SSS, sıradaki işleri birer birer çizelgeler. Hâlihazırdaki iş için SSS, LRS'ye, *REZERVASYON_SORGULA* := [*RezervasyonBüyüklüğü* = *İşBüyüklüğü*, *Tip* = *İşlem, KaynakListesi* = *SiteİçindekiTümİşlemElemanları*, *RezervasyonSonZamanı* = *İşSonZamanı*] biçiminde bir mesaj gönderir.
2. LRS, rezervasyon tablosundaki işlem elemanlarıyla ilgili rezervasyon satırları üzerinde, işi son zamanından önce tamamlayabilecek uygun pencereler için arama yapar. Arama sonuçlarını *REZERVASYON_SORGU_CEVAP* := [*Kaynak* = *İşlemElemanı*, *RezervasyonPencereBaşlangıçZamanı* = *PencereBaşlangıçZamanı*, *RezervasyonPencereBitişZamanı* = *PencereBitişZamanı*, *RezervasyonPencereBüyüklüğü* = *PencereBüyüklüğü*] şeklinde rezervasyon pencerelerinden oluşan bir liste biçiminde SSS'ye iletir.
3. Eğer SSS'ye iletilen rezervasyon sorgu cevabı listesinde, en az bir rezervasyon penceresi var ise; SSS, rezervasyon pencereleri arasından, en geç *PencereBaşlangıçZamanı* değerine sahip olan pencereyi bulur. Bu rezervasyon penceresini sağlayan *İşlemElemanı*, *SeçilenİşlemElemanı* olur. Bu rezervasyon penceresinin *PencereBitişZamanı* değeri *İşBitişZamanı* olarak atanır. *İşBaşlangıçZamanı* = *İşBitişZamanı* - *İşBüyüklüğü*/*PencereBüyüklüğü* olarak hesaplanır. Eğer SSS'ye iletilen rezervasyon sorgu cevabı listesinde, en az bir rezervasyon penceresi yok ise *SeçilenİşlemElemanı* = *Boş* olarak atanır.

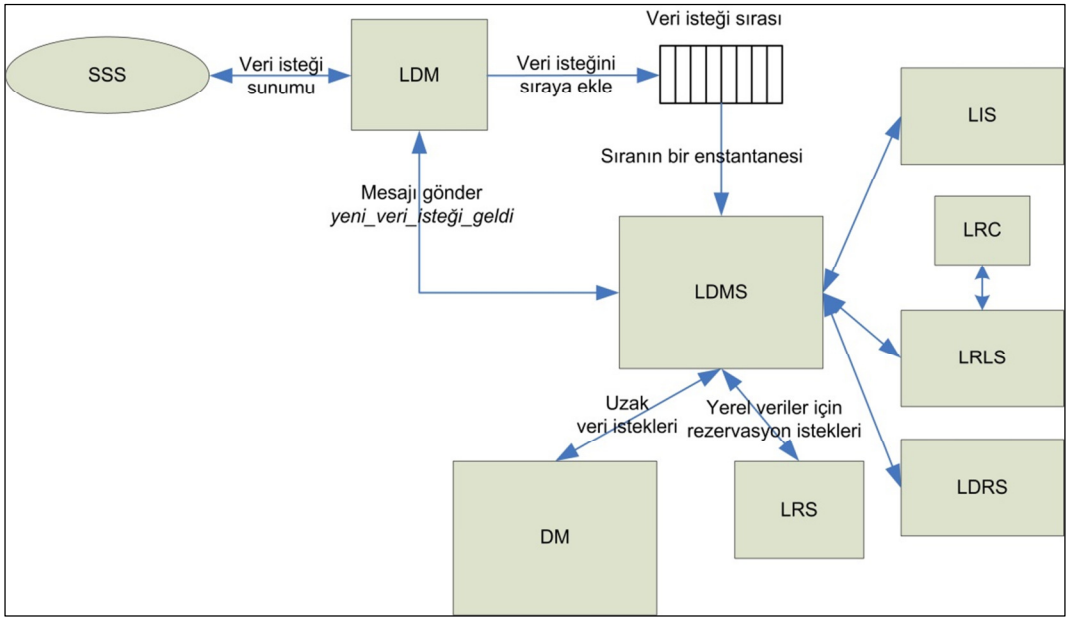
2.4. Hiyerarşik Veri Dağıtımı

İş çizelgelemeye benzer olarak, veri çizelgeleme de hiyerarşik bir düzende tasarlanmıştır. Herbir sitede bir adet Yerel Veri Yönetim Servisi (Local Data

Management Service - LDMS) ile tüm Veri Grid sistemi için bir Veri Yönetim Servisi (Data Management Service - DMS) bulunmaktadır. Bu servislerin çalışmaları aşağıda anlatılmıştır.

2.4.1. Veri Transferlerinin Site İçi Koordinasyonu

SSS, işleri çizelgelerken, LDMS'de işin ihtiyaç duyduğu verilerin depolama elemanından transfer edilmesinde görevlidir. Şekil 2.5'te iletilen veri transferi isteklerinin karşılanması sırasında LDMS ve diğer sistem servislerinin etkileşimleri gösterilmiştir.



Şekil 2.5. SSS tarafından gönderilen veri transferleri isteklerinin karşılanışı

1. Şekil 2.5'te görüldüğü üzere, LDM, SSS'den, iş ile ilgili tüm verileri içeren *VERİ_TRANSFERİ_İSTEĞİ* := [*TransferEdilecekVeriListesi* = *İşVeriListesi*, *Hedef* = *SeçilenİşlemElemanı*, *TransferSonZamanı* = *İşBaşlamaZamanı*] mesajını alır.
2. LDM, bu isteği veri istek sırasına koyar ve LDMS'yi haberdar eder.
3. LDMS çevrim içi veya çevrim dışı çalışarak, veri isteklerini alır. LDMS, *İşVeriListesi*'nde yer alan tüm LFN'ler için, LRLS'de

sorgulama yapar. LDMS, LRLS'den gelen cevaba göre iki liste oluşturur: *İşVeriListesi*'nde ve sitede yerel olarak bulunan verileri içeren ve PFN'lerden oluşan *İşYerelVeriListesi* ve *İşVeriListesi*'nde bulunan ancak sitedeki depolama elemanlarında bulunmayan verileri içeren ve LFN'lerden oluşan *İşUzakVeriListesi*.

4. LDMS, *İşYerelVeriListesi*'ndeki her PFN için LRS'ye üç farklı mesaj gönderir: 1) verinin veri depolama elemanından okunması ile ilgili olan *REZERVASYON_OLUŞTUR* := [*Kaynak* = *KaynakDepolamaElemanı*:*Okuma*, *RezervasyonBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBitişZamanı* = *İşBitişZamanı*, *RezervasyonBüyüklüğü* = *YerelRotaMinBantGenişliği*] mesajı; 2) *KaynakDepolamaElemanı* ile *SeçilenİşlemElemanı* arasında bulunan rota üzerindeki her link için *REZERVASYON_OLUŞTUR* := [*Kaynak* = *RotaÜzerindekiLink*, *RezervasyonBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBitişZamanı* = *İşBitişZamanı*, *RezervasyonBüyüklüğü* = *YerelRotaMinBantGenişliği*] mesajı; 3) verinin, *SeçilenİşlemElemanı*'na yazılması ile ilgili olan *REZERVASYON_OLUŞTUR* := [*Kaynak* = *SeçilenİşlemDepolamaElemanı*:*Yazma*, *RezervasyonBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBitişZamanı* = *İşBitişZamanı*, *RezervasyonBüyüklüğü* = *YerelRotaMinBantGenişliği*] mesajı. LDMS, LRS'den *REZERVASYON_ONAY* := [*RezervasyonBilgisi* = **RezervasyonBilgisi*] veya *REZERVASYON_HATA* mesajlarını bekler. Burada, **RezervasyonBilgisi*; *KaynakDepolamaElemanı*'ndaki okuma bant genişliği, *RotaÜzerindekiLink*'ler üzerindeki ağ bant genişlikleri ve *SeçilenİşlemDepolamaElemanı*'ndaki yazma bant genişliği rezervasyonlarından ilgili olanının bilgisini ifade eder. Ayrıca, *YerelRotaMinBantGenişliği* = *VeriBüyüklüğü*/(*İşBitişZamanı*-*İşBaşlangıçZamanı*-*YerelToplamLinkGecikmesi*) olarak hesaplanır. Formüldeki *YerelToplamLinkGecikmesi* değeri, site *KaynakDepolamaElemanı*'ndan aynı sitede bulunan

SeçilenİşlemElemanı'na olan rota üzerindeki linklerin toplam gecikmesini ifade eder.

   a. Eğer *İşYerelVeriListesi*'ndeki herhangi bir PFN için LRS'den *REZERVASYON_HATA* mesajı gelirse, iş ile ilgili veri transferleri karşılanamamıştır. Bu durumda, LDMS LDM'ye, LDM'de SSS'ye *VERİ_TRANSFERİ_HATA* mesajı iletir. Yerel veri tranferi istekleri ile ilgili yapılmış olan rezervasyonları silmek için de, LDMS, LRS'ye *REZERVASYON_SİL* := [*RezervasyonBilgisi* = **RezervasyonBilgisi*] mesajları gönderir. Veri transferi prosedürü sonlandırılır.
   b. Eğer *İşYerelVeriListesi*'ndeki her PFN için LRS'den *REZERVASYON_ONAY* := [*RezervasyonBilgisi* = **RezervasyonBilgisi*] mesajı gelirse, yerel veri transferi ile ilgili tüm rezervasyon objelerindeki bilgiler, *YVİ_RezervasyonBilgisi* veri yapısında depolanır ve veri transferi prosedürüne devam edilir.

5. LDMS, *İşYerelVeriListesi*'ndeki her PFN için *KaynakDepolamaElemanı* yöneticisine *VERİ_KİLİTLE* := [*Veri* = *LFN*, *KilitBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *KilitBitişZamanı* = *İşBitişZamanı*] biçiminde bir mesaj gönderir. LDMS, *KaynakDepolamaElemanı* yöneticisinden *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = *YerelVeriKilitBilgisi*] veya *VERİ_KİLİT_HATA* mesajlarını bekler.
   a. Eğer *İşYerelVeriListesi*'ndeki herhangi bir PFN için *KaynakDepolamaElemanı* yöneticisinden *VERİ_KİLİT_HATA* mesajı gelirse, iş ile ilgili veri transferleri karşılanamamıştır. Bu durumda, LDMS LDM'ye, LDM'de SSS'ye *VERİ_TRANSFERİ_HATA* mesajı iletir. Ayrıca LDMS, yerel veri tranferi istekleri ile ilgili yapılmış olan rezervasyonları silmek için LRS'ye *REZERVASYON_SİL* := [*RezervasyonBilgisi* = **RezervasyonBilgisi*] mesajları ve yapılmış olan veri kilitlerinin açılması için *KaynakDepolamaElemanı* yöneticisine

VERİ_KİLİT_SİL := [*VeriKilitBilgisi* = *YerelVeriKilitBilgisi*] mesajlarını gönderir. Veri transferi prosedürü sonlandırılır.

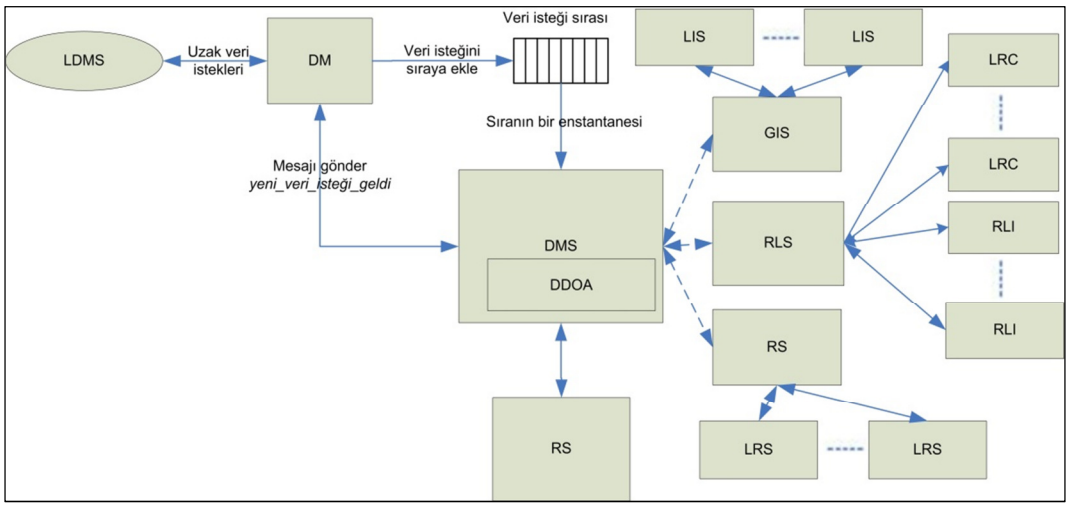
b. Eğer *İşYerelVeriListesi*'ndeki her PFN için *KaynakDepolamaElemanı* yöneticisinden *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = *YerelVeriKilitBilgisi*] mesajı gelirse, yerel veri transferi ile ilgili tüm veri kilit bilgileri, *YVİ_VeriKilitBilgisi* veri yapısında depolanır ve veri transferi prosedürüne devam edilir

6. Veri kopyalama algoritmaları çalıştırılarak *VeriKopyalamaListesi* bilgisi alınır. Veri Grid sisteminde kullanılan veri kopyalama algoritmaları üç farklı modelden birisine uygun olabilir: çekme tabanlı dağıtık, itme tabanlı dağıtık ve itme tabanlı merkezi.
   a. *Çekme tabanlı dağıtık*: LDMS, çekme tabanlı dağıtık veri kopyalama için LDRS'ye *VERİ_İSTEĞİ_GELDİ* := [*TransferEdilecekVeriListesi* = *İşVeriListesi*, *İstekSonZamanı* = *TransferSonZamanı*] mesajı gönderir. LDRS, LDMS'ye *VERİ_KOPYALAMA_CEVAP* := [*KopyalanacakVeriListesi* = *VeriKopyalamaListesi*] biçiminde bir mesajla cevap verir.
   b. *İtme tabanlı dağıtık*/*itme tabanlı merkezi*: LDMS, itme tabanlı dağıtık ve itme tabanlı merkezi veri kopyalama için LDRS'ye *VERİ_İSTEĞİ_GELDİ* := [*TransferEdilecekVeriListesi* = *İşVeriListesi*, *İstekSonZamanı* = *TransferSonZamanı*] mesajı gönderir. LDRS, LDMS'ye *VERİ_KOPYALAMA_CEVAP* := [*KopyalanacakVeriListesi* = *Boş*] biçiminde bir mesajla cevap verir.
7. LDMS, DM'ye, *UZAK_VERİ_TRANSFERİ_İSTEĞİ* := [*TransferEdilecekVeriListesi* = *İşUzakVeriListesi*, *Hedef* = *SeçilenİşlemElemanı*, *TransferSonZamanı* = *İşBaşlamaZamanı*, *KopyalanacakVeriListesi* = *VeriKopyalamaListesi*] biçiminde bir mesaj gönderir. LDMS, DM'den *UZAK_VERİ_TRANSFERİ_ONAY* := [*OnayID* = *UzakVeriTransferiOnayID*, *RezervasyonBilgisi* = *UVİ_RezervasyonBilgisi*, *KilitBilgisi* = *UVİ_KilitBilgisi*] veya *UZAK_VERİ_ TRANSFERİ_HATA* mesajlarını bekler. DM'den gelen

UVİ_RezervasyonBilgisi veri yapısında, uzak veri transferi isteğinin karşılanması için gerekli olan ağ elemanı rezervasyonu objelerinde yer alan bilgiler bulunur. *UVİ_KilitBilgisi* veri yapısında ise, uzak veri transferi isteğinin karşılanması için gerekli olan veri kilit bilgileri bulunur.

a. Eğer LDMS, *UZAK_VERİ_ TRANSFERİ_HATA* mesajı alır ise, istek ile ilgili yapılmış olan tüm yerel rezervasyonları iptal etmek için, LRS'ye, kaydedilmiş olan tüm rezervasyonlar için birer tane olmak üzere *REZERVASYON_SİL* := [*RezervasyonBilgisi* = **RezervasyonBilgisi*] mesajı; *KaynakDepolamaYönetici*'sine iş ile ilgili yerel veriler için *VERİ_KİLİT_SİL* := [*VeriKilitBilgisi* = *YerelVeriKilitBilgisi*] mesajı gönderir. LDMS iş ile ilgili veri transferi isteklerini, istek sırasından siler. LDMS LDM'ye, LDM'de SSS'ye *VERİ_TRANSFERİ_HATA* mesajı gönderir.
b. Uzak veri erişim istekleri için de DM'den *UZAK_VERİ_ TRANSFERİ_ONAY* := [*OnayID* = *UzakVeriTransferiOnayID*, *RezervasyonBilgisi* = *UVİ_RezervasyonBilgisi*, *KilitBilgisi* = *UVİ_KilitBilgisi*] mesajı alan LDMS, *YVİ_RezervasyonBilgisi* ve *UVİ_RezervasyonBilgisi* veri yapılarını birleştirerek *VeriTransferiİstekRezervasyonBilgisi* veri yapısını oluşturur. Ayrıca, *YVİ_KilitBilgisi* ve *UVİ_KilitBilgisi* veri yapılarını da birleştirerek *VeriTransferiİstekKilitBilgisi* veri yapısını oluşturur. LDMS LDM'ye, LDM'de SSS'ye *VERİ_TRANSFERİ_İSTEĞİ_ONAY* := [*OnayID* = *VeriTransferiOnayID*, *RezervasyonBilgisi* = *VeriTransferiİstekRezervasyonBilgisi*, *KilitBilgisi* = *VeriTransferiİstekKilitBilgisi*] mesajlarını gönderir.

2.4.2. Veri Transferlerinin Siteler Arası Koordinasyonu



Şekil 2.6. LDMS'den gelen veri transferi istekleri

1. LDMS, Şekil 2.6'da gösterildiği üzere, DM'ye, *UZAK_VERİ_TRANSFERİ_İSTEĞİ* := [*TransferEdilecekVeriListesi* = *İşUzakVeriListesi*, *Hedef* = *SeçilenİşlemElemanı*, *TransferSonZamanı* = *İşBaşlamaZamanı*, *KopyalanacakVeriListesi* = *VeriKopyalamaListesi*] biçiminde uzak veri transferi isteği mesajı gönderir.
2. DM, isteği istek sırasına ilave eder ve DMS'yi haberdar eder.
3. DMS çevrim içi veya çevrim dışı çalışarak, uzak veri isteklerini alır. DMS, RLS'den istek mesajında gelen her bir LFN için muhtemel kaynakları ister.
4. RLS, RLI'yı LFN'yi içeren LRC'ler için sorgular. RLS, LFN'yi içeren LRC'leri sorgulayarak PFN'leri ve her PFN için *VeriBüyüklüğü*'nü alır. RLS muhtemel kaynak veri depolama elemanları, ilgili diğer bilgilerle birlikte DMS'ye iletir.
5. DMS, Veri Dağıtımı Optimizasyonu Algoritması (Data Dissemination Optimization Algorithm - DDOA)'nı çalıştırarak veri transferi isteğini sağlayacak şekilde her bir LFN için en iyi kaynak ve en iyi rotayı bulmaya çalışır. Şekil 2.6'daki kesikli çizgi, DDOA algoritmasına bağlı olarak oluşabilecek servis etkileşimini gösterir. Algoritma,

KarşılanabilirİstekListesi'ni ve bu listedeki her bir veri transferi isteği için, *KaynakDepolamaElemanı*'ı, *Hedef*'i (*HedefDepolamaElemanı* veya *SeçilenİşlemElemanı*), *Rotalar*'ı ve *RotaÜzerindeKullanılanBantGenişliği*'ni geri döndürür. Örnek bir DDOA olarak SP_MinHop sunulmuştur.

6. Eğer DDOA'dan gelen *KarşılanabilirİstekListesi*'nde, iş ile ilgili en az bir LFN transferi yer almıyor ise veya *RotaÜzerindeKullanılanBantGenişliği* değeri, *UzakRotaMinBantGenişliği* değerinden küçük ise, DMS, veri isteğini istek sırasından siler ve DM'ye *UZAK_VERİ_İSTEĞİ_HATA* mesajı gönderir ve uzak veri transferi isteklerini karşılama prosedürü sona erer.
7. Eğer DDOA'dan gelen *KarşılanabilirİstekListesi*'nde, iş ile ilgili tüm veri transferleri yer alıyor ise; DMS, RS'ye rezervasyon oluşturma mesajları gönderir ve aşağıda belirtilen mesajları bekler.
   a. Eğer *LFN*, *VeriKopyalamaListesi*'nde var ise; DMS, RS'ye altı farklı mesaj gönderir: 1) *KaynakDepolamaElemanı*'ndaki verinin okuma bant genişliği ile ilgili olan *REZERVASYON_OLUŞTUR* := [*Kaynak* = *KaynakDepolamaElemanı*:*Okuma*, *RezervasyonBaşlangıçZamanı* = *ŞimdikiZaman, RezervasyonBitişZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBüyüklüğü* = *UzakRotaMinBantGenişliği*] mesajı; 2) *KaynakDepolamaElemanı* ile *HedefDepolamaElemanı* arasında bulunan ve *Rotalar* içindeki rota üzerinde yer alan her link için *REZERVASYON_OLUŞTUR* := [*Kaynak* = *RotaÜzerindekiLink*, *RezervasyonBaşlangıçZamanı* = *ŞimdikiZaman*, *RezervasyonBitişZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBüyüklüğü* = *UzakRotaMinBantGenişliği*] mesajı; 3) *HedefDepolamaElemanı*'nda verinin yazma bant genişliği ile ilgili *REZERVASYON_OLUŞTUR* := [*Kaynak* = *HedefDepolamaElemanı*:*Yazma*, *RezervasyonBaşlangıçZamanı* = *ŞimdikiZaman, RezervasyonBitişZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBüyüklüğü* = *UzakRotaMinBantGenişliği*] mesajı; 4)

verinin *HedefDepolamaElemanı*'ndan okunması ile ilgili olan *REZERVASYON_OLUŞTUR* := [*Kaynak* = *HedefDepolamaElemanı*:*Okuma*, *RezervasyonBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBitişZamanı* = *İşBitişZamanı*, *RezervasyonBüyüklüğü* = *YerelRotaMinBantGenişliği*] mesajı; 5) *HedefDepolamaElemanı* ile *SeçilenİşlemElemanı* arasında bulunan rota üzerindeki her link için *REZERVASYON_OLUŞTUR* := [*Kaynak* = *RotaÜzerindekiLink*, *RezervasyonBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBitişZamanı* = *İşBitişZamanı*, *RezervasyonBüyüklüğü* = *YerelRotaMinBantGenişliği*] mesajı; 6) verinin, *SeçilenİşlemElemanı*'na yazılması ile ilgili olan *REZERVASYON_OLUŞTUR* := [*Kaynak* = *SeçilenİşlemDepolamaElemanı*:*Yazma*, *RezervasyonBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBitişZamanı* = *İşBitişZamanı*, *RezervasyonBüyüklüğü* = *YerelRotaMinBantGenişliği*] mesajı. DMS, RS'den, her bir rezervasyon oluşturma isteğine karşılık *REZERVASYON_ONAY* := [*RezervasyonBilgisi* = *#RezervasyonBilgisi*] veya *REZERVASYON_HATA* mesajları bekler. Mesajdaki *#RezervasyonBilgisi*; yukarıda tanımlanan altı farklı rezervasyon mesajı ile ilgili oluşturulmuş olan rezervasyon objelerinde bulunan bilgilerdir. Burada, *UzakRotaMinBantGenişliği* = |*Veri*|/(*İşBaşlangıçZamanı*-*ŞimdikiZaman*-*UzakToplamLinkGecikmesi*) olarak hesaplanır. Formüldeki *UzakToplamLinkGecikmesi* değeri, farklı sitelerde bulunan *KaynakDepolamaElemanı*'ndan *HedefDepolamaElemanı*'na olan rota üzerindeki linklerin toplamgecikmesini ifade eder.

b. Eğer *LFN*, *KopyalanacakVeriListesi*'nde yok ise; DMS, RS'ye üç farklı mesaj gönderir: 1) *REZERVASYON_OLUŞTUR* := [*Kaynak* = *KaynakDepolamaElemanı*:*Okuma*, *RezervasyonBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *RezervasyonBitişZamanı* = *İşBitişZamanı*, *RezervasyonBüyüklüğü*

= *UzakRotaMinBantGenişliği*] mesajı; 2) *KaynakDepolamaElemanı* ile *SeçilenİşlemElemanı* arasında bulunan ve *Rotalar* içindeki rota üzerinde yer alan her link için *REZERVASYON_OLUŞTUR* := [*Kaynak = RotaÜzerindekiLink*, *RezervasyonBaşlangıçZamanı = İşBaşlangıçZamanı*, *RezervasyonBitişZamanı = İşBitişZamanı*, *RezervasyonBüyüklüğü = UzakRotaMinBantGenişliği*] mesajı; 3) *REZERVASYON_OLUŞTUR* := [*Kaynak = SeçilenİşlemDepolamaElemanı*:*Yazma*, *RezervasyonBaşlangıçZamanı = İşBaşlangıçZamanı, RezervasyonBitişZamanı = İşBitişZamanı*, *RezervasyonBüyüklüğü = UzakRotaMinBantGenişliği*] mesajı. DMS, RS'den, her bir rezervasyon oluşturma isteğine karşılık *REZERVASYON_ONAY* := [*RezervasyonBilgisi = &RezervasyonBilgisi*] veya *REZERVASYON_HATA* mesajları bekler. Mesajdaki *&RezervasyonBilgisi*; yukarıda tanımlanan üç farklı rezervasyon mesajı ile ilgili oluşturulan rezervasyon objelerinin bilgileridir. Bu durumda, *UzakRotaMinBantGenişliği = |Veri|/(İşBitişZamanı-İşBaşlangıçZamanı-UzakToplamLinkGecikmesi*) olarak hesaplanır. Formüldeki, *UzakToplamLinkGecikmesi* değeri, farklı sitelerde bulunan *KaynakDepolamaElemanı*'ndan *SeçilenİşlemElemanı*'na kadar olan rota üzerindeki linklerin toplam gecikmesini ifade eder.

c. Eğer *İşUzakVeriListesi*'ndeki herhangi bir LFN için RS'den *REZERVASYON_HATA* mesajı gelirse, iş ile ilgili veri transferleri karşılanamamıştır. Bu durumda, DMS DM'ye, DM'de LDS'ye *UZAK_VERİ_TRANSFERİ_HATA* mesajı iletir. Uzak veri tranferi istekleri ile ilgili yapılmış olan rezervasyonları silmek için de RS'ye *REZERVASYON_SİL* := [*RezervasyonBilgisi = #RezervasyonBilgisi*] veya *REZERVASYON_SİL* := [*RezervasyonBilgisi = &RezervasyonBilgisi*] mesajları gönderir. Veri transferi prosedürü sonlandırılır.

d. Eğer *İşUzakVeriListesi*'ndeki her PFN için RS'den *REZERVASYON_ONAY* := [*RezervasyonBilgisi* = *#RezervasyonBilgisi*] veya *REZERVASYON_ONAY* := [*RezervasyonBilgisi* = *&RezervasyonBilgisi*] mesajı gelirse, uzak veri transferi ile ilgili tüm rezervasyon objelerindeki bilgiler, *UVİ_RezervasyonBilgisi* veri yapısında depolanır ve veri transferi prosedürüne devam edilir.

8. DMS, veri depolama elemanı yöneticilerine kilit oluşturma mesajları gönderir ve veri depolama elemanı yöneticilerinden aşağıda belirtilen mesajları bekler.
   a. Eğer *LFN*, *VeriKopyalamaListesi*'nde var ise; verinin, *KaynakDepolamaElemanı*'nda [*ŞimdikiZaman*, *İşBaşlangıçZamanı*] aralığında kilitlenmesi için *KaynakDepolamaElemanı* yöneticisine, *VERİ_KİLİTLE* := [*Veri* = *LFN*, *KilitBaşlangıçZamanı* = *ŞimdikiZaman*, *KilitBitişZamanı* = *İşBaşlangıçZamanı*] biçiminde bir mesaj gönderir. Aynı verinin, kopyalandıktan sonra, *HedefDepolamaElemanı*'nda [*İşBaşlangıçZamanı*, *İşBitişZamanı*] aralığında kilitlenmesi için *HedefDepolamaElemanı* yöneticisine *VERİ_KİLİTLE* := [*Veri* = *LFN*, *KilitBaşlangıçZamanı* = *İşBaşlangıçZamanı*, *KilitBitişZamanı* = *İşBitişZamanı*] biçiminde bir mesaj gönderilir. DMS, *KaynakDepolamaElemanı* ve *HedefDepolamaElemanı* yöneticilerinden *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = **UzakVeriKilitBilgisi*] veya *VERİ_KİLİT_HATA* mesajlarını bekler. Buradaki **UzakVeriKilitBilgisi*; *KaynakDepolamaElemanı* veya *HedefDepolamaElemanı* yöneticilerine gönderilen veri kilit istekleri için hazırlanan veri kilitlerinin bilgilerini içerir. Eğer *LFN*, *KopyalanacakVeriListesi*'nde yok ise; *KaynakDepolamaElemanı*'na *VERİ_KİLİTLE* := [*Veri* = *LFN*, *KilitBaşlangıçZamanı* = *ŞimdikiZaman*, *KilitBitişZamanı* = *İşBitişZamanı*] biçiminde bir mesaj gönderilir. DMS, *KaynakDepolamaElemanı* yöneticisinden *VERİ_KİLİT_ONAY* :=

[*VeriKilitBilgisi = #UzakVeriKilitBilgisi*] veya *VERİ_KİLİT_HATA* mesajlarını bekler. Mesajdaki *#UzakVeriKilitBilgisi*; *KaynakDepolamaElemanı* yöneticisine gönderilen veri kilit isteği için hazırlanan veri kilitlerinin bilgilerini içerir.

b. Eğer *LFN*, *VeriKopyalamaListesi*'nde var ise ve eğer: 1) itme tabanlı dağıtık veri kopyalama modeli uygulanıyor ise, DMS, *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi = *UzakVeriKilitBilgisi*] mesajının bir kopyası ile istekte bulunan siteyi, (*HedefDepolamaElemanı* veya *SeçilenİşlemElemanı*'nın bulunduğu site) verinin bulunduğu kaynak site (*KaynakDepolamaElemanı*'nın bulunduğu site) içerisindeki LDRS'ye gönderir, 2) eğer itme tabanlı merkezi veri kopyalama modeli uygulanıyor ise, DMS, *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi = *UzakVeriKilitBilgisi*] mesajının bir kopyası ile istekte bulunan siteyi, DRS'ye gönderir. Eğer *LFN*, *VeriKopyalamaListesi*'nde var ise ve eğer: 1) itme tabanlı dağıtık veri kopyalama modeli uygulanıyor ise, DMS, *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi = #UzakVeriKilitBilgisi*] mesajının bir kopyası ile istekte bulunan siteyi, verinin bulunduğu kaynak site içerisindeki LDRS'ye gönderir, 2) itme tabanlı merkezi veri kopyalama modeli uygulanıyor ise, DMS, *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi = #UzakVeriKilitBilgisi*] mesajının bir kopyası ile istekte bulunan siteyi, DRS'ye gönderir.

c. Eğer *İşUzakVeriListesi*'ndeki herhangi bir LFN için *KaynakDepolamaElemanı* yöneticisinden –veya gerekiyor ise *HedefDepolamaElemanı* yöneticisinden- *VERİ_KİLİT_HATA* mesajı gelirse, iş ile ilgili veri transferleri karşılanamamıştır. Bu durumda, DMS DM'ye, DM'de LDMS'ye *UZAK_VERİ_TRANSFERİ_HATA* mesajı iletir. Uzak veri tranferi istekleri ile ilgili yapılmış olan rezervasyonları silmek için de RS'ye *REZERVASYON_SİL* := [*RezervasyonBilgisi =*

#RezervasyonBilgisi] veya *REZERVASYON_SİL* := [*RezervasyonBilgisi* = *&RezervasyonBilgisi*] mesajları gönderir. Ayrıca, DMS, *KaynakDepolamaElemanı* yöneticisine –ve gerekiyor ise *HedefDepolamaElemanı* yöneticisine- uzak veri transferi istekleri ile ilgili yapılmış olan veri kilitlerini silmek için *VERİ_KİLİT_SİL* := [*VeriKilitBilgisi* = **UzakVeriKilitBilgisi*] veya *VERİ_KİLİT_SİL* := [*VeriKilitBilgisi* = *#UzakVeriKilitBilgisi*] mesajları gönderir. Veri transferi prosedürü sonlandırılır.

d. Eğer *İşUzakVeriListesi*'ndeki her PFN için RS'den *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = **UzakVeriKilitBilgisi*] veya *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = *#UzakVeriKilitBilgisi*] mesajı gelirse, uzak veri transferi ile ilgili tüm veri kilit bilgileri, *UVİ_KilitBilgisi* veri yapısında depolanır ve veri transferi prosedürüne devam edilir

9. DMS DM'ye, DM'de LDMS'ye *UZAK_VERİ_İSTEĞİ_ONAY* := [*OnayID* = *UzakVeriTransferiOnayID*, *RezervasyonBilgisi* = *UVİ_RezervasyonBilgisi*, *KilitBilgisi* = *UVİ_KilitBilgisi*] mesajı gönderir.

2.4.2.1. Tek Aşamalı En Az Atlamalı

DDOA olarak Tek Aşamalı En Az Atlamalı (Single Pass Minimum Hop – SP_MinHop) algoritması kullanılabilir. SP_MinHop algoritması, veri transferi isteklerini aşağıdaki şekilde karşılar:

1. Çevrim dışı çalışır. Veri transferi istekleri, girdi olarak alınır.
2. GIS bileşeni sorgulanarak Veri Grid sisteminin ağ topolojisi öğrenilir.
3. Tüm veri transferi istekleri için
   a. Veri transferi isteğinin gereksinim duyduğu bant genişliği hesaplanır.
   b. Veri Grid sisteminin ağ topolojisinden, veri transferi isteği için yeterli bant genişliğine sahip olmayan linkler silinir.

c. Silinmeyen linkler kullanılarak Dijkstra'nın en az atlamalı algoritması çalıştırılır. Dijkstra'nın en az atlamalı yol algoritması ile, hedef siteden diğer tüm ulaşılabilir sitelere olan en az atlamalı yollar bulunur.
d. En az atlamayı sağlayan kaynak ile kaynak-hedef arasındaki en az atlamalı uygun rota seçilir.
e. Eğer en az bir uygun rota bulunmuş ise, bulunan rota üzerindeki linklerin kullanılan bant genişliklerini arttır. İstek için *KaynakDepolamaElemanı*, *Hedef* (*HedefDepolama Elemanı* veya *SeçilenİşlemElemanı*), *Rotalar* ve *RotalarÜzerindeKullanılanBantGenişliği*'ni bul. İsteği *KarşılanabilirİstekListesi*'ne ilave et.

2.5. Veri Kopyalama

Literatürdeki veri kopyalama algoritmaları çekme tabanlı, itme tabanlı dağıtık ve itme tabanlı merkezi omak üzere gruplanabilirler. Bu bölümde, önerilen modelin, literatürdeki farklı veri kopyalama modellerini nasıl desteklediği açıklanacaktır. Bu tez kapsamında geliştirilen veri kopyalama algoritmalarının detayları, 5. Bölüm'de anlatılacaktır.

Eğer veri depolama elemanı, yeni kopyalanacak veriyi depolayacak yeterli alana sahip değilse, yeterli veri depolama alanı oluşturmak için bir veri yenileme algoritması çalıştırılarak, veri depolama elemanından bazı veriler silinirler. Örnek bir veri yenileme algoritması olan LFU algoritması, veri depolama elemanından, en az sayıda istekte bulunulmuş olan verilerin silinmesini baz alır.

2.5.1. Çekme Tabanlı Veri Kopyalama

Çekme-tabanlı veri kopyalama algoritmaları, LDRS'nin bir parçası olarak gerçeklenmiştir. LDRS, LDMS'ye gelen ancak yerel olarak sağlanması mümkün olmayan veri transferi isteklerini değerlendirir ve çekme tabanlı veri kopyalama algoritması çalıştırarak, diğer sitelerden kopyalanacak olan verinin yerel olarak kopyalanıp kopyalanmayacağına karar verir. Verinin yerel olarak kopyalanıp

kopyalanmayacağı kararı, LDMS tarafından DM'ye uzaktan veri erişimi isteğinde bulunacağı zaman kullanılır.

1. LDRS, LDMS'den *VERİ_İSTEĞİ_GELDİ* := [*TransferEdilecekVeriListesi* = *İşVeriListesi*, *İstekSonZamanı* = *TransferSonZamanı*] mesajını alır.
2. LDRS'nin veri erişim istatistikleri güncellenir.
3. LDRS, *TransferEdilecekVeriListesi*'nde yer alan verilere ait veri erişim istatistiklerini baz alan bir çekme tabanlı dağıtık veri kopyalama algoritması çalıştırarak, transfer edilecek olan verilerden hangilerinin yerel olarak kopyalanması gerektiğini belirten *VeriKopyalamaListesi*'ni hazırlar.
4. LDRS, LDMS'ye *VERİ_KOPYALAMA_CEVAP* := [*KopyalanacakVeriListesi* = *VeriKopyalamaListesi*] biçiminde bir mesaj gönderir.

2.5.2. İtme Tabanlı Dağıtık Veri Kopyalama

İtme tabanlı dağıtık veri kopyalamada, her bir sitede bulunan LDRS, site içerisindeki verilere, diğer sitelerden gelen erişim isteklerinin istatistiğini tutar. LDRS'ler çevrim içi veya çevrim dışı çalışan veri kopyalama algoritmalarını çalıştırarak, kopyalanacak verilerin olup olmadığını bulur. Eğer bir verinin kopyalanmasına karar verir ise, verinin, hangi sitede kopyalanacağına da karar verir.

1. DMS, *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = **UzakVeriKilitBilgisi*] veya *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = *#UzakVeriKilitBilgisi*] mesajlarının bir kopyası ile istekte bulunan siteyi, verinin bulunduğu kaynak site içerisindeki LDRS'ye gönderir.
2. LDRS'nin bulunduğu sitede yer alan veriler için LDRS'nin veri erişim istatistikleri güncellenir.
3. LDRS, sitede yer alan verilere ait erişim istatistiklerini baz alan bir itme tabanlı dağıtık veri kopyalama algoritması olan RT_DIW algoritmasını çalıştırarak, sitede yer alan verilerden hangilerinin diğer sitelerde

kopyalanması gerektiğine ve kopyalanacak verilerin hangi sitelerde kopyalanacağına karar verir.

4. LDRS, kendisinin de içinde bulunduğu *KaynakSite*'de bulunan *KopyalanacakVeri*'nin başka bir *HedefSite*'de kopyalanması gerektiğine karar verdikten sonra, DM'ye, *UZAK_VERİ_TRANSFERİ_İSTEĞİ* := [*TransferEdilecekVeriListesi* = *KopyalanacakVeri*, *Hedef* = *HedefSite*, *TransferSonZamanı* = *VeriKopyalamaTransferiSonZamanı*, *KopyalanacakVeriListesi* = *KopyalanacakVeri*] biçiminde bir veri transferi isteği gönderir. *VeriKopyalamaTransferiBaşlangıçZamanı*, *ŞimdikiZaman*'dır. *VeriKopyalamaTransferiSonZamanı* ise *VeriKopyalamaTransferiBaşlangıçZamanı* + *VeriBüyüklüğü*/*VeriKopyalamaTransferHızı* olarak hesaplanır. Formülde belirtilen *VeriKopyalamaTransferHızı*, Veri Grid sistemi üzerinde bulunan linklerin en az bant genişliğinden küçük bir hızdır ve sistem yöneticisi tarafından sağlanan bir parametre kullanılarak hesaplanır. LDRS, DM'den herhangi bir mesaj beklemez.
5. DM'ye istekte bulunulan veriler ile ilgili istatistikler sıfırlanır.

2.5.3. İtme Tabanlı Merkezi Veri Kopyalama

İtme tabanlı merkezi veri kopyalamada, Veri Grid sisteminde bulunan merkezi bir servis olan DRS, tüm sitelerde bulunan tüm verilere yapılan veri erişim isteklerinin istatistiklerini tutar. DRS çevrim içi veya çevrim dışı çalışan veri kopyalama algoritmasını çalıştırarak, kopyalanacak veri olup olmadığını hesaplar. Eğer kopyalanacak veri var ise, hangi sitede kopyalanması gerektiğine karar verir.

1. DRS, DMS tarafından gönderilen *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = **UzakVeriKilitBilgisi*] veya *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = *#UzakVeriKilitBilgisi*] mesajlarının bir kopyasını alır.
2. DRS'nin veri erişim istatistikleri güncellenir.
3. DMS, Veri Grid sisteminde yer alan verilere ait erişim istatistiklerini baz alan bir itme tabanlı merkezi veri kopyalama algoritması

çalıştırarak, sitelerde yer alan verilerden hangilerinin (*KopyalanacakVeri*) diğer sitelerde kopyalanması gerektiğine ve kopyalanacak verilerin hangi sitelerde kopyalanacağına (*HedefSite*) karar verir.

4. DRS, DM'ye, *UZAK_VERİ_TRANSFERİ_İSTEĞİ* := [*TransferEdilecekVeriListesi* = *KopyalanacakVeri*, *Hedef* = *HedefSite*, *TransferSonZamanı* = *VeriKopyalamaTransferiSonZamanı*, *KopyalanacakVeriListesi* = *KopyalanacakVeri*] biçiminde bir veri transferi isteği gönderir. *VeriKopyalamaTransferiSonZamanı* Bölüm 2.5.2'de anlatıldığı gibi hesaplanır.

2.6. Ön Rezervasyon Sistemi

Bu çalışmada önerilen Veri Grid modeli, temel Grid altyapısını oluşturan elemanların önceden rezerve edilebildiğini varsaymaktadır. Önerilen modelde bulunan rezervasyon servisleri, ön rezervasyon sisteminin çatısını oluşturmaktadırlar. Bu servisler, rezervasyon isteklerinin karşılanması için gerekli olan fonksiyonları yerine getirirler.

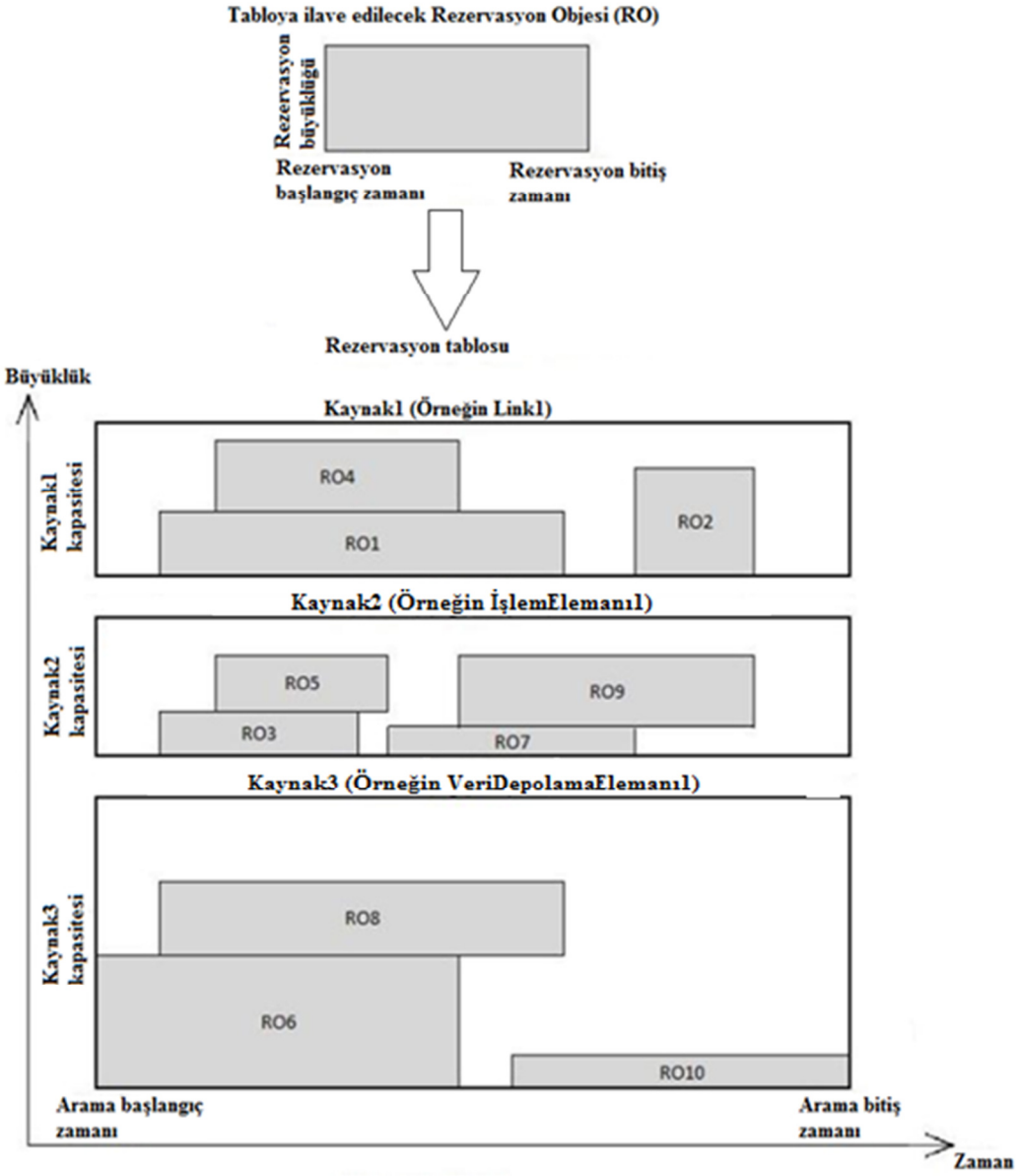
Veri Grid sistemi içerisindeki sitelerde bulunan LRS'ler, öncelikle site içerisindeki işlem, veri depolama ve ağ elemanlarının rezervasyonundan sorumludur. Merkezi rezervasyon servisi olan RS ise, öncelikle siteler arası ağ kaynaklarının rezervasyonundan sorumludur. RS ile LRS'ler işbirliği içerisinde çalışırlar. RS, gelen rezervasyon isteklerini site içindeki LRS bileşenlerine iletebilir veya LRS'den belirli zaman aralığında uygun olan kaynakları sorgulayabilir.

Rezervasyon servislerinde, birer rezervasyon tablosu bulunur. Örnek bir rezervasyon tablosu, Şekil 2.7'de sunulmuştur. Şekilde de görüldüğü üzere, rezervasyon tablolarında, rezervasyon servisinin kontrol alanında bulunan kaynaklar için birer rezervasyon satırı bulunur. Her kaynak için oluşturulmuş olan rezervasyon objeleri, o kaynağın satırında saklanırlar.

Rezervasyon objeleri altı değişken içerirler:

1. *REZERVASYON_ID*: Rezervasyon tanımlama numarası.
2. *Başlangıç zamanı*: Rezervasyonun başlangıç zamanı.

3. *Bitiş zamanı*: Rezervasyonun bitiş zamanı.
4. *Tip*: İşlem, veri depolama ve ağ elemanı çeşitlerinden birisi.
5. *Kaynak adresi*: Rezervasyon yapılan kaynağı gösteren bir işaretçi.
6. *Büyüklük*: Rezervasyonun ne kadarlık bir alan için yapıldığı. İşlem elemanları için işlem bant genişliği, veri depolama elemanları için depolama okuma/yazma bant genişliği, ağ elemanları için de rezerve edilen bant genişliğini ifade eder.



Şekil 2.7. Örnek rezervasyon tablosu

Rezervasyon servislerine, rezervasyon ile ilgili talepler, çizelgeleme servisleri (SSS) ve veri yönetim servisleri (LDMS ve DMS) tarafından oluşturulur. Rezervasyon oluşturma süreci aşağıda belirtildiği üzere gerçekleştirir:

1. İş çizelgeleme ve veri yönetimi servisleri, işlem, veri depolama ve ağ elemanları için rezervasyon istekleri oluşturmadan önce, rezervasyon tablolarında, rezervasyon tablolarında *uygun pencere*leri sorgularlar. Uygun pencereler, rezerve edilmek istenen büyüklüğü, rezervasyon sorgu mesajında belirtilen son zamandan önce karşılayabilecek kapasitedeki, rezervasyon tablolarındaki boş alanlardır. Sorgulama isteği için *REZERVASYON_SORGULA* := [*RezervasyonBüyüklüğü*, *Tip*, *KaynakListesi*, *RezervasyonİlkZamanı*, *RezervasyonSonZamanı*] biçiminde sorgu istek mesajları kabul edilir. Bu istek, *KaynakListesi*'ne ait Rezervasyon tablolarında *RezervasyonSonZamanı*'ndan başlayarak *RezervasyonİlkZamanı*'a kadar aynı *Tip*'teki tüm kaynaklarda *RezervasyonBüyüklüğü* kadarlık alanın taranmasını gerektirir.
2. Rezervasyon servisi, sorgulamayı yapan servislere, en az *RezervasyonBüyüklüğü* alanına sahip ve *RezervasyonSonZamanı*'ndan önce tamamlanmış olan pencereleri, *REZERVASYON_SORGU_CEVAP* := [*Kaynak, RezervasyonPencereBaşlangıçZamanı*, *RezervasyonPencereBitişZamanı*, *RezervasyonPencereBüyüklüğü*] listesi biçiminde bir mesaj ile iletir. Bu mesajda belirtilen her bir uygun pencere, *Kaynak*'ın *RezervasyonPencereBaşlangıçZamanı*'ndan *RezervasyonPencereBitişZamanı*'na kadar *RezervasyonPencereBüyüklüğü*'ne kadar rezerve edilebileceğini ifade eder. Bu pencere, rezervasyon sorgu mesajında belirtilen *RezervasyonBüyüklüğü*'nden büyük olmak zorundadır. Bir *Kaynak* üzerinde bir veya birden fazla uygun pencere olabilir. Eğer hiç uygun uygun pencere bulunamaz ise *REZERVASYON_SORGU_CEVAP* := *REZERVASYON_HATA* mesajı iletilir.
3. İş çizelgeleme ve veri yönetim servisleri, uygun pencereler arasından bir seçim yapar ve rezervasyon servisine *REZERVASYON_OLUŞTUR* :=

[*Kaynak*, *RezervasyonBaşlangıçZamanı*, *RezervasyonBitişZamanı*, *RezervasyonBüyüklüğü*] şeklinde bir rezervasyon isteğinde bulunur.

4. *REZERVASYON_OLUŞTUR* isteği ile, rezervasyon tablolarında önceden var olmayan bir rezervasyon, tabloya eklenmeye çalışılmaktadır. Eğer rezervasyon oluşturulabilir ise, istekte bulunan iş çizelgeleme ve veri yönetimi servislerine oluşturulmuş olan rezervasyonun adresini içeren bir *REZERVASYON_ONAY* := [*RezervasyonBilgisi*] mesajı iletilir. Rezervasyon için uygun aralıkların sorgulanması ile rezervasyonun oluşturulması arasında, rezervasyon tablosuna başka ilaveler yapılması veya kaynağın kullanılamaz duruma gelmesi gibi bir sebeple rezervasyon oluşturulamamış ise *REZERVASYON_HATA* mesajı gönderilir.
5. Çizelgeleme servisleri, herhangi bir sebeple, rezervasyonların silinmesine karar verirse, rezervasyon servislerine *REZERVASYON_SİL* := [*RezervasyonBilgisi*] mesajı gönderirler.

Tanımlanan üç fonksiyonun hızlı bir şekilde çalıştırılabilmesi için, aralık ağacı (interval tree) yaklaşımı kullanılabilir. Aralık ağacı, aralıkları tutan bir düzenlenmiş ağaç veri yapısıdır. Verilen aralıklar ile çakışan aralıkların hızlı ve verimli bir şekilde bulunması için kullanılır. Aralık ağacı, [60] ve benzeri kaynaklarda anlatılan kırmızı-siyah ağaç veri yapısı kaynak kodları kullanılarak verimli şekilde gerçeklenebilir.

2.7. Önerilen Modelin Desteklediği Özellikler

Önerilen modeli daha iyi analiz etmek için, literatürde kabul görmüş olan sınıflandırmalara göre yerini belirtmemiz gerekmektedir. Bu bölümün birinci kısımda, GridFTP benzeri bir çok standardın oluşmasını sağlayan OGF [62]'nin tanımladığı OGSA mimarisi, ikinci kısmında ise Venugopal-Buyya-Ramamohanarao sınıflandırması [74] kullanılacaktır.

2.7.1. OGSA

OGSA'nın son versiyonu olan OGSA v1.5'e göre Grid sistemleri aşağıdaki özelliklere sahip olmalıdır:

1. Müşterek çalışma ile dinamik ve heterojen ortam desteği: Grid sistemine dâhil olan farklı, heterojen ve dağıtık yapıdaki kaynakların işbirliği içinde çalıştırılabilmesi için gerekli olan özelliklerdir. Bu özellikler, önerilen modelde karşılanmaktadır.
    a. Kaynak sanallaştırılması (virtualization): Heterojen kaynakların yönetimini kolaylaştırmak için gerekli bir özelliktir. Önerilen modelde, tüm kaynaklar için ilgili yönetim mekanizmaları tanımlanmıştır.
    b. Ortak yönetim yetenekleri: Heterojen sistemlerin yönetimini kolaylaştırmakta kullanılır. Önerilen modeldeki katmanlı mimari, ortak yönetim yeteneklerini arttırmaktadır.
    c. Kaynak bulma ve sorgulama: Sisteme dâhil olan kaynakların kullanılabilrmesi için, istenen özelliklere sahip olan kaynakları arama ve bulma mekanizmaları olmalıdır. Önerilen modelde kaynakların bulunması ve sorgulanması için kaynak yöneticileri, rezervasyon ve bilgi servisleri kullanılır.
2. Organizasyonlar arası kaynak paylaşımı: Grid sistemleri yekpare sistemler değillerdir. Sisteme dâhil olan farklı özellikteki sitelere bağlı kaynaklardan oluşmaktadır. Bu kaynaklar, ortak amaçlar doğrultusunda işlev görebilmesi için aşağıdaki özelliklere sahip olmalıdır. Önerilen modelde, bu özellikler sağlanmaktadır.
    a. Global ad uzayı (namespace): Veri ve kaynak erişimi için bir ön şarttır. Önerilen modelde global ad uzayı kullanıldığı varsayılmıştır. Bu sayede, LFN ile tanımlanmış olan verilerin kopyalarının bulunduğu kaynaklar belirlenebilmektedir.
    b. Metadata servisleri: Grid sistemine bağlı birimlerin bulunması, çalıştırılması ve takip edilmesi için önemlidir. Önerilen modelde tüm servisler, çalışmaya başlamadan önce kendilerini

tanıtmaktadırlar. Pasif olan sistem kaynakları da, aktif olan ve kendilerini tanıtmış olan yönetici servisler tarafından kontrol edilmektedir.

c. Site özerkliği: Siteler arası kaynaklara ulaşırken yerel kontrol mekanizmalarının da varlığının devam etmesi gereklidir. Önerilen modelde, her bir sitede bulunan yerel yönetim elemanları, farklı algoritmaları çalıştırabilmektedirler.

d. Kaynak kullanım verileri: Kaynak kullanımı ile ilgili veriler ve istatistikler siteler arasında paylaştırılabilmelidir. Önerilen modelde yer alan bilgi servisleri, kaynak kullanım verilerini tutmakta ve gerektiğinde paylaşmaktadır.

3. Optimizasyon: Kullanıcının isteklerini karşılamak için verimli bir çizelgeleme yapmayı gerektirir. Önerilen modelde, (yerel ve global) iş çizelgeleme ve (yerel ve global) veri yönetimi servisleri, gerekli optimizasyonları yaparlar.
4. Servis Kalitesi (Quality of Service - QoS) güvencesi: İş çizelgeleme ve veri servisleri, istenen QoS güvencesini sağlamak için çalışmalıdırlar. Yararlanılırlık (availability), güvenlik, performans gibi farklı QoS gereksinimleri istenebilir. Önerilen modelde, son zaman QoS olarak tanımlanmış olup, sisteme arz edilen işlerin ve veri transferlerinin belli bir son zamandan önce tamamlanmış olması beklenmektedir.
5. İş çalıştırılması: Kullanıcı tarafından sisteme sunulan işlerin işlem elemanlarında çalıştırılması sürecindeki ara işlemlerin yönetilmesi için iyi tanımlanmış süreçler, önerilen modelde mevcuttur.
   a. Farklı iş tiplerinin desteklenmesi: Literatürde süreç-tabanlı, bağımsız işler, iş torbası, iş akışları gibi farklı iş modelleri önerilmiştir. Bu çalışmada, bağımsız işler uygulama modeli varsayılmıştır. Diğer iş modellerinin entegrasyonu, ileri araştırma konusudur.
   b. İş yönetimi: İşlerin, tüm yaşam süreçleri boyunca yönetilebilmesi gereklidir. Önerilen modelde, işlerin sunulması, kabul edilmesi, çizelgelenmesi, dağıtılması, verilerin toplanması, başlatılması ve

tamamlanması aşamalarının hangi servisler tarafından gerçekleştirileceği net biçimde anlatılmıştır.

c. Çizelgeleme: Önerilen modelde, iş öncelikleri veya kaynak kullanım oranları gibi veriler göz önünde bulundurularak, işlerin en verimli şekilde çalıştırılabilmesi için uygun çizelgeleme algoritmaları çalıştırılabilmektedir. Ayrıca, yeni çizelgeleme metotlarının sisteme hızlı entegrasyonu için hazır bir tanımlama sunulmuştur.

d. Kaynakları kullanıma hazırlama: Kaynak tahsisi ve yapılandırılması gibi karmaşık süreçler otomatik olarak yapılabilmelidir. Rezervasyon servisleri ve iş başlatma servisleri gibi yapılar ile, kaynaklar kullanıma hazırlanmaktadır.

6. Veri servisleri: Farklı alanlardaki işlem gereksinimlerini karşılamak için kurulan Grid sistemlerinin çalıştırdığı işlerin gereksinim duyduğu veri miktarı artma eğilimi göstermektedir. Yüksek miktardaki bu verilerin paylaşımı, entegrasyonu ve dağıtımı gerekmektedir. Önerdiğimiz model, verilerin oluşturulduğu ana bir merkezin varlığını ve diğer merkezlerin bu verileri sadece okumaya ihtiyaç duyduklarını varsaymaktadır.

   a. Yönetim mekanizmaları: Verilerin kimler tarafından kullanılacağı, ne kadar kaynak ayrılacağı gibi mekanizmalar, önerilen modelde bulunan çizelgeleme servisleri tarafından yönetilirler. İşlenmiş verilerin tekrar dağıtımı olmadığı için veri tutarlılığı gibi konular kapsam dışı tutulmuştur. Ayrıca güvenlik ile ilgili yönetim mekanizmaları da yine bu çalışmanın kapsamı dışındadır.

   b. Veri depolama: Verilerin depolandığı disk ve benzeri sistemlerin modelleri, bu çalışmaya dâhil edilmiştir. Depolama elemanları, kapasite ve okuma/yazma hızları ile modellenmektedir.

   c. Veri erişimi: Sistem üzerindeki tüm verilere, sisteme dâhil olan tüm sitelerden erişim yapılabildiği varsayılmıştır. Veri güvenliği kapsam dışında tutulmuştur.

   d. Veri transferi: Verilerin yüksek bant genişliği ile kaynaktan hedefe ulaştırılması gerekmektedir. Sistem kaynaklarının verimli

kullanılması için gerekli olan veri yönetimi, bilgi, kopya kataloğu, veri transferi ve rezervasyon servisi gibi servisler, bu çalışma kapsamında tanımlanmıştır ve anlatılmıştır.

e. Veri konum yönetimi: Sistem üzerinde verilerin hangi depolama kaynaklarında saklanması gerektiği, önerilen modelde bulunan veri kopyalama servisleri tarafından yönetilir.
f. Veri güncelleştirmesi: Sistem üzerinde ihtiyaç duyulan verilerin sadece okuma izinli veriler olduğu varsayılmıştır. Dolayısı ile veri güncelleştirme mekanizmalarına ihtiyaç duyulmamıştır.
g. Veri kalıcılığı: Verilerin hangi kaynaklarda depolanacağına ve depolama kaynaklarında ne kadar tutulacağına ilişkin kararlar, veri yönetim ve kopyalama servisleri tarafından verilir.
h. Veri federasyonu: Sistemde bulunan tüm verilerin dağıtıldığı bir merkez bulunmakla birlikte, verilerin farklı sitelerde kopyaları bulunabilir. Sistemde bulunan kopyaların yerlerini bulmak için kopya konum göstergesi, kopya kataloğu ve kopya servisleri kullanılır. Hangi kopyanın kullanılmasının uygun olduğuna, veri yönetim servisleri karar verir.

7. Güvenlik: Verilerin doğruluğu ve yetki düzeyi ile ilgili yönetimsel unsurlar, bu çalışmanın kapsamı dışında tutulmuştur.
8. Yönetimsel maliyetin azaltılması: Servislerin bünyesinde bulunan algoritmaların karmaşıklığına göre yönetimsel maliyet değişkenlik göstermektedir. Düşük karmaşıklığa sahip olan algorimaların tercih edilmesi, yönetimsel maliyeti düşürecektir.
9. Yararlanılırlık (Availability): Sistem kaynaklarının arızalanması ve arıza giderilmesi gibi unsurlar, bu çalışmanın kapsamı dışında tutulmuştur.
10. Kullanım kolaylığı ve genişletilebilirlik: Bu çalışmada önerilen modele, farklı algoritmalar ve kullanma durumları (use case), tanımlı servisleri kullanarak rahatlıkla ilave edilebilir.

2.7.2. Venugopal-Buyya- Ramamohanarao Sınıflandırması

[74] çalışmasında önerilen sınıflandırma yapısına göre Veri Grid sistemleri, organizasyon modeli, veri transferi mekanizması, veri kopyalama metotları ve çizelgeleme mimarisi olmak üzere farklı özelliklerine göre sınıflandırılabilirler. Önerilen model, çok farklı özelliklere sahip Veri Grid sistemlerinin oluşturulmasını desteklemektedir. Bu bölümde, önerilen model ile desteklenebilecek Veri Grid sistemleri sınıfları tanıtılacaktır.

2.7.2.1. Veri Grid organizasyon modeli

[74]'e göre Veri Grid sistemleri organizasyonuna beş farklı açıdan bakılabilir: model, kapsam, sanal organizasyon, veri kaynakları ve yönetim.

1. Model: Model, Grid sistemi üzerindeki verilerin organizasyonu ile ilgilidir. Literatürde bulunan birli (monadic), federasyon, hiyerarşik veya hibrit gibi farklı veri organizasyon modellerinin tümü, bu çalışmada önerilen model tarafından desteklenir.
2. Kapsam: Önerilen Veri Grid sistemi modeli, farklı alıtların (domain) bulunabileceği ortak bir platform sunacak yapıdadır.
3. Sanal Organizasyon: *İşbirlikçi (collaborative)* sanal organizasyonlar veya siteler desteklenir. Grid sistemi, sisteme arz edilen işleri çalıştırmak için sistem kaynaklarını işbirliği içerisinde kullanır.
4. Yönetim: Yönetim stili, günümüz Veri Grid sistemlerinde de kullanıldığı üzere, *yönetilmiş* (*managed*)'tir.

2.7.2.2. Veri transferi mekanizması

Veri transferi mekanizmasına dört farklı açıdan bakılabilir:

1. Fonksiyon: İki site arasındaki veri transferi, GridFTP benzeri bir transfer protokolü ile gerçekleştirilir.
2. Güvenlik: Tüm veri erişimleri ve transferleri, uygun biçimde kimlik denetlenmiş (authenticated) olmaktadır ve erişim kontrol kurallarına uymaktadır.

3. Hata toleransı: Önerilen modelde hata toleransı için herhangi bir mekanizma bulunmamaktadır. Transfer başladığında, başarılı biçimde tamamlanacağı varsayılmaktadır.
4. Transfer modu: Önerilen model, verilerin bir veya birden fazla kanaldan paralel veri transferi gibi gelişmiş özellikler desteklenerek gönderildiğini varsaymaktadır.

2.7.2.3. Veri kopyalama

[74]'te, veri kopyalama, kopya mimarisi ve kopyalama stratejisi olmak üzere iki alt sınıf olarak tanımlanmıştır. Kopya mimarisi alt sınıfına aşağıdaki bakış açıları ile bakılabilir:

1. Model: Önerilen modelde, bir ana verinin birden fazla Veri Grid sitesinde kopyalarının saklandığı *merkezi* kopyalama sistemi varsayılmıştır. Ayrıca, kopyalanacak olan verinin sadece bir kere yazılabildiği varsayılmıştır.
2. Topoloji: Kopyaların yerleştirileceği Veri Grid sitelerinin organizasyonu, gerçeklenecek olan veri kopyalama algoritmasına bağlıdır. Önerilen model, tüm veri kopyalama topolojileri (hiyerarşik, federatif ve hibrit) için uygundur. Ayrıca, çekme ve itme tabanlı veri kopyalama algoritmaları için de uygundur.
3. Depolama entegrasyonu: Depolama sistemleri, depolama yöneticileri tarafından kontrol edilirler. Bunun sonucunda, kopya yönetim sistemi ile depolama sistemleri arasındaki iletişim veri transfer servisi ile beraber depolama yöneticileri üzerinden yapılır. Bu durumda, önerilen modelde gevşek birliktelikli (loosely coupled) depolama entegrasyonu mevcuttur.
4. Transfer protokolleri: Açık veya kapalı protokoller desteklenir. Verilerin herhangi bir kaynak sitesinden herhangi bir hedef sitesine gönderilebildiği varsayılmıştır.
5. Veriler hakkında bilgiler (metadata): Önerilen modelde, tipik olarak Veri Grid sistemlerinde bulunan metadata kataloğunun bulunmasına gerek yoktur.

6. Kopya güncelleme aktarımı: Ana kopya ile Grid sistemi üzerindeki diğer kopyalar birbirleri ile birebir aynı olduğundan dolayı kopya güncelleme aktarımı modellenmemiştir.
7. Katalog organizasyonu: Sistem üzerindeki verilerin yerlerinin bulunması için tüm sitelerde yerel kopya kataloglarının ve Grid sisteminin bütünü için de global kopya kataloğunun var olduğu varsayılmıştır.

Kopyalama stratejisi alt sınıfında ise aşağıdaki bakış açıları mevcuttur:

1. Metot: Kopyalama stratejileri verilerin ne zaman ve nerede kopyalanacağına karar verirler. Önerilen modelde, statik ve dinamik kopyalama stratejileri desteklenir.
2. Tanelilik (granularity): Özgün veriler, Veri Grid sistemi üzerinde farklı sitelere kopyalanabilir.
3. Amaç fonksiyonları: Yerellik veya popülerlik gibi amaç fonksiyonlarından herhangi birisi, kopyalama algoritmasında amaç fonksiyonu olarak kullanılabilir.

2.7.2.4. Çizelgeleme

Çizelgelemeye aşağıdaki beş bakış açıları ile bakılabilir:

1. Uygulama modeli: Bir veya birden fazla veri transferi isteğini içeren *bağımsız işler*'in çizelgelenmesi desteklenir. Bu modelde her iş, birbirinden bağımsız olarak çizelgelenir. Her işin çalışması için gerekli olan sistem kaynaklarını elde etmesi amaçlanır.
2. Veri kopyalama: Çizelgeleme servisleri tarafından işler çizelgelenirken, veri kopyalama servisleri de verilerin kopyalanıp kopyalanmayacağına karar verirler. İş çizelgeleme algoritmaları da, iş için gerekli verilerin bulunduğu sitelere öncelik verebilir. Önerilen modelde, veri kopyalama ile iş çizelgeleme birlikte çalışabilmektedir.
3. Fayda fonksiyonu: Veri Grid sistemi, birincil planda, gerçek-zamanlı operasyonlar göz önünde bulundurularak hazırlanmıştır. Bu sebeple fayda fonksiyonu olarak *QoS* tanımlanmıştır. Tüm iş ve veri transferleri,

önceden belirlenmiş son zamanlarından önce tamamlanmalıdırlar. Aksi durumda sistemden çıkarılırlar.

4. Yerellik (Locality): Mekansal (spatial) ve zamansal (temporal) yerelliğin ikisi de desteklenir. Mekansal yerellik, işi, gereksinim duyduğu verilerin yakınındaki işlem elemanlarına yerleştirilmesidir. Zamansal yerellik ise, halihazırda sistemde çalıştırılan işlerden sonra sisteme arz edilecek ve aynı verilere ihtiyaç duyacak olan işlerin, halihazırdaki işlerin yakınındaki işlem elemanlarına yerleştirilmesidir.

3. DGridSim SİMÜLATÖRÜ

Veri Grid sistemleri, yüksek veri ihtiyacı bulunan uygulamaların çalıştırılmasında sıklıkla kullanılan dağıtık sistemlerdir. Ancak, sistemin performansına etki eden çok fazla sayıda parametre olması ve bu parametrelerin bir kısmının da dinamik olarak değişmesi nedeni ile, sistem performansının değerlendirilmesinde kullanılabilecek analitik modellerin geliştirilmesi çok zordur. Analitik modellerin yokluğunda, performans incelemeleri için en doğru sonuçlar, gerçek uygulamalar gerçek Grid sistemleri üzerinde çalışırken yapılan performans ölçümleri ile elde edilir. Günümüzde, bu amaç için özel olarak gerçeklenmiş olan sistemler bulunmaktadır [78].

Gerçek sistemlerle performans ölçümü daha doğru sonuçlar verse de, simülasyon programlarıyla performans ölçümü ile karşılaştırıldığında, bazı dezavantajları bulunmaktadır. Öncelikle, gerçek sistemler üzerinde ölçüm yapmanın uzun zaman alması, fazla sayıda ölçüm yapmanın önünde bir engel teşkil etmektedir. Simülasyon programları ile yapılan performans ölçümleri daha az zaman almakta ve daha fazla sayıda ölçüm yapmak mümkün olmaktadır. Ayrıca, gerçek sistemler üzerinde sistem parametreleri üzerinde değişiklikler yapmak zordur. Örneğin, link sayısının veya link kapasitelerinin performans üzerine etkisini gerçek sistemler üzerinde ölçebilmek için link eklemek/çıkarmak veya kapasitelerini değiştirmek gibi oldukça zor ve zaman alıcı işlemler yapmak gerekmektedir. Aynı ölçümler simülasyon ortamlarında, parametre dosyasındaki bir değişkeni değiştirmek kadar kolay bir şekilde yapılabilmektedir. Tüm bunlara ilave olarak, uygulamaların performans ölçümleri gerçek bir sistem üzerinde çalışırken yapılmış olsa bile, farklı zamanlarda tekrarlanan ölçümlerin sonuçları çok farklı olabilmektedir. Çünkü, sistem kaynaklarına binen yük ve kaynakların kullanılabilir kapasiteleri zamana bağlı olarak değişmektedir. Simülasyon ortamları ise tekrar edilebilir bir ölçüm platformu sunarlar.

Simülatörlerle çok sayıdaki veri transferi isteğinin hızlı ve doğru biçimde simülasyonunun yapılabilmesi için, literatürde iki farklı yaklaşım bulunmaktadır: *paket tabanlı* ve *akış tabanlı* simülatörler. Paket tabanlı simülatörlerde, veriler önce paketlere ayrılır ve bu paketlerin ağ üzerindeki transferlerinin simülasyonu

yapılır. Bu simülatörler, akış tabanlı simülatörlere göre daha gerçekçi sonuçlar vermektedir. Ancak, oldukça yavaş çalışan bu simülatörler, az sayıda veri transferi isteğinin simülasyonunda kullanılabilirler.

Akış tabanlı simülatörlerde ise, her veri transferi akış veya akışlar topluluğu biçiminde modellenir. Bu simülatörler, paket tabanlı simülatörlere göre oldukça hızlı çalışırlar ve daha fazla sayıda veri transferi isteğinin daha büyük sistemler üzerinde simülasyonu için kullanılırlar. Ancak paket tabanlı simülatörler kadar gerçekçi sonuç vermezler.

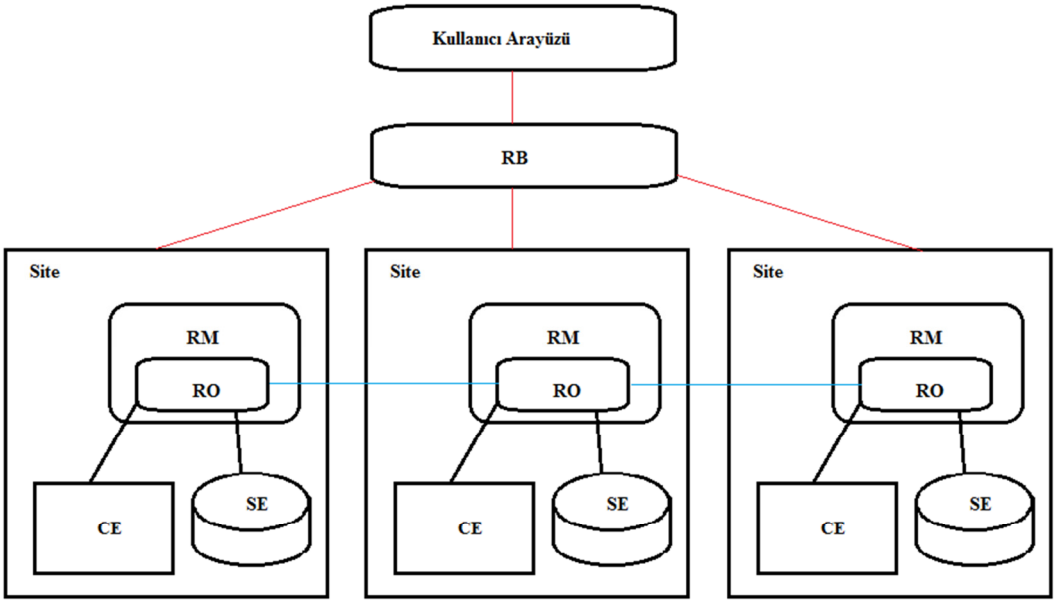
2. Bölüm'de önerilen modelin gerçeklenebilirliğini görmek ve farklı algoritmalar/parametreler için performans ölçümleri yapabilmek amacı ile DGridSim isimli bir Veri Grid simülatörü geliştirilmiştir.

DGridSim, modülerlik, genişletilebilirlik ve katmanlı mimari gibi gereksinimleri karşılayacak şekilde tasarlanmış olup; alt katmanların sağladığı fonksiyonlardan üst katmanların faydalandığı bir yapı benimsenmiştir. Simülatör, önerilen Veri Grid sistemi modeline benzer şekilde, temel Grid altyapısı, iletişim, Veri Grid servisleri ve uygulama gibi katmanlardan oluşmaktadır. İş çizelgeleme, veri dağıtımı ve veri kopyalama algoritmalarının da performans ölçümlerinin yapılabileceği platformda, katmanlararası etkileşimler ve ön rezervasyon sistemi gibi bileşenler, tezin ikinci bölümünde anlatıldığı şekilde gerçeklenmiştir.

3.1. İlgili Çalışmalar

Grid sistemleri, [76, 79, 80] gibi çalışmalarda belirtilen genel amaçlı simülatörler ile simülasyonu yapılamayacak kadar karmaşık yapılardır. Literatürde, Grid sistemlerinin simülasyonu için özel olarak yazılmış birçok simülatör bulunmaktadır: Optorsim, GridSim, GridNet, SimGrid, MicroGrid, Bricks, ChiSim, GangSim ve Sugato Bagchi.

Optorsim: Optorsim [81], Grid sistemleri için yazılmış en çok ilgi gören simülatörlerden birisidir. EU DataGrid projesi [9] kapsamında dinamik kopyalama senaryolarını test etmek amacı ile Java dilinde gerçeklenmiştir. Optorsim, veri kopyalama ile iş çizelgelemenin birlikte çalışacağı biçimde dizayn edilmiştir. Optorsim, Veri Grid üzerinde çalışan parçaların aralarındaki bağlantıları mümkün olan en gerçekçi biçimde modelleyecek şekilde geliştirilmiştir. Simülatörün kullandığı modelde, Grid sistemi, her birisinde işlem ve depolama kaynakları bulunan sitelerden oluşur. Sistem, Şekil 3.1'de gösterildiği gibi, şu temel parçalardan oluşur: İşlem Elemanı (Computing Element - CE), Depolama Elemanı (Storage Element - SE), Kaynak Simsarı (Resource Broker - RB) ve Kopya Eniyileştiricisi (Replica Optimizer - RO). Yapılacak işler, RB tarafından Grid'e arz edilir. RB, işleri, toplam işlem hacmini en fazla düzeye çıkaracak biçimde, CE'ler üzerine çizelgeler. Herbir bölgedeki Kopya Yöneticisi (Replica Manager - RM) bileşeni, bölgeler arasındaki veri akışını ve işlem ve depolama kaynakları ile Grid arasındaki arayüzü yönetir. RM'ler içerisindeki RO'lar ise veri kopyalarının dinamik olarak oluşturulması ve silinmesinden sorumludur.



Şekil 3.1. Optorsim Veri Grid yapısı

GridSim: GridSim [82], öncelikli olarak sisteme arz edilen işler için gerekli olan çizelgeleme algoritmalarını test etmek için tasarlanmış bir simülatördür. Veri Grid sisteminin simülasyonunu içermez. Genel amaçlı bir kesikli olay simülasyon paketi olan SimJava üzerinde yazılmıştır. Simülasyon sonuçlarında oldukça detaylıdır. Grid üzerindeki kaynakların yönetimi için kaynakların satılması ve alınmasını temel alan bir ekonomik modeli (Nimrod-G [4]) kullanır.

GridSim gerçekçi ağ modelleri oluşturmak için:

1. Kullanıcıların ağ yapısını biçimlendirmelerine izin verir.
2. Verileri ağ üzerinden küçük paketler halinde gönderir.
3. Ağın modellenmesinde, ağ üzerinde arka plan trafiğinin oluşturulmasına olanak sağlar.
4. Paketleri gönderirken farklı servis seviyeleri kullanır.

Grid üzerindeki herbir işin Java Virtual Machine'de ayrı bir iş parçacığı (thread) üzerinde yapılandırılmış olması ve ağ modelinin paket tabanlı çalışması, simülatörün geniş ölçekli denemelerde çok yavaş çalışmasına yol açar.

GridSim simülatörü, 2. Bölüm'de anlatılan Veri Grid sisteminin organizasyonuna benzer bir şekilde, katmanlı bir mimari kullanmaktadır. Bu sayede, mevcut yapıya yeni parça veya katman eklemek mümkün olmaktadır. En

alt tabakada SimJava paketi bulunmakta, bu katmanın üzerinde ise temel altyapı katmanı yer almaktadır. Üçüncü ve dördüncü katmanlarda, İşlem ve Veri Grid sistemlerinin modellenmesi ve simülasyonu ile ilgili modüller yer almaktadır. Beşinci ve altıncı katmanlarda ise, kullanıcıların simülatörü genişletmesi için gerekli altyapı yer almaktadır.

GridSim simülatörünü popüler yapan özelliklerden bazıları aşağıda sıralanmaktadır:

1. Farklı kaynak özellik ve çeşitlerinin modellenmesini destekler.
2. Gerçek süperbilgisayarlardan elde edilen iş yüklerinin simülasyonlarına olanak tanır.
3. Kaynakların kullanılmasında rezervasyon bazlı bir mekanizma varsayar.
4. Kullanıcı tarafından arz edilen işler, uzay veya zaman paylaşımlı olarak çizelgelenebilir.
5. İşlem ve/veya veri ağırlıklı işlerin simülasyonunu gerçekleştirebilir.

GridNet: GridNet [54], ns (Network Simulator) programı üzerinde C++ dilinde yazılmıştır. Hiyerarşik bir yapı varsayar. Bu yapıda, en alt seviyedeki elemanlar işleri çalıştırır, üstteki elemanlar ise depolama birimleridirler.

GridNet simülatörü; herbir birimin depolama kapasitesi, yerel verilerin organizasyon modelleri, işlemci bant genişlikleri, komşu listeleri ve kopyalama yapılacak olan birimler gibi farklı simülatör parametrelerinin, kullanıcı tarafından değiştirilmesine olanak tanır. Simülatör üzerindeki birimler; istemci, sunucu ve ön bellek olabilirler. Sunucu birimleri, Veri Grid sistemi üzerindeki verilerin ana kaynaklarıdır. Ön bellek birimleri, yüksek depolama kabiliyetine sahip ve bazı verilerin kopyalarının saklanacağı yerlerdir. İstemci birimleri ise, veri transferi isteklerinin kaynağıdırlar ve çok sınırlı veri depolama kabiliyetine sahiptirler. Herbir GridNet birimi; depolama elemanı, kopya yöneticisi ve kopya yönlendirme tablosu içerir. Kopya yöneticisi, kopyaların ne zaman oluşturulup ne zaman silineceğine karar verir. Bu kararları verirken herbir birim tarafından toplanmış bulunan veri isteklerinin istatistikleri ile ağın karakteristiğini temel alır. Ancak GridNet ile yüksek sayıda veri transferi isteklerinin simülasyonunun yapılması mümkün değildir.

SimGrid: SimGrid [83, 84], TCP trafik modelini içeren iyi bir ağ yapısı modeline sahiptir. Ayrıca, arka plandaki trafiği Ağ Durumu Servisi (Network Weather Service - NWS) kullanarak modeller. Bu sayede, iki makine arasındaki kullanılabilir bant genişliğinin daha gerçekçi bir biçimde modellenmesine imkân tanımaktadır. SimGrid, veri transferlerini paket tabanlı değil akış tabanlı olarak gerçekleştirir.

SimGrid'te kaynaklar, sunucu veya link olabilirler. Sunucular, işlemci hızları ve kullanılabilirlik (availability) değerleri ile, linkler ise gecikme ve bant genişliği değerleri ile modellenirler. Linklerdeki bant genişliklerinin ve sunuculardaki işlem bant genişliklerinin paylaşımı için Max-Min stratejisini kullanır.

GridSim, ns, OPNET, OMNET gibi bazı simülatör programları, TCP/IP ağlarında olduğu gibi paket bazlı veri transferlerini desteklemektedirler. Bu simülatörler, simülasyonlarında oldukça gerçeğe yakın sonuçlar vermektedirler. Ancak, oldukça da yavaş çalışmaktadırlar. Optorsim gibi bazı simülatör programları ise veri transferlerini akışlar olarak modellemektedirler. Bu simülatörler, paket bazlı olanlara göre çok hızlı çalışırlar, fakat sonuçların güvenilirliği de o kadar düşüktür. SimGrid ise, her iki tarafın iyi özeliklerini birleştirme çabalarının sonucudur.

MicroGrid: MicroGrid [85, 86], farklı transfer ve yönlendirme protokolleri gibi karışık ağ yapılarını modelleyebilir. Ağ modellemesi açısından oldukça kapsamlı bir simülatördür. Ancak MicroGrid gerçekte bir simülatör değil, denemeler sırasında fiziksel uygulama kodları çalıştıran bir emülatördür. Globus [3] programından gelen çağrıları alır ve homojen bir işlemci kümesi üzerinde çalıştırır. Bu sebeple çok sayıda deney ve bu deneylerin tekrar yapılması mümkün değildir. Ancak MicroGrid simülatörünün sonuçları, kimlik denetimi ve benzeri destek işlemleri için gerekli süreleri de dâhil ettiği için, diğer simülatörlerin sonuçlarından çok daha gerçekçidir.

Bricks: Bricks [87, 88], işlem ağırlıklı uygulamaların çalıştırıldığı Grid sistemlerinde çizelgeleme algoritmalarının performanslarının ölçülmesi için

hazırlanmıştır. Çok kullanıcılı ve çok servis sunuculu bir ortam varsayar. Kullanıcılar ile servis sunucuları arasındaki ağ yapısının, kullanıcısı tarafından tanımlanmasına olanak tanımaz. Bricks Grid simülatörünün, Veri Grid sistemleri için özel olarak yazılmış versiyonu da bulunmaktadır [88]. Bu versiyonda, depolama ünitelerinde bulunan veri girişi/çıkışı destek işlemleri ve sabit diskin kontrol mekanizmaları da simülatöre dâhil edilmiştir. Ancak ağ yapısı oluşturmadaki eksiklik devam etmektedir.

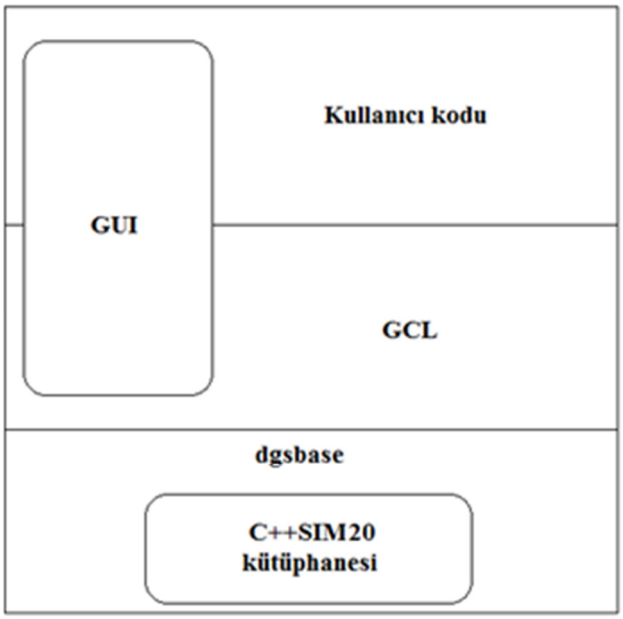
ChiSim: ChiSim simülatörü [49], iş çizelgeleme ve veri kopyalama algoritmalarının simülasyonu için yazılmıştır. Bu simülatör, farklı ağ topolojilerini desteklemez. Tüm kaynakların Grid sistemine sınırlı bant genişliğine sahip bir link aracılığı ile bağlandığını varsayar. Herhangi iki site arasındaki iletişimler için sabit bir bant genişliği değeri kullanır.

GangSim: GangSim [89], Grid ortamlarındaki çizelgeleme stratejilerini test etmek için yazılmıştır. Sitelerin birleşip sanal organizasyonlar kurabilmesine olanak tanır. Sanal organizasyonlardaki iş atama stratejilerinin ve sitelerdeki kullanım politikalarının performans üzerine etkisini araştırır. Sanal organizasyonların da kullanıcıları ve planlayıcıları bulunur. Kaynak kullanım politikaları da site ve sanal organizasyon bazında modellenebilir.

Sugato Bagchi: Sugato Bagchi simülatörü [90], Grid işlem yükünün analizinde kullanılır. Kaynaklar, kaynak üzerine düşen yüklerin dengelenmesi, gelen işlerin mümkün olan en erken sürede tamamlanması, kaynakların sırayla kullanılması, tercihli kaynakların daha fazla kullanılması gibi farklı stratejiler uygulanarak çizelgelenebilir. Ayrıca, simülatörün grafik arayüzü de bulunmaktadır.

3.2. Sistem Mimarisi

DGridSim Şekil 3.2'de gösterilen katmanlı yapı göz önünde bulundurularak tasarlanmıştır. Bu sayede, örneğin, en alt katmanda bulunan simülasyon motoru değiştirildiğinde üst katmanların en az şekilde etkilenmesi ön görülmüştür.



Şekil 3.2. DGridSim sistem mimarisi

C++ dilinde yazılan DGridSim simülatörünün kalbinde C++SIM20 kesikli olay simülatör kütüphanesi [91] yer almaktadır. C++SIM20 kütüphanesi, süreç güdümlü kesikli olay simülasyonu yapan programların gerçeklenmesine olanak sağlar. Kütüphaneyi kullanarak yazılan programlarda sistem, birbirleri ile etkileşim içerisinde olan C++SIM20 süreçlerinden (*process*) oluşur. Süreçler birbirleri ile özel servisler (*facility*) ve tanımlanmış olaylar (*event*) gibi C++SIM20 yapılarını kullanarak haberleşirler.

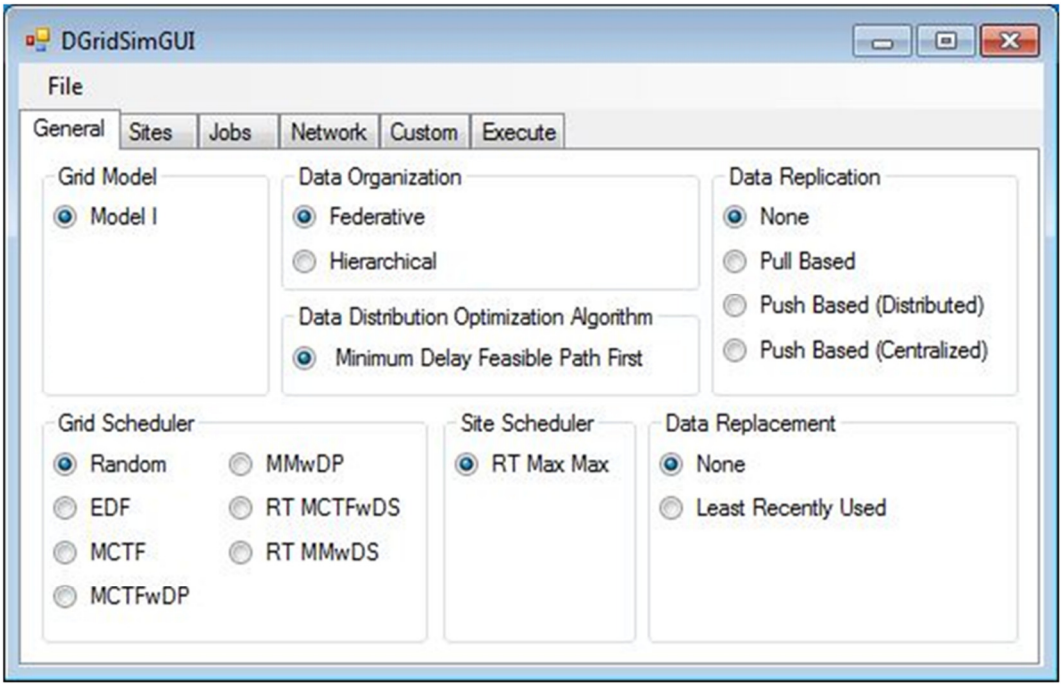
DGridSim yazılımında, C++SIM20 kütüphanesini üst katmanların daha rahat kullanmaları için, *dgsbase* kütüphanesi tasarlanmıştır. Üst katmandaki fonksiyonlar, *dgsbase* kütüphanesinde bulunan nesne tanımlamalarını ve fonksiyonları kullanırlar.

İşlem, veri depolama ve ağ elemanları ile bu elemanları kontrol eden yönetici servislerin arayüzleri, bu katmanda bulunmaktadır. Rezervasyonların tutulduğu veritabanına erişim sağlayan kaynak kodları da bu katmanda bulunmaktadır. Ayrıca, Veri Grid sisteminin simülasyonunda kullanılacak olan Grid, Site ve Kullanıcı gibi nesnelerin arayüzlerinin tanımlandığı yer de *dgsbase* kütüphanesidir. Bu kütüphane doğrudan alt seviye simülasyon kodu geliştirmek isteyen kişiler tarafından da kullanılabilir.

GCL (Grid Component Library – Grid Bileşen Kütüphanesi) bölümünde, *dgsbase* kütüphanesinde arayüzleri tanımlanmış olan bileşenlerin kaynak kodları bulunmaktadır. Bu bölümde, literatürde kabul görmüş var olan standartlar ve referans modeller göz önünde bulundurularak hazırlanan modeller kullanılmıştır. Henüz standartlaştırılamamış bileşenler için, DGridSim kendi modellerini önerip kullanmaktadır ve kullanıcıların da kendi modellerini geliştirmelerine olanak sunacak şekilde tasarlanmıştır.

Kullanıcı kodu seviyesi iş çizelgeleme, veri çizelgeleme ve veri kopyalama algoritmalarının gerçeklendiği bölümdür. *GCL* kullanılarak gerçeklenen modeller, simülasyonu yapılan Veri Grid'ini oluşturan servisler, kaynaklar ve yardımcı sınıflardan oluşmaktadır. Bu katmandaki fonksiyonlar değiştirilerek, DGridSim, farklı modelleri kullanacak şekilde genişletilebilir. Örneğin, bu çalışmada anlatılan hiyerarşik iş çizelgeleme modeline, merkezi iş çizelgeleme modeli ilave edilmek gerektiğinde, Kullanıcı kodu seviyesine yeni iş çizelgeleme modülleri eklemek yeterli olmaktadır. *GCL* veya *dgsbase* katmanında değişlikliğe gerek duyulmamaktadır.

Grafik Kullanıcı Arayüzü (Graphical User Interface – *GUI*) katmanı, simülatörün genel özelliklerinin değiştirilebileceği, simülasyon parametrelerinin belirlenebileceği bir grafik ara birimidir. Şekil 3.3'te, DGridSim simülatörünün sağladığı *GUI*'den örnek bir ekran alıntısı gösterilmiştir. DGridSim *GUI*'si ile hem Veri Grid sistemini tanımlayan modeller ve algoritmalar, hem de Veri Grid sisteminde bulunan işlem elemanı sayısı, link sayısı gibi Bölüm 3.5'te anlatılan parametreler değiştirilebilmektedir.

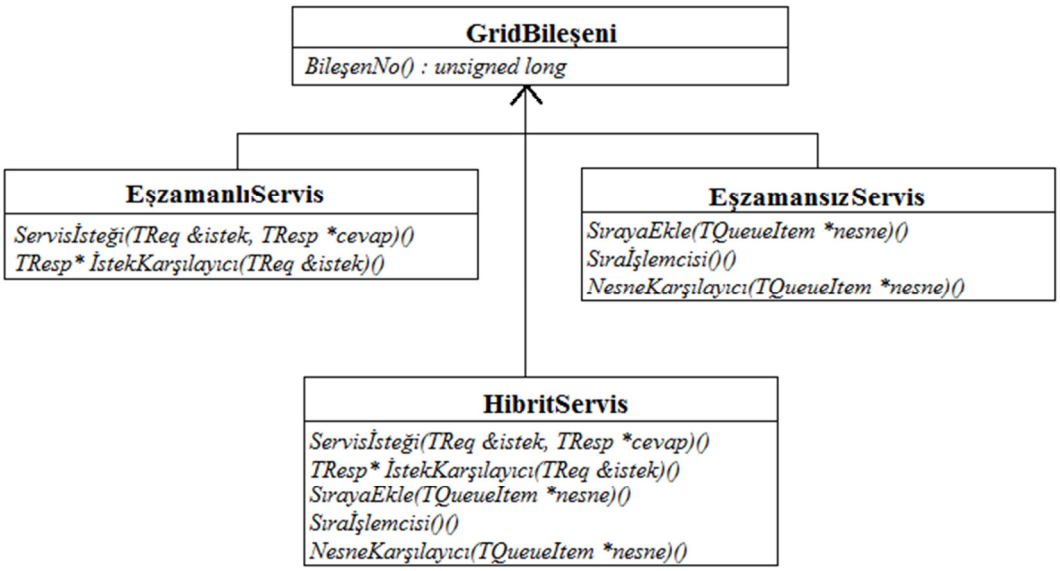


Şekil 3.3. DGridSim GUI örnek ekran alıntısı

3.3. Servis Mimarisi

Veri Grid yapılarındaki bileşenler iki ana başlık altında toplanabilir, *Kaynak Bileşenleri* ve *Servis Bileşenleri*.

Kaynak bileşenleri doğrudan yalın programlama nesneleri gibi görülebilirler. Belirli özelliklere sahiptirler, fakat her hangi bir iş mantıkları yoktur. Örneğin, veri depolama elemanının, disk alanı ve okuma/yazma hızı gibi özellikleri bulunur, ancak her hangi bir fonksiyonu mevcut değildir. Pasif bir eleman olduğu için üzerinde gerçekleştirilmesi gereken işlemler bir veri depolama yöneticisi aracılığıyla sağlanır. Bu veri depolama yöneticisi ise Grid mimarisinde bir servis bileşenidir.



Şekil 3.4. Temel servis bileşenleri

Servis bileşenleri, kaynak bileşenlerinin aksine aktif elemanlardır. Şekil 3.4'te gösterildiği gibi eşzamansız, eşzamanlı ve hibrit servisler olmak üzere üç kısma ayrılabilirler.

Eşzamansız servisler çoğunlukla çizelgeleme servisleridir. İş çizelgeleme ya da veri çizelgeleme servisleri bir kuyruk üzerinde bulunan Grid işlerini ya da veri transferi isteklerini bir çizelgeleme algoritması kullanarak çizelgelerler. Eşzamanlı olanlar ise, doğrudan fonksiyon çağrılarıyla hizmet verebilirler ve genellikle bilgi servisleridir. Çizelgeleme servisleri karar aşamasında bilgi servislerini kullanabilirler. Eğer çizelgeleme servisleri, bir kuyruk yönetimi olmadan çevrim içi çalışırlar ise, bu durumda eşzamanlı bir servis haline gelirler.

Veri kopyalama servisleri, veri transferi isteği geldiğinde veri kopyalama algoritmasını çevrim içi çalıştıran çekme tabanlı bir yaklaşım kullanıyor ise, eşzamanlı servis olurlar. Veri kopyalama servisleri, itme tabanlı merkezi yaklaşımda olduğu gibi, belirli periyotlar ile veri kopyalama algoritmalarının çalıştırılıp verilerin kopyalanmasına karar verildiği yaklaşımda ise, eşzamansız servis olarak nitelendirilirler.

Hibrit servisler, aynı anda hem bir sırayı kontrol eder hem de basit fonksiyon çağrılarına cevap verirler. Rezervasyon servisleri, hem bir sırada

bulunan rezervasyon isteklerini karşılarken, hem de üst katmanlara rezervasyon oluştur/kaldır gibi fonksiyonlar sunan hibrit servislerdir.

Burada ihtiyaç duyulan eşzamanlı/eşzamansız ve hibrit yapıları destekleyebilecek olan servis tasarımları Şekil 3.4'te verilmiştir. Tek bir arayüz yerine üç farklı arayüz tanımlanmasının sebebi kod basitliğini/anlaşılabilirliğini sağlamak ve daha önemlisi şu anda öngörülemeyen yeni bir mimariye ihtiyaç duyulduğunda var olan gerçeklemeleri bozmadan gerekli değişikliklerin yapılabilmesine olanak sağlamaktır.

3.4. Algoritmalar

DGridSim, Veri Grid sistemlerinin performansına etki eden en temel üç bileşen olan iş çizelgeleme, veri dağıtımı ve veri kopyalama bileşenlerinin ortak tek bir çatı altında simülasyonunun yapılabilmesi hedef alınarak geliştirilmiştir. DGridSim hiyerarşik iş çizelgeleme, hiyerarşik veri dağıtımı ve çekme/dağıtık-itme/merkezi-itme veri kopyalama şeklinde modellenen Veri Grid sistemlerinin simülasyonunu desteklemektedir.

DGridSim'de Grid Çizelgeleme Servisi (Grid Scheduling Service - GSS) için, Bölüm 2.3.1'de anlatılan, beş farklı algoritma gerçeklenmiştir: Rand , EDF, MCTF, MCTFwDP ve MMwDP. DGridSim'de Site Çizelgeleme Servisi (Site Scheduling Service - SSS) için RT MaxMax algoritması gerçeklenmiştir.

Veri Grid sisteminde veri çizelgelemesi hiyerarşik modele uygun bir sekilde gerçekleştirilmektedir. Bu bölümde, Veri Yönetici Servisi (Data Management Service - DMS) bileşeni olarak Bölüm 2.4.2'de anlatılan SP_MinHop veri dağıtımı algoritması kullanılmıştır. İlerleyen bölümlerde, gerçek-zamanlı veri dağıtımı problemi üzerinde yapılan diğer çalışmaların sonuçları da gösterilecektir.

DGridSim, federatif ve hiyerarşik veri organizasyonu modellerini desteklemektedir. Federatif veri organizasyonu modelinde, Veri Grid sistemindeki tüm sitelerde veri depolama elemanı bulunmaktadır. Verilerin birincil kopyaları, rassal biçimde sitelerdeki veri depolama elemanlarında yer alırlar ve asla silinmezler. Veriler, veri kopyalama servislerinin kontrolünde, farklı sitelerde kopyalanırlar. Veri kopyalama servisi tarafından kopyalanacak olan verileri depolayacak yeterli veri depolama alanı bulunmadığı durumda, veri depolama

elemanında bulunan bazı veriler, veri yenileme algoritmalarının kontrolünde, silinebilirler.

Hiyerarşik modelde ise Veri Grid sistemindeki siteler üç farklı hiyerarşi seviyesinde gruplandırılmışlardır. Hiyerarşi-0 seviyesinde sadece veri depolama elemanlarına sahip tek bir site vardır ve başlangıçta tüm verilerin birincil kopyalarının bu sitede bulunduğu kabul edilir. Federal modelde olduğu gibi, birincil kopyalar Hiyerarşi-0 veri depolama elemanlarından silinmezler. Hiyerarşi-0 sitenin altında yine sadece veri depolama elemanlarına sahip birden fazla Hiyerarşi-1 site bulunur. Her bir Hiyerarşi-1 sitenin altında ise bir veya birden fazla veri depolama elemanına sahip Hiyerarşi-2 site(ler) bulunur. Veriler, Hiyerarşi-0 siteden Hiyerarşi-1 sitelere ve Hiyerarşi-1 sitelerden Hiyerarşi-2 sitelere sırasıyla kopyalanır. Verilerin birincil kopyaları haricindeki tüm veriler, veri yenileme algoritmasının kontrolünde, Hiyerarşi-1 ve 2 sitelerdeki veri depolama elemanlarından silinebilirler.

DGridSim simülatörünü literatürdeki diğer simülatörlerden ayıran en önemli özelliklerinden birisi de, DGridSim'in literatürde bugüne kadar önerilmiş olan tüm veri kopyalama yaklaşımlarını, hem federal hem de hiyerarşik veri organizasyonu modelleri için destekliyor olmasıdır. DGridSim, literatürde bulunan çekme tabanlı, itme tabanlı dağıtık ve itme tabanlı merkezi tüm veri kopyalama stratejilerini desteklemektedir. Literatürde kabul görmüş ve yaygın olarak kullanılan simülatörler ise genellikle tek bir veri kopyalama yaklaşımını ve sadece belirli bir veri organizasyon modelini desteklemektedirler.

DGridSim'de iki veri yenileme modeli seçeneği sunulmaktadır: 1) *None*: Depolama elemanlarına kopyalanan veriler yenilenmezler, 2) LRU: Eğer veri depolama alanında yeni gelen veriyi alacak kadar yer yoksa, depolama alanındaki verilerden en uzun zamandır kullanılmayanları silinirler.

3.5. Simülasyon Parametreleri

DGridSim simülasyon parametreleri dört başlık altında toplanabilir: Siteler (Sites), Ağ (Network), İşler (Jobs) ve Özel (Custom). Çizelge 3.1'de gösterilen siteler ile ilgili parametreler, simülasyonu yapılacak olan Veri Grid sisteminin temel özelliklerini belirleyen parametreleri içerir. Çizelge 3.2'de gösterilen ağ ile

ilgili parametreler, Veri Grid sisteminin ağ altyapısının özelliklerini belirleyen parametreleri içerir. Çizelge 3.3'de gösterilen işler ile ilgili parametreler, Veri Grid sistemine sunulacak olan iş yükünü karakterize etmek için kullanılan parametreleri içerir. Çizelge 3.4'te gösterilen özel parametreler, Veri Grid sisteminin simülasyonu için gerekli olan diğer tüm parametreleri içerir.

Çizelge 3.1. Siteler ile ilgili parametreler

Parametre	**Açıklama**
Site Count	Veri Grid sistemindeki site sayısı
Min Storage Element	Her bir sitedeki veri depolama elemanlarının asgari sayısı
Max Storage Element	Her bir sitedeki veri depolama elemanlarının azami sayısı
Min SE Capacity	Her bir veri depolama elemanın asgari kapasitesi (MBayt)
Max SE Capacity	Her bir veri depolama elemanın azami kapasitesi (MBayt)
Min Computing Element	Her bir sitedeki işlem elemanlarının asgari sayısı
Max Computing Element	Her bir sitedeki işlem elemanlarının azami sayısı
Min CE Capacity	Her bir işlem elemanının asgari işlem kapasitesi (MIPS)
Max CE Capacity	Her bir işlem elemanının azami işlem kapasitesi (MIPS)
Tier 1 Site Count	Hiyerarşik veri organizasyonunda, 1 tane Hiyerarşi 0 site, *Tier 1 Site Count* kadar Hiyerarşi 1 site ve (*Site Count* – *Tier 1 Site Count* - 1) kadar Hiyerarşi 2 site bulunur.
Tier SE Capacity Ratio	Hiyerarşik veri organizasyonunda, Hiyerarşi 2 sitelerin toplam veri depolama alanının Hiyerarşi 1 sitelerin toplam veri depolama alanına oranını belirler

Çizelge 3.2. Ağ ile ilgili parametreler

Parametre	**Açıklama**
Min Routers	Siteleri birbirine bağlayan internet ağındaki yönlendirici sayısının asgari değeri
Max Routers	Siteleri birbirine bağlayan internet ağındaki yönlendirici sayısının azami değeri
Min Internet Links	Internet ağındaki linklerin sayısının asgari degeri
Max Internet Links	Internet ağındaki linklerin sayısının azami degeri
Min Internet Link Bandwidth	Internet ağındaki linklerin bant genişliklerinin asgari değeri (MBayt/sn)
Max Internet Link Bandwidth	Internet ağındaki linklerin bant genişliklerinin azami değeri (MBayt/sn)
Min Internet Link Delay	Internet ağındaki linklerdeki gecikmenin asgari değeri (sn)
Max Internet Link Delay	Internet ağındaki linklerdeki gecikmenin azami değeri (sn)
Min Edge Link Bandwidth	Internet kenar yönlendiricisi ve site ağ geçidi arasındaki linklerin bant genişliklerinin asgari değeri (MBayt/sn)
Max Edge Link Bandwidth	Internet kenar yönlendiricisi ve site ağ geçidi arasındaki linklerin bant genisliklerinin azami değeri (MBayt/sn)
Min CE Link Bandwidth	İşlem elemanları ve site ağ geçitleri arasındaki linklerin bant genişliklerinin asgari değeri (MBayt/sn)
Max CE Link Bandwidth	İşlem elemanları ve site ağ geçitleri arasındaki linklerin bant genişliklerinin azami değeri (MBayt/sn)
Min SE Link Bandwidth	Veri depolama elemanları ve site ağ geçitleri arasındaki linklerin bant genişliklerinin asgari değeri (MBayt/sn)
Max SE Link Bandwidth	Veri depolama elemanları ve site ağ geçitleri arasındaki linklerin bant genişliklerinin azami değeri (MBayt/sn)
Min Site Link Delay	Site içi linklerdeki gecikmenin asgari değeri (sn)
Max Site Link Delay	Site içilinklerdeki gecikmenin azami değeri (sn)

Çizelge 3.3. İşler ile ilgili parametreler

Parametre	**Açıklama**
Job Count	Sisteme sunulacak olan iş sayısı
Inter-arrival time	İki ardışık işin gelişleri arasında geçen ortalama zaman olup, işler sisteme Poisson sürecine uygun olarak gelir
Min Job Size	Asgari iş büyüklüğü
Max Job Size	Azami iş büyüklüğü
Min Deadline	İşlerle ilişkilendirilebilecek asgari son zaman
Max Deadline	İşlerle ilişkilendirilebilecek azami son zaman
Min Job Data	Her bir işle ilişkilendirilebilecek asgari veri elemanı sayısı
Max Job Data	Her bir işle ilişkilendirilebilecek azami veri elemanı sayısı
Data-item Count	Sistemde bulunan birbirinden farklı veri elemanı sayısı
Min DI Size	Her bir veri elemanın asgari büyüklüğü (MBayt)
Max DI Size	Her bir veri elemanın azami büyüklüğü (MBayt)

Çizelge 3.4. Özel parametreler

Parametre	**Açıklama**
is-dqp	Bilgi servisi dinamik sorgulama periyodu (sn)
file-replication-threshold	Federal veri kopyalama algoritmaları tarafından kullanılan veri kopyalama eşik değeridir
file_replication_threshold_tier2	Dağıtık çekme tabanlı veri kopyalama algoritmalarında Tier 2 sitelere için kullanılan veri kopyalama eşik değeridir
file_replication_threshold_tier1	Hiyerarşik veri kopyalama algoritmalarında Tier 1 siteler için kullanılan veri kopyalama eşik değeridir
file_replication_threshold_tier0	Hiyerarşik veri kopyalama algoritmalarında Tier 0 siteler için kullanılan veri kopyalama eşik değeridir
grid-dms-period	Grid Veri Yönetimi Servisi çalışma periyodu (sn)
drs-period	Veri kopyalama algoritmaları için çalışma periyodu (sn)
gss-period	Çevrim dışı Grid seviyesi iş çizelgeleme algoritmaları için çalışma periyodu (sn)

3.6. Simülasyon Sonuçları

Bölüm 2'de önerilen gerçek-zamanlı Veri Grid sistemi modelini baz alarak gerçeklenen DGridSim simülatöründe Rand, EDF, MCTF, MCTFwDP ve MMwDP iş çizelgeleme algoritmalarının performans ölçümleri aşağıda anlatıldığı şekilde gerçekleştirilmiştir.

Simülasyonun başlaması ile yeni Veri Grid sistemi oluşturulur. Sistem, sıfır veya daha fazla işlem elemanı ile bir adet veri depolama elemanı bulunan sitelerden oluşmaktadır. Her sitede işlem ve depolama elemanlarının doğrudan bağlı olduğu bir adet geçit yönlendiricisi bulunmaktadır. İşlem elemanları ile geçit yönlendiricisi U~[400, 600] Mbayt/sn ve depolama elemanı ile geçit yönlendiricisi U~[800, 1200] Mbayt/sn bant genişliğine sahip linkler ile bağlanmıştır. U~[A, B], A ile B arasında düzgün olarak yayılmış anlamına gelmektedir. Geçit yönlendiricileri ise rassal olarak yerleştirilmiş U~[8, 12] çekirdek yönlendirici ve U~[120, 180] Mbayt/sn bant genişliğine sahip U~[16, 24] link ile birbirlerine bağlanmışlardır.

Kullanılan veri organizasyon modeline göre, sistem federatif ve hiyerarşik veri organizasyon modeline uyan sistemler olarak ikiye ayrılır. Bu bölümde belirtilen performans ölçümleri sırasında, simülasyonlar her iki veri organizasyon modeli için tekrar edilmiştir. Federatif ve hiyerarşik veri organizasyonuna uyan sistemler, aşağıdaki şekilde oluşturulur:

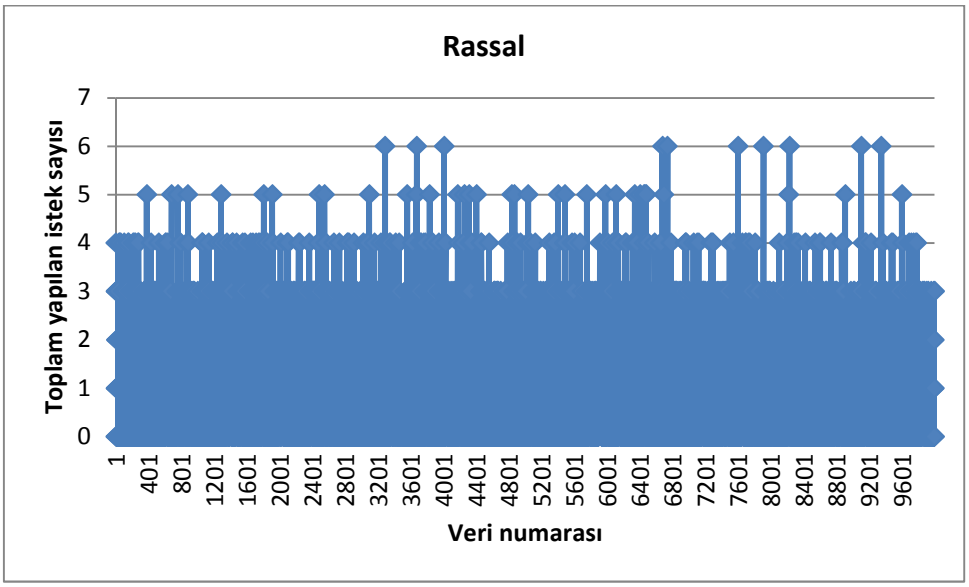
Federatif veri organizasyon modeline uyan sistemler: Sistem üzerinde her biri U~[24, 36] heterojen işlem elemanı ve bir adet depolama elemanı bulunduran 20 adet site bulunur. İşlem elemanları U~[8000, 12000] MIPS değerine sahiptirler. Depolama elemanları ise U~[80000, 120000] Mbayt kapasitelidirler. Sistem üzerinde bulunan U~[800, 1200] Mbayt büyüklüğündeki 10000 farklı veri, farklı sitelere rassal bir biçimde dağıtılmışlardır.

Hiyerarşik veri organizasyon modeline uyan sistemler: Sistem üzerinde biri Hiyerarşi-0, dördü Hiyerarşi-1 ve yirmisi Hiyerarşi-2 olmak üzere 25 adet site bulunmaktadır. Hiyerarşi-1 sitelerde bulunan depolama elemanları U~[200000, 300000] Mbayt, Hiyerarşi-2 sitelerde bulunan depolama elemanları ise U~[40000, 60000] Mbayt kapasitelidir. Simülasyon başladığında, sistem üzerinde bulunan

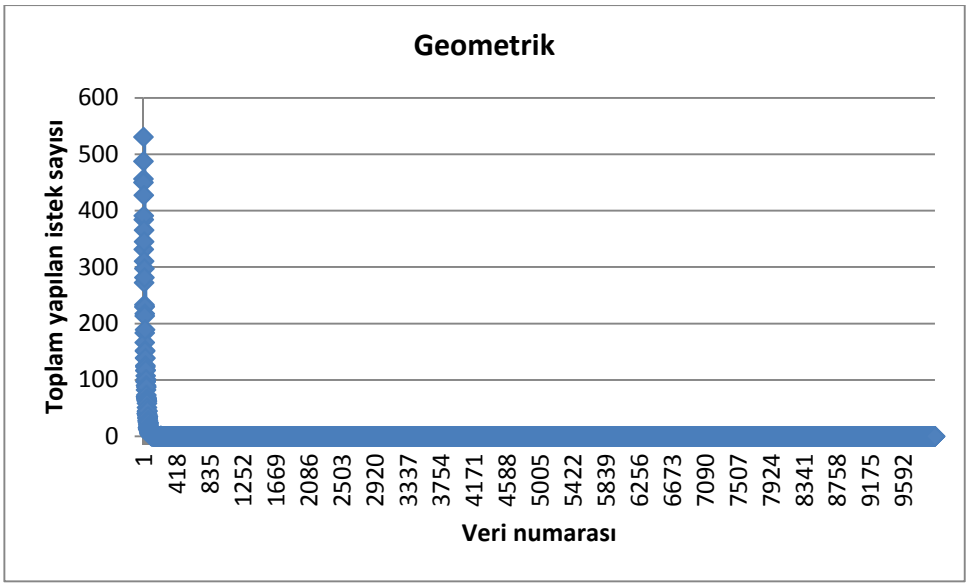
10000 farklı verinin tamamı, sadece depolama elemanı barındıran Hiyerarşi-0 sitede yer alır. İşlem kapasitesine sahip olan Hiyerarşi-2 sitelere, veriler, Hiyerarşi-0 sitesinden dağıtılır.

Veri Grid sisteminin oluşturulmasından sonra, sisteme iş arzı başlar. İşler, işlem büyüklüğü, son zaman ve istekte bulunulacak olan veri sayısı ile tanımlanır. Temel çalışmalarda U~[4800000, 7200000] MI (Million Instruction – Milyon Komut) büyüklüğünde, U~[400, 600] sn son zamana sahip ve U~[2, 4] veri isteğinde bulunan 10500 iş oluşturulmuştur. Bu isteklerden ilk 500 tanesi ile ilgili sonuçlar, yavaş başlangıç etkisini sıfırlamak için ihmal edilmektedir. Dolayısı ile simülasyonlar 10500 iş için yapılmakta olup sadece son 10000 iş için elde edilen sonuçlar bu bölümde rapor edilmiştir.

Yukarıdaki simülasyon parametreleri ile simülasyonlar yapılmıştır. Simülasyonlar sırasında, Bölüm 2'de anlatılan beş iş çizelgeleme algoritması test edilmiştir (Rand, EDF, MCTF, MCTFwDP, MMwDP). Çalışmalarda rassal, geometrik ve zipf olmak üzere üç adet veri erişim düzeni kullanılmıştır. 10000 gerçek-zamanlı işin 10000 veri arasından ortalama üç veri isteğinde bulunduğu durumda oluşan, her bir veri için oluşan toplam istek sayıları Şekil 3.5 – 3.7'de gösterilmiştir.

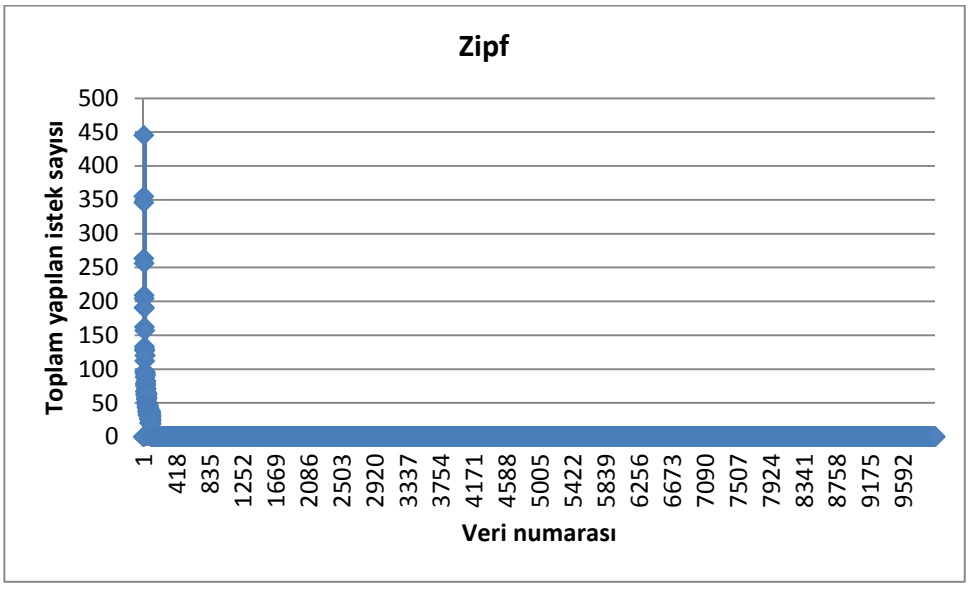


Şekil 3.5. Rassal veri erişim düzeni



Şekil 3.6. Geometrik veri erişim düzeni

Şekil 3.5'te gösterildiği üzere, rassal veri erişim düzeninde, verilere yapılan toplam istek sayıları arasında fazla bir değişiklik olmamaktadır. Şekil 3.6'da gösterildiği üzere, geometrik veri erişim düzeninde, bazı verilere oldukça fazla sayıda istek yapılırken, verilerin büyük bölümüne hiç istekte bulunulmamaktadır.



Şekil 3.7. Geometrik veri erişim düzeni

Şekil 3.7'de gösterildiği üzere, zipf veri erişim düzeninde, bazı verilere oldukça fazla sayıda istek yapılırken, verilerin büyük bölümüne hiç istekte bulunulmamaktadır. Geometrik veri erişim düzeninden farklı olarak zipf veri erişim düzeninde, hiç istekte bulunulmamış veri sayısı daha azdır.

Baz çalışmalarda veri dağıtımı algoritması olarak SP_MinHop kullanılmıştır. Veri kopyalama algoritması olarak 5. Bölüm'de anlatılacak olan çekme tabanlı MRD algoritması; veri yenileme algoritması olarak LRU algoritması kullanılmıştır. Baz çalışmaların sonuçları Çizelge 3.5'te gösterilmiştir. Çizelgede belirtilen değerler, 20 simülasyon sonucunda 10000 gerçek-zamanlı iş arasından başarılı bir şekilde tamamlanan işlerin ortalama yüzdesini göstermektedir. Örneğin; 20 simülasyon tekrarında 10000 gerçek-zamanlı işten ortalama olarak 5000 tanesi son zamanından önce tamamlanabilmiş ise, çizelgedeki ilgili hücreye yüzde 50 değeri yazılmıştır.

Çizelge 3.5. Baz çalışma sonuçları

İş çizelgeleme algoritması	**Rassal**		**Geometrik**		**Zipf**	
	Fed.	**Hiye.**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**
Rand	61	56	63	94	59	64
EDF	60	55	63	94	59	64
MCTF	56	52	62	93	56	59
MCTFwDP	58	52	64	94	56	60
MMwDP	59	52	65	96	60	66

Baz çalışmadaki simülasyon sonuçlarına göre, federatif ve hiyerarşik veri organizasyon modellerinde rassal veri erişim düzeni kullanıldığında Rand iş çizelgeleme algoritması en iyi sonuçları vermektedir. Rand algoritmasını, sırasıyla, EDF, MMwDP, MCTFwDP ve MCTF algoritmaları izlemektedir. Rassal veri erişim düzeni için oluşan bu sıralamanın sebebini şu şekilde açıklayabiliriz: Algoritmaların göstermiş oldukları performanslarda veri transferleri önemli rol oynamaktadır. Rand ve EDF algoritmaları işleri rassal bir

şekilde sitelere çizelgeledikleri için, işlerin ihtiyaç duyduğu verilerin transferleri sırasında kullanılan ağ ve veri depolama elemanları bant genişliği kaynakları Grid genelinde düzgün dağılmaktadır. Diğer taraftan, MMwDP, MCTFwDP ve MCTF algoritmaları bazı sitelere daha çok sayıda iş çizelgelemekte ve bu sitelerdeki ilgili kaynaklarda tıkanıklığa neden olarak sistemin gerçek-zaman performansı düşürmektedirler. Ayrıca, hem MMwDP hem de MCTFwDP veri transferlerini azaltabilmek için işleri olabildiğince işin ihtiyaç duyduğu verinin bulunduğu sitelere çizelgelemeye çalışırlar. MMwDP ve MCTFwDP algoritmaları, bu stratejileri sayesinde, MCTF'den daha iyi sonuçlar vermektedirler.

Rassal veri erişim düzeninde, iş için gerekli olan veriler 10000 farklı veri arasından rassal olarak seçildiği için veri kopyalamanın etkisi oldukça sınırlı kalmaktadır. Dolayısıyla, veri transferleri için kullanılan ağ ve veri depolama elemanları bant genişliği kaynaklarının Grid genelinde daha düzgün dağılım gösterdiği federatif veri organizasyon modelinde algoritmalar daha yüksek performans göstermektedirler.

Baz çalışmada, federatif ve hiyerarşik veri organizasyon modellerinde geometrik veri erişim düzeni kullanıldığında MMwDP iş çizelgeleme algoritması en iyi sonuçları vermektedir. MMwDP algoritmasını, sırasıyla, MCTFwDP, Rand, EDF ve MCTF algoritmaları izlemektedir. MMwDP ve MCTFwDP algoritmalarının performanslarıyla öne çıkmalarının sebebini şu şekilde açıklayabiliriz: İşlerle ilişkilendirilen veriler 10000 veri arasından küçük bir gruptan seçilmektedir. Bu iki algoritma, verilerin bulunduğu sitelere öncelik vererek ve verilerin küçük bir gruptan seçilmesi sayesinde, veri transferlerini rassal veri erişim düzenine göre olabildiğince azaltıp daha yüksek performans sunmayı başarmışlardır. Diğer taraftan, Rand ve EDF algoritmaları işleri rassal olarak sitelere çizelgelemenin sağlamış olduğu avantajla MCTF'den daha iyi sonuçlar vermektedirler.

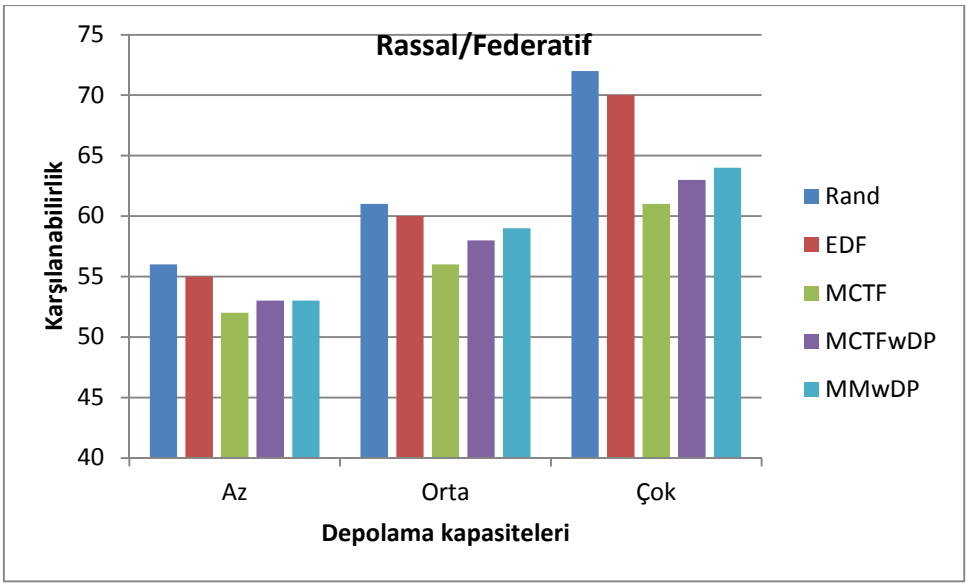
Geometrik veri erişim düzeninde, veri kopyalamanın gerçek-zaman performansına olan olumlu katkısı rahatlıkla görülebilmektedir. Her iki veri organizasyonu modeli içinde, tüm algoritmaların performansları iyileşmiştir. Özellikle hiyerarşik veri organizasyonu modeline uyan sistemlerde, rassal veri erişim düzenine göre %40'lara varan performans artışları gözlemlenmektedir.

Çizelge 3.6. Baz çalışmalardaki ortalama çalışma zamanları

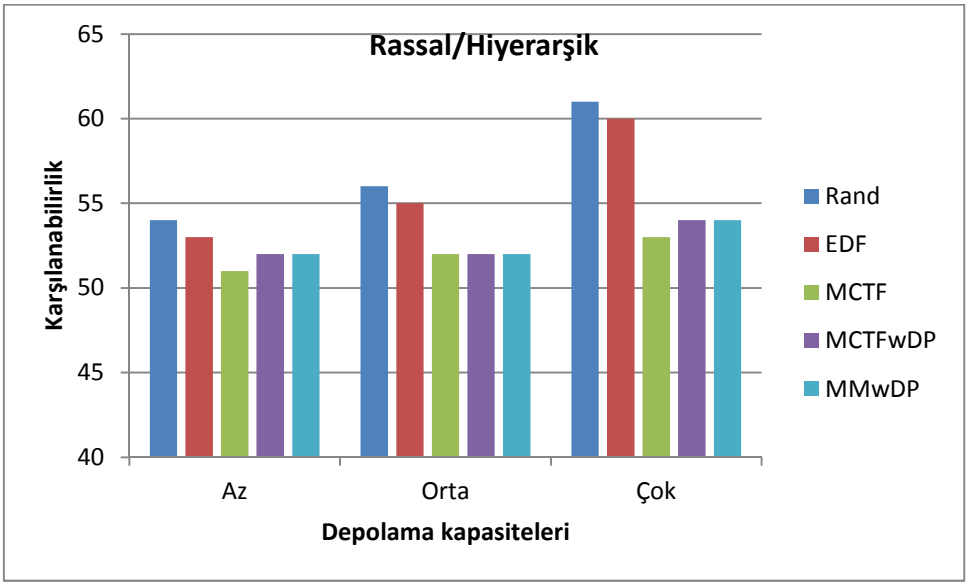
İş çizelgeleme algoritması	Rassal		Geometrik		Zipf	
	Fed.	Hiye.	Fed.	Hiye.	Fed.	Hiye.
Rand	156	684	268	185	188	150
EDF	196	872	278	185	197	152
MCTF	290	912	586	454	517	425
MCTFwDP	298	1574	584	461	558	486
MMwDP	521	3003	523	490	572	452

10000 gerçek-zamanlı işin simülasyonu için gerekli ortalama zamanlar ise Çizelge 3.6'da gösterilmiştir. Çizelgede bulunan her bir değer, bir simülasyonun tamamlanması için gerekli ortalama zamanı saniye cinsinden ifade etmektedir. Elde edilen sonuçlara göre Rand iş çizelgeleme algoritması, diğer algoritmalara göre daha kısa zamanda sonuçlanmaktadır. EDF algoritması da Rand algoritmasına yakın sonuçlar göstermektedir. Geometrik ve zipf veri erişim düzenlerinde hiyerarşik veri organizasyonu modelinde yüksek performans sonuçları elde edilmiştir. Buna bağlı olarak, gerekli olan simülasyon süreleri de rassal veri erişim düzenine göre düşüktür.

Şekil 3.8 – 3.13'te rassal/geometrik/zipf veri erişim düzenleri için federatif ve hiyerarşik veri organizasyon modeline sahip sistemlerde, depolama elemanlarının kapasitelerinin, gerçek zaman performansı üzerindeki etkisi gösterilmiştir. Şekillerde *orta* ile gösterilen depolama kapasitesi yapısında, federatif veri organizasyon modelinde, depolama elemanları U~[80000, 120000] Mbayt kapasitelidir. Hiyerarşik modelde ise, Hiyerarşi-1 sitelerde bulunan depolama elemanları U~[200000, 300000] Mbayt, Hiyerarşi-2 sitelerde bulunan depolama elemanları ise U~[40000, 60000] Mbayt kapasitelidir. Depolama elemanlarının kapasiteleri, *az* ile gösterilen depolama kapasitesi yapısında yarıya düşürülmüş; çok ile gösterilen depolama kapasitesi yapısında ise iki katına çıkarılmıştır. Şekillerde de görüldüğü gibi, depolama elemanlarının kapasitesinin arttırılması ile algoritmaların gerçek zaman performansı artmaktadır.



Şekil 3.8. Rassal veri erişim düzeni ve federatif veri organizasyon modeline sahip sistemde depolama elemanı kapasitelerinin performans üzerine etkisi

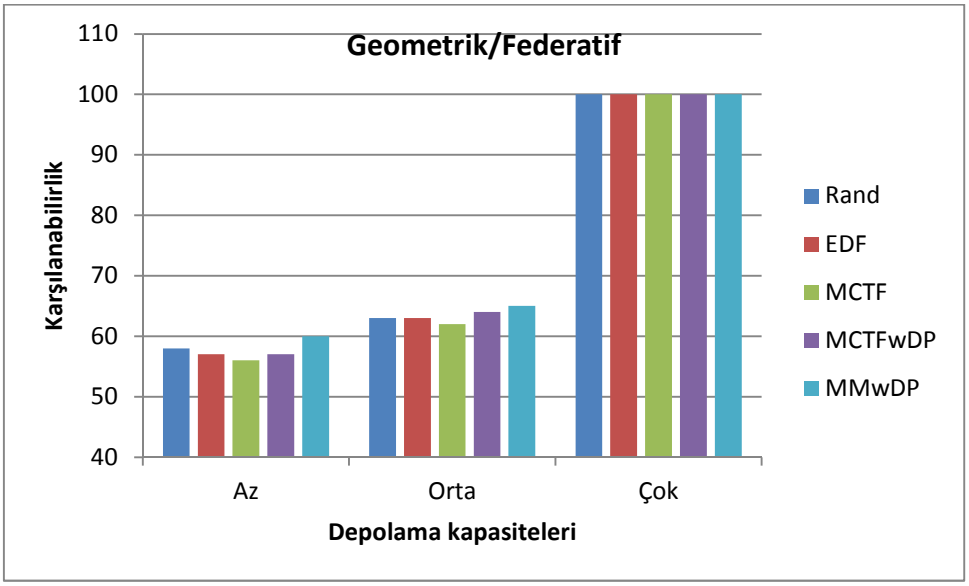


Şekil 3.9. Rassal veri erişim düzeni ve hiyerarşik veri organizasyon modeline sahip sistemde depolama elemanı kapasitelerinin performans üzerine etkisi

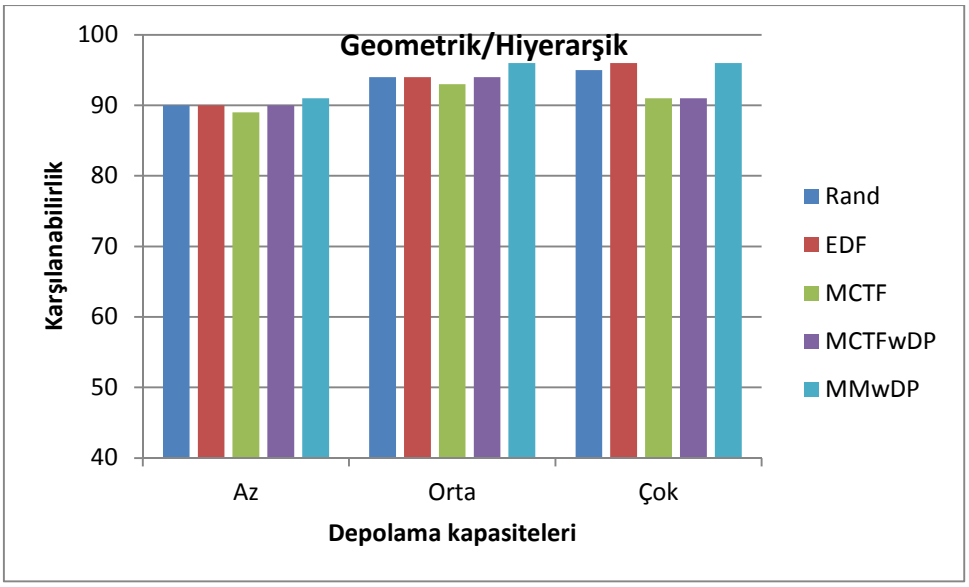
Şekil 3.8 ve 3.9'da gösterilen simülasyon sonuçlarına göre, depolama kapasiteleri arttıkça, sistemin performansı artmaktadır. Depolama kapasitelerinin

arttırılmasının, federatif veri organizasyon modelinde, yüzde 15'lere varan performans artışları ile daha etkili olduğu görülmektedir. Rassal veri erişim düzeninde Rand iş çizelgeleme algoritması en iyi sonuçları vermektedir. Bu algoritmayı EDF, MMwDP ve MCTFwDP algoritmaları takip etmektedir.

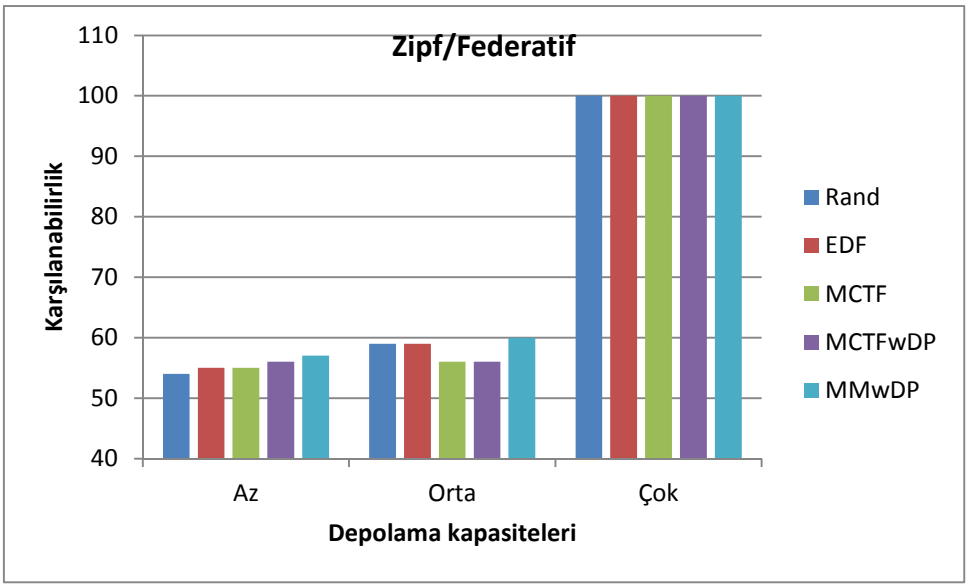
Şekil 3.10 ve 3.11'de ise geometrik veri erişim düzenine sahip sistemler üzerinde, depolama elemanı kapasitelerinin, sistem performansı üzerine etkisi gösterilmiştir. Depolama kapasitelerinin arttırılması yine federatif veri organizasyon modelinde daha etkili olmaktadır. En iyi performans sonuçları MMwDP iş çizelgeleme algoritması ile elde edilmektedir. Depolama kapasitesinin çok olduğu durumda, işlerin tamamına yakını başarı ile tamamlanmıştır. Bu durum, işlerin tamamlanması için gerekli olan verilerin saklanacağı depolama alanının fazla olmasının ve veri kopyalamanın önemlerini teyit etmektedir. Hiyerarşik veri organizasyonu modeline uyan sistemlerde, az miktarda depolama elemanı bile yüksek performans sonuçları için yeterli olmaktadır.



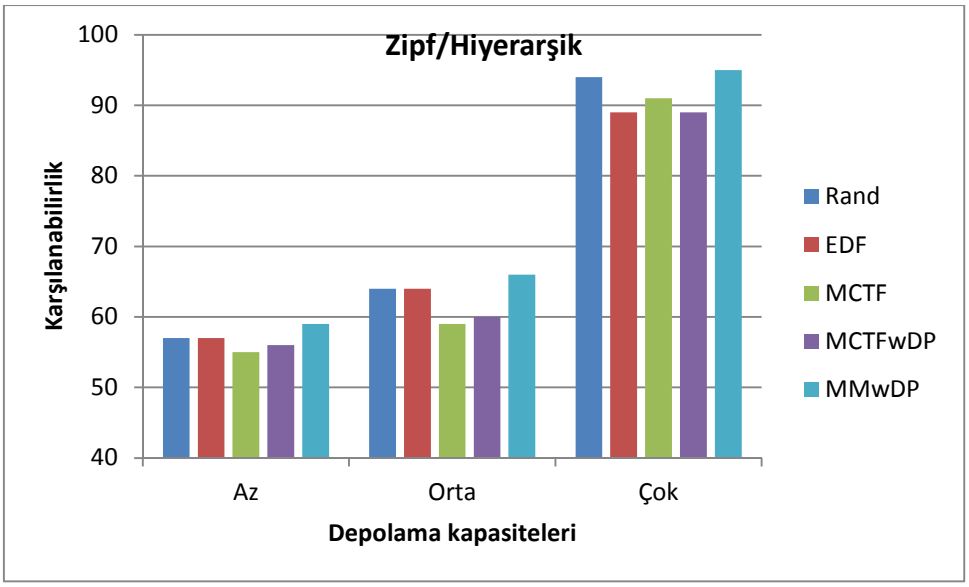
Şekil 3.10. Geometrik veri erişim düzeni ve federatif veri organizasyon modeline sahip sistemde depolama elemanı kapasitelerinin performans üzerine etkisi



Şekil 3.11. Geometrik veri erişim düzeni ve hiyerarşik veri organizasyon modeline sahip sistemde depolama elemanı kapasitelerinin performans üzerine etkisi



Şekil 3.12. Zipf veri erişim düzeni ve federatif veri organizasyon modeline sahip sistemde depolama elemanı kapasitelerinin performans üzerine etkisi



Şekil 3.13. Zipf veri erişim düzeni ve hiyerarşik veri organizasyon modeline sahip sistemde depolama elemanı kapasitelerinin performans üzerine etkisi

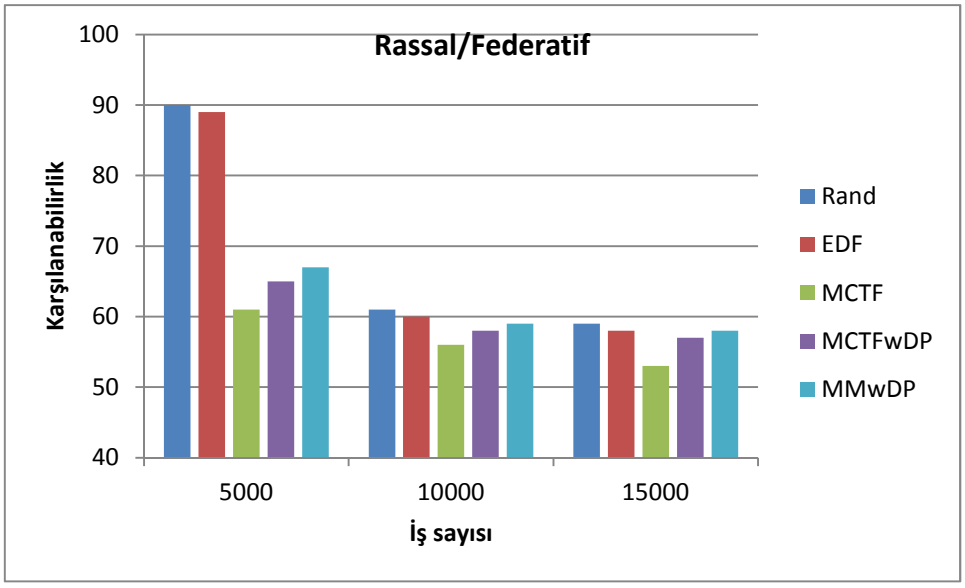
Şekil 3.12 ve 3.13'te zipf veri erişim düzenine sahip sistemler üzerinde, depolama elemanı kapasitelerinin, sistem performansı üzerine etkisi gösterilmiştir. Geometrik veri erişim düzenine sahip sistemler için belirtilen performans ölçümlerine benzer şekilde, zipf veri erişim düzenine sahip sistemler üzerinde de MMwDP iş çizelgeleme algoritması en iyi sonuçları vermektedir. Zipf veri erişim düzenine sahip sistemlerdeki gerçek zaman performans sonuçları, geometrik düzene göre daha düşük kalmaktadır. Zipf veri erişim düzeninde, dağılım daha fazla sayıda erişilen veriyi içermektedir. Bu durumun sonucu olarak da, iş ile ilgili verilerden en az bir tanesinin, site içerisinde kopyalanmamış veri olması olasılığı, zipf veri erişim düzeninde, geometrik veri erişim düzenine göre daha fazladır. Dolayısıyla, zipf veri erişim düzeninde performans sonuçları, geometrik veri erişim düzenine göre daha düşük çıkmaktadır.

Şekil 3.14 – 3.19'da rassal/geometrik/zipf veri erişim düzenleri için federatif ve hiyerarşik veri organizasyon modeline sahip sistemlerde, Veri Grid sistemine arz edilen iş sayısının, gerçek zaman performansı üzerindeki etkisi gösterilmiştir. Şekillerde görüldüğü gibi, iş sayısının arttırılması, gerçek zaman performansını

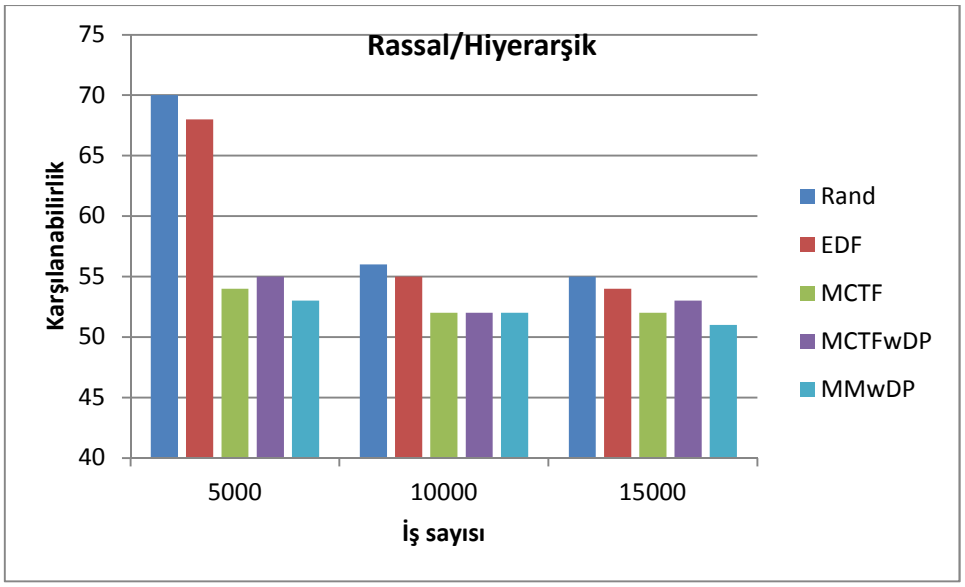
büyük oranlarda düşürmektedir. İş sayısının azalması, ağ elemanları üzerindeki baskıyı azaltmakta ve daha fazla işin başarı ile tamamlanmasını sağlamaktadır.

Şekil 3.14 ve 3.15'de gösterilen simülasyon sonuçlarına göre, rassal veri erişim düzenine sahip sistemlerde, iş sayısının azalması ile özellikle Rand ve EDF iş çizelgeleme algoritmalarının performansı etkilenmektedir. Sisteme arz edilen iş sayısı 5000'e kadar düşürüldüğünde, hiyerarşik veri organizasyonunda %15, federatif veri organizasyonunda ise %30'lara varan performans iyileşmeleri gözlemlenmektedir. Diğer algoritmaların performansları ise sadece %5 civarında etkilenmektedir. MCTF, MCTFwDP ve MMwDP iş çizelgeleme algoritmaları, işleri sürekli belirli sitelere atamaktadır. Bu nedenle, söz konusu algoritmaların performans ölçümlerinin sonuçları, iş sayısı azalsa bile performansta önemli iyileşmelerin görülmediğini göstermektedir.

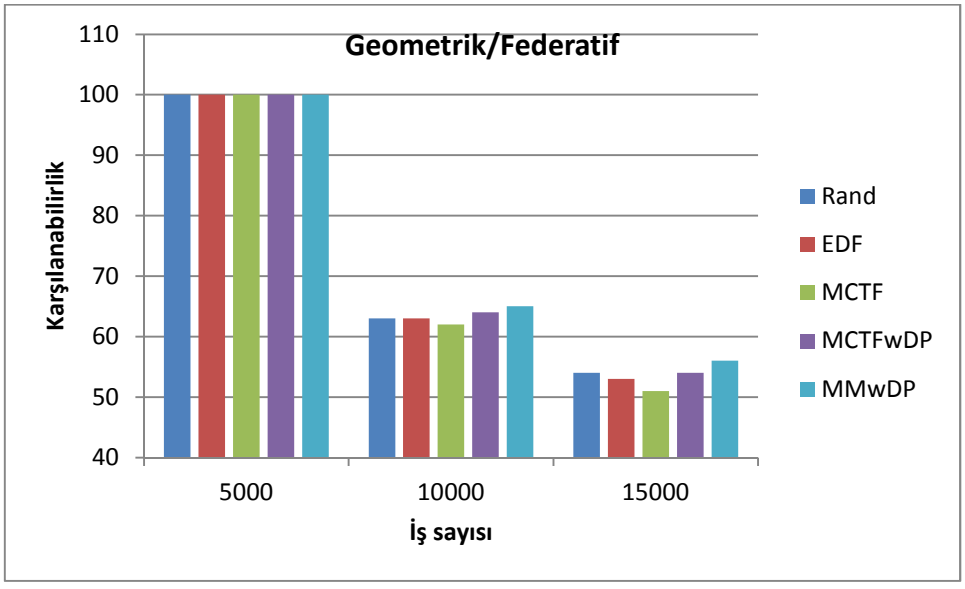
Şekil 3.16 ve 3.17'de gösterilen sonuçlara göre, sisteme arz edilen iş sayısından özellikle federatif veri organizasyon modeline sahip sistemlerin etkilendiği görülmektedir. Geometrik veri erişim düzenine sahip sistemlerde, sisteme arz edilen iş sayısı 5000'e kadar düşürüldüğünde, işlerin tamamına yakını başarı ile tamamlanmaktadır.



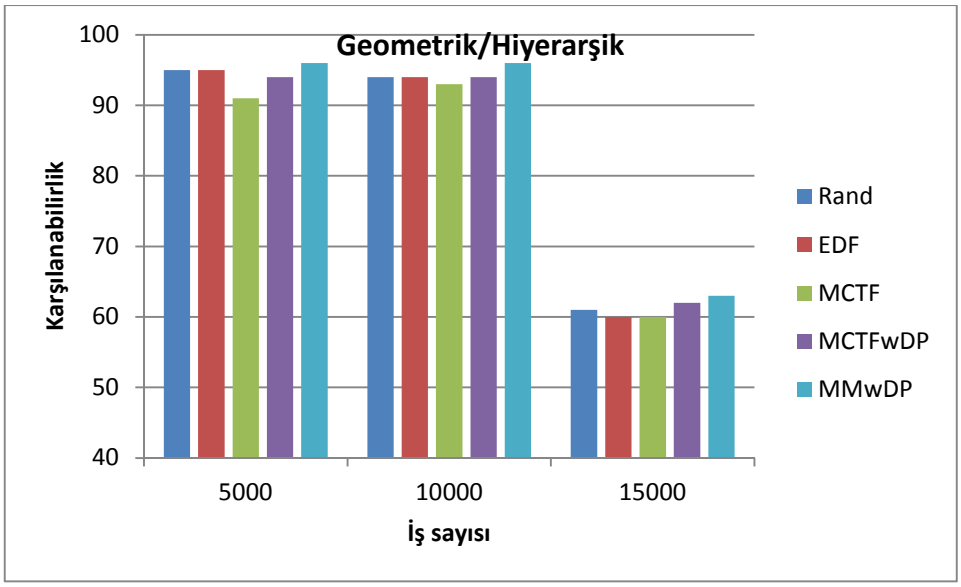
Şekil 3.14. Rassal veri erişim düzeni ve federatif veri organizasyon modeline sahip sistemde arz edilen iş sayısının performans üzerine etkisi



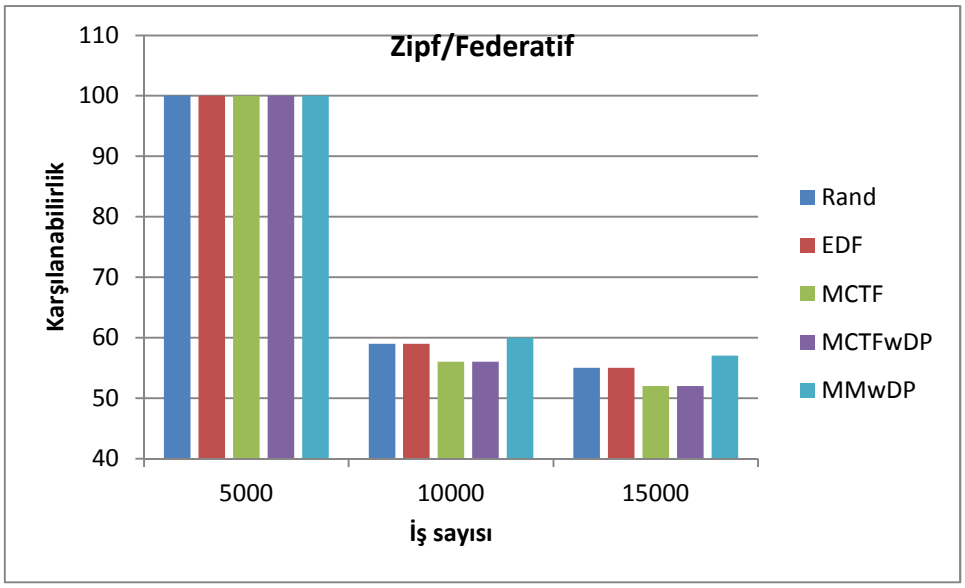
Şekil 3.15. Rassal veri erişim düzeni ve hiyerarşik veri organizasyon modeline sahip sistemde arz edilen iş sayısının performans üzerine etkisi



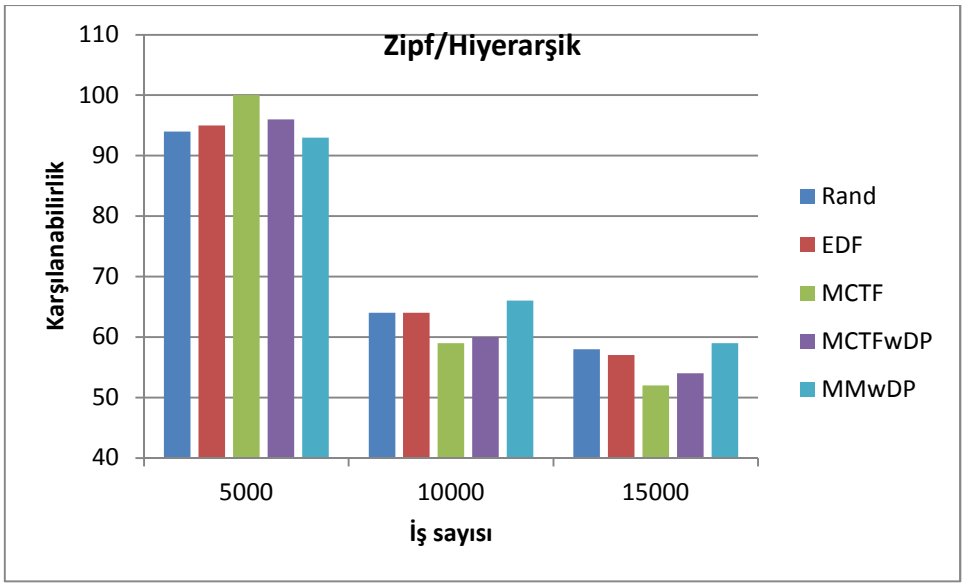
Şekil 3.16. Geometrik veri erişim düzeni ve federatif veri organizasyon modeline sahip sistemde arz edilen iş sayısının performans üzerine etkisi



Şekil 3.17. Geometrik veri erişim düzeni ve hiyerarşik veri organizasyon modeline sahip sistemde arz edilen iş sayısının performans üzerine etkisi



Şekil 3.18. Zipf veri erişim düzeni ve federatif veri organizasyon modeline sahip sistemde arz edilen iş sayısının performans üzerine etkisi



Şekil 3.19. Zipf veri erişim düzeni ve hiyerarşik veri organizasyon modeline sahip sistemde arz edilen iş sayısının performans üzerine etkisi

Şekil 3.18 ve 3.19'da gösterilen sonuçlara göre, geometrik veri erişim düzenine benzer biçimde, zipf veri erişim düzenine sahip sistemlerde de, federatif veri organizasyon modeline sahip sistemler, iş sayısından, hiyerarşik veri organizasyon modeline sahip sistemlere göre daha fazla etkilenmektedirler. Her iki veri organizasyon modelinde de, iş sayısı 5000'e kadar düşürüldüğünde, işlerin tamamına yakın bir bölümü başarı ile tamamlanmaktadır.

Şekil 3.8-3.19'da gösterilen sonuçlara genel olarak bakıldığında, Çizelge 3.5 ve 3.6'da belirtilen simülasyon sonuçları ile de uyumlu olarak, rassal veri erişim düzenine sahip sistemlerde Rand, geometrik ve zipf düzenine sahip sistemlerde ise MMwDP algoritmaları, en iyi performansları göstermektedirler.

Bu çalışmanın ilerleyen bölümlerinde, rassal veri erişim düzenine sahip sistemlerin simülasyonunda Rand, diğer sistemlerin simülasyonunda ise MMwDP iş çizelgeleme algoritmaları kullanılacaktır. Söz konusu algoritmalar, baz çalışmalar sonucunda en iyi performansı gösteren iş çizelgeleme algoritmalarıdır.

4. GERÇEK-ZAMANLI VERİ DAĞITIMI

Bilimsel ve ticari uygulamaların ihtiyaç duyduğu işlem gücü, veri depolama alanı ve ağ bant genişliği gereksinimi, gün geçtikçe artmaktadır. Özellikle, yüksek miktarlarda verinin işlenmesini gerektiren veri yoğun uygulamalar, yüksek enerji fiziği (EU-DataGrid [9]), iklim modellemesi (Earth System Grid [17]), depremler (NEESit [21]) gibi farklı bilim dallarında ortaya çıkmaktadırlar. Veri kaynakları, hadron çarpıştırıcısında bulunan sensörler olabileceği gibi, Dünya'nın farklı bölgelerinde bulunan sismik sensörler veya uydulardan elde edilen resimler de olabilir. Bu uygulamaların büyük bir kısmında, veriler veri kaynakları tarafından yazılırlar ve coğrafik olarak farklı bölgelerde bulunan sistemler üzerinde çalışan uygulamalar tarafından okunurlar.

Örneğin, LHC'de bulunan CMS dedektörünü temel alan deneyde yıllık ortalama 1 petabayt mertebesinde veri üretilir. Bu veri, CERN'de depolanır ve ihtiyaç halinde, araştırmaya katılan diğer bölgelerde bulunan bilim insanlarına gönderilir. Söz konusu verilerin yüksek hızlı bağlantılar üzerinden dağıtılması ve veri yoğun işlerin çalıştırılması için WLCG [18] adı verilen Grid sistemi oluşturulmuştur. Bu sistemde, ana kopyaları CERN'de olmak üzere, veriler farklı sitelerde depolanabilmektedirler.

WLCG ve benzeri Veri Grid sistemlerinde ortaya çıkan veri dağıtımı problemi, farklı sitelerde bulunan verilerin transfer edileceği rotaların bulunması ve her bir transfer için kullanılabilecek olan bant genişliğinin hesaplanması ile ilgilenir. Internet'in en-iyi-girişim (best-effort) servisleri, bant genişliği, gecikme, kayıp oranı ve benzeri konularda herhangi bir garanti vermezler. En-iyi-girişim servisler FTP ve e-posta gibi geleneksel uygulamalar için yeterli görülebilir ise de; VoIP, video konferans, video-on-demand (talebe bağlı video), savunma ve gözetleme [92], kablosuz algılayıcı ağları [93] ve Grid sistemleri [59] gibi son yıllarda artan sıklıkta karşılaşılan uygulamaların gereksinim duyduğu gerçek zaman kriterlerinin karşılanması için yeterli değildirler. *Gerçek-zamanlı* adı verilen bu uygulamalar için, verilerin mümkün olan en kısa zamanda transfer edilmesi yeterli görülmez. Veri transferlerinin *başarılı* sayılabilmeleri için,

zorunlu kılınan Servis Kalitesi (Quality of Service – QoS) gereksinimlerini de karşılamaları gerekmektedir.

Servis kalitesi garantilerini karşılamanın dışında veri dağıtımı problemlerinin, ağ kullanım oranlarını arttırmak gibi amaçları da olabilir. Bu bağlamda, akışları, servis kalitesi gereksinimlerini karşılayacak biçimde, bir rota üzerinde göndermek yeterli olmayabilir. Kaynakların verimli kullanımları da dikkate alınmalıdır. Örneğin, az maliyet içeren akışlar, öncelikli olarak çizelgelenebilir. Gelen veri transferi istekleri arasında bir seçicilik öngören bu mekanizmaya *kabul kontrolü* (*admission control*) [94, 95] veya *istek paketleme* (*request packing*) [96, 97] ismi verilir.

Veri Grid sistemine arz edilmiş gerçek-zamanlı veri yoğun uygulamalar için son zaman servis kalitesi bulunan veri isteklerinin başarısını arttıracak şekilde çizelgeleme yapmak önemli ve zor problemdir. Literatürde, gerçek-zamanlı veri dağıtımı olarak bilinen bu problemin çözümünde verilerin tekli rotadan ve çoklu rotadan dağıtımı olmak üzere iki farklı yaklaşım kullanılmaktadır.

Bu bölümde, öncelikle verilerin tek rotadan transferini öngören Gerçek-Zamanlı Tekli Rotadan Veri Dağıtımı Problemi (Real-Time Unsplittable Data Dissemination Problem – RTU/DDP) ile ilgili çalışma yer almaktadır. Çok bilinen ve polinom zamanda bilinen çözümü olmayan Bölünemez Akış Problemi (Unsplittable Flow Problem)'ni genelleştiren bu problem, biçimsel olarak tanımlanacaktır. Gerçek-zamanlı veri isteklerinin tekli rotadan gönderilmesi probleminin, polinom zamanda çözülemeyeceği gösterilecektir.

RTU/DDP probleminin çözümü için iki farklı yaklaşım sunulacaktır. Birinci yaklaşımda, çizelgeleme algoritmasına arz edilen veri transferi istekleri, sıra ile işleme alınır. İşleme alınan veri transferi isteğinin karşılanacağı rota bulunabilir ise gerekli rezervasyonlar yapılır ve sıradaki veri transferi isteğine geçilir. İkinci yaklaşımda ise; öncelikle tüm veri transferi istekleri için potansiyel rotalar hesaplanır. Sonrasında ise, rotaları önceden hesaplanmış olan veri transferi isteklerinden hangilerinin karşılanacağı bulunur.

Veri transferlerinin, tek rota yerine birden fazla rotadan eşzamanlı olarak gerçekleştirilmesi, ağ kaynaklarının daha verimli kullanılmasını sağlamaktadır [98-100]. Çoklu (birden fazla) rotadan veri transferi, son zamanlarda geliştirilmiş

olan yönlendirme protokolleri tarafından da desteklenmektedir. Örneğin, Grid sistemleri için de çok kullanılan bir protokol olan GridFTP [27], çoklu rotadan eşzamanlı veri transferini desteklemektedir. Çoklu rotadan veri transferi sınıfında sayılan eşit maliyetli çoklu rota yönlendirme yaklaşımı olan Eşit Maliyet Çoklu Rota (Equal Cost Multi Path – ECMP), En Kısa Yolu İlk Aç (Open Shortest Path First - OSPF) [43] ve Ara Sistemden Ara Sisteme (Intermediate System to Intermediate System - IS-IS) yönlendirme protokolleri tarafından desteklenmektedir.

Çoklu rotadan veri dağıtımı bölümünde, veri transferlerinin birden fazla rotadan gerçekleşmesine olanak tanıyan Gerçek-Zamanlı Çoklu Rotadan Veri Dağıtımı Problemi (Real-Time Splittable Data Dissemination Problem – RTS/DDP) tanımlanacaktır. RTU/DDP problemini genelleştiren RTS/DDP problemi de polinom zamanda çözülemez ve bu problemin çözümü için de keşifsel algoritmalar önerilecektir.

RTS/DDP çözümünde ise, RTU/DDP çözümünde en iyi sonuçları veren algoritmalar geliştirilerek kullanılacak ve çoklu rotadan veri gönderilmesini sağlayan yaklaşımlar sunulacaktır. RTS/DDP bölümünün son kısmında ise, çoklu rotadan veri dağıtımının, performans üzerine yaptığı etkiler gösterilecektir.

4.1. İlgili Çalışmalar

Bu bölümde öncelikle tekli rotadan veri dağıtımı ile ilgili çalışmalar, sonrasında ise çoklu rotadan veri dağıtımı üzerine yapılan çalışmalar anlatılacaktır.

4.1.1. Gerçek-Zamanlı Tekli Rotadan Veri Dağıtımı İle İlgili Çalışmalar

Gerçek-zamanlı veri dağıtımı probleminde amaç, başarılı olarak tamamlanan gerçek-zamanlı veri transferi isteği sayısını arttırmaktır. Literatürde, farklı servis kalitesi bazlı problemler ve servis kalitesi bazlı yönlendirme algoritmaları önerilmiştir. Önerilen bu algoritmaların birçoğu, en-iyi-girişim yönlendirme algoritmalarının genişletilmesi ile geliştirilmişlerdir. Bu nedenle bu

bölümde, en-iyi-girişim ve servis kalitesi bazlı yönlendirme algoritmaları özetlenecektir.

4.1.1.1. En-iyi-girişim yönlendirme algoritmaları

En-iyi-girişim yönlendirme problemleri, oldukça geniş bir çözüm uzayına sahiptirler ve en iyi çözümler, polinom zamanda bulunamazlar. Bu problemlerde, en iyi çözümü bulmak yerine iyi çözümler ile yetinilir ve iyi çözümleri bulmak için de keşifsel algoritmalar kullanılır [101]. Dijkstra [40] ve Bellman-Ford [41] gibi keşifsel algoritmalar, en kısa yol probleminin çözümünde kullanılırlar. Benzer algoritmalar, günümüzde kullanılan ağ sistemlerinde sıklıkla kullanılmaktadır. Literatürdeki veri dağıtımı algoritmalarının birçoğu, çok bilinen en kısa yol algoritmalarından türetilmiştir [102-104]. IP bazlı ağlarda, en kısa yol algoritmaları içeren protokollerin kullanılması geleneksel hale gelmiştir.

Yönlendirme Bilgi Protokolü (Routing Information Protocol - RIP) [42], OSPF [43] ve Sınır Geçit Protokolü (Border Gateway Protocol – BGP) [44] gibi güncel Internet protokolleri, en-iyi-girişim yönlendirme protokolleridir. Bu protokoller, sadece hedefe olan en kısa yolu kullanırlar. En kısa yol, en kısa fiziksel mesafeye sahip olan yol olmak zorunda değildir. En az maliyetli yol veya en az atlamalı yollar da literatürde en kısa yol olarak nitelendirilmiştir. En kısa yol algoritmalarında, tüm trafik, en kısa yollar üzerinden yönlendirilir. Alternatif rotalar bulunsa bile, bu rotalar kullanılmazlar. Bu durumda, bazı linkler tam kapasite olarak kullanılmazken, diğer bazı linklerde tıkanıklık oluşabilir.

Güncel bir çalışmada, literatürdeki farklı veri dağıtımı algoritmalarının performansları ölçülmüş ve karşılaştırılmıştır [105]. Bu çalışmada yazarlar, uygun rota, en az atlamalı uygun rota, en geniş/en kısa uygun rota, en kısa/en geniş uygun rota, en kısa mesafeli uygun rota, dinamik alternatifli uygun rota, OSPF benzeri algoritmalar, *k* dinamik rotalar, *k* durağan rotalar, genişletilmiş Bellman-Ford algoritmaları gibi farklı algoritmaları incelemişlerdir. Çalışma sonuçlarına göre en az atlamalı uygun rota ve dinamik alternatifli uygun rota algoritmaları, ağ kullanım oranını en yüksek değerine çıkarma performansında, diğer algoritmalara göre daha iyi sonuçlar vermektedir.

Literatürde, veri dağıtımı problemini çözen benzetimli tavlama (simulated annealing) [106], yasaklı arama (tabu search) [107], karınca kolonisi (ant colony) [108], taşma (flooding) [109] ve vektör dönüştürücü (vector converting) [110] tabanlı alternatif en-iyi-girişim yaklaşımlar da mevcuttur. En popüler alternatif ise genetik algoritma tabanlı olanlardır. Chang [111] ve Munetomo [112, 113], genetik algoritmayı en kısa yol yönlendirme problemini çözmede kullanmışlardır. İki araştırmacı da, kaynak-hedef rotası üzerinde bulunan düğüm noktalarını içeren değişken uzunlukta kromozomlar kullanmışlardır. Munemoto'nun algoritması [112] kablolu veya kablosuz ortamlar için elverişlidir. Sinclair [114], Shimamoto [115], ve Hamdan [116]'da, en kısa yol yönlendirme probleminin çözümünde genetik algoritmadan faydalanmışlardır. [117]'de QoS yönlendirme probleminde genetik algoritma kullanan bir yöntem önerilmiştir. Bu çalışmada, genetik algoritmanın içerisinde, labirent algoritması da kullanılarak daha kararlı sonuçlar üretilebildiği iddia edilmektedir. Çoklu hedef veya çok olana gönderim gibi farklı yönlendirme problemleri için de genetik algoritmayı kullanan yaklaşımlar önerilmiştir [118-120].

4.1.1.2. Servis kalitesi bazlı yönlendirme algoritmaları

Akışlara servis kalitesi sağlayabilmek için iki görevin başarılması gerekmektedir. Birincisi, kaynak-hedef arasında, servis kalitelerini karşılayabilecek uygun bir rota bulunmasıdır. İkincisi ise, rota üzerindeki kaynakların rezerve edilmesidir. Birinci görev servis kalitesi bazlı yönlendirme; ikinci görev ise kaynak rezervasyonu protokolleri tarafından yapılır. Servis kalitesi bazlı yönlendirme ile kaynak rezervasyonu protokolleri, sistemin veri dağıtımı gereksinimlerini karşılayabilmek için ortak çalışmalıdırlar.

Servis Kalitesi Bazlı Yönlendirme: PNNI [45] ve QOSPF [46], ağ durum (link-state) tabanlı algoritmalar ile servis kalitesi desteği sunmaktadır. PNNI, ATM ağlarında, ağ durumu ve topoloji bilgilerine göre yönlendirme yapmak için kullanılır. PNNI, standartlara bağlanmış tek servis kalitesi tabanlı yönlendirme protokolüdür. Servis kalitesi desteği için QOSPF'de, yönlendiriciler topoloji bilgilerinin yanı sıra ağ kaynak bilgilerini de duyurmak zorundadırlar. Ağ kaynak

bilgileri, yönlendirici bilgileri ve ağ bilgilerini içerir. QOSPF'de bulunan rota hesaplama algoritması, servis kalitesi gereksinimlerini karşılamayan linkleri kullanmaksızın, en geniş/en kısa yolu hesaplar. Bu yol en fazla bant genişliğine sahip olan en az atlamalı yoldur. Algoritmik karmaşıklığı, Bellman-Ford'un en kısa yol algoritması gibidir.

Servis kalitesi bazlı yönlendirme problemlerinin farklı çeşitleri mevcuttur. Bu problemlere sunulan bazı çözümler şöyle özetlenebilir:

Bant genişliği sınırlamalı yönlendirme: Gelen veri transferi istekleri için, öncelikle gerekli olan bant genişliği hesaplanır. Yeterli bant genişliğine sahip olmayan rotalar, uygun rota olarak değerlendirilmezler. Bu problemin farklı çözümleri önerilmiştir [121-123].

Bant genişliği sınırlamalı, gecikme en iyilemeli yönlendirme: Bu problem en geniş/en kısa yol problemi gibi veya en kısa/en geniş yol problemi gibi çözülebilir [124]. [125]'te, kaynaktan hedefe, linkler üzerinde yayılım gecikmeleri ve dinamik bant genişliği sınırlamasının olduğu bir ortamda, en az gecikme ile veri transferini amaçlayan, en çabuk yol problemi tanımlamıştır. [126]'da her giriş-çıkış çifti için gecikme ağırlıklı kapasiteyi hesaplayıp, kritik linklere büyük ağırlık değerleri atayan ve bu linklerin kullanımını engelleyen bir algoritma önerilmiştir. Kritik linkler, bir rotaya dâhil edildiğinde, bazı giriş-çıkış çiftlerinin gecikme ağırlıklı kapasite değerlerini düşüren linkler olarak tanımlanmışlardır.

Bant genişliği ve maliyet sınırlamalı yönlendirme: Bu problemin çözümünde, tipik olarak, maliyet veya bant genişliğine sınırlı bir sayı değeri atanır ve Bellman-Ford algoritması veya Dijkstra algoritmasının farklı uzantıları kullanılarak çözülür [127].

Bir çok koşullu yönlendirme: Bu problemin amacı, bir çok farklı koşulun eş zamanlı olarak sağlanmasıdır [122-128]. [128]'de, bir çok koşullu problemlerin çözümü için keşifsel en iyi çözümleri üreten algoritmalar önerilmiştir.

Kaynak Rezervasyonu Protokolleri: En-iyi-girişim yaklaşımlarından farklı olarak, servis kalitesi bazlı yönlendirme yaklaşımları, önceden kaynakları rezerve edecek mekanizmalara ihtiyaç duyarlar. Bu yaklaşımlarda, bir rota bulunur; rota üzerindeki kaynaklardan rezerve edilmesi gerekli olan miktarlar (ağ

bant genişliği, bellek alanı gibi) hesaplanır; transfer gerçekleşmeden önce ilgili tüm kaynaklar rezerve edilir. Transfer gerçekleştikten sonra ise, rota üzerinde kullanılan kaynaklar serbest bırakılırlar.

4.1.2. Gerçek-Zamanlı Çoklu Rotadan Veri Gönderimi İle İlgili Çalışmalar

Geleneksel araştırmalarda [98, 99, 129], uçtan uca gecikmeleri en aza indirmek için uygun rotaların bulunması ve akış yükünün rotalar arasında paylaşımı problemleri incelenmiştir. Ancak, çoklu rotadan veri dağıtımında, "belirli rotalara öncelik verilmesi", "belirli linklerin kullanılmaması" gibi ön koşullara göre kaynak çizelgelenmesi konusu yeni popülerlik kazanmıştır [99]. Son zamanlarda yapılan bazı çalışmalarda [130, 131], verinin birden fazla rota üzerinden transferi probleminde, bant genişliği, izin verilen en fazla sayıda atlama, kaynak yakınlığı gibi koşullar da dikkate alınmaya başlamıştır. [47]'de, link üzerindeki bant genişliklerinin önceden rezerve edilebildiği bir ortamda, birden fazla rotadan QoS yönlendirmeli veri transferi algoritması önerilmiştir. [132]'de, akışları uzun ve kısa süreli diye ikiye ayırarak linklerdeki tıkanmaları büyük oranda azaltan br yaklaşım önerilmiştir.

Çoklu emtia akışı problemini, akışların parçalı şekilde gönderilmesiyle çözme ile ilgili çalışmalar, [48] ile hız kazanmıştır. [133], her rota üzerinde gönderilebilecek akışlara üst limit getirmiş ve linkler üzerindeki yığılmayı en aza indirmeye çalışmıştır. [134] ise problemi iki aşamaya ayırmıştır. Birinci aşamada aday rotaların bulunması aşamasıdır. İkinci aşamada ise hangi rotadan ne kadarlık akışın geçeceğini bulmaya çalışmışlardır. [135]'de, her bir akışın en fazla k parçaya ayrıldığını varsayılarak, farklı k değerleri için teorik performans garantilerinin bulunması üzerinde çalışılmıştır.

Görüldüğü üzere bir veya birden fazla kaynaktan gelen verilerin birden fazla rota üzerinden hedefe gönderilmesi ile ilgili çalışmalar bulunmaktadır. Ancak literatürde gerçek-zamanlı veri transferi taleplerinin birden fazla rota üzerinden gönderilmesi ile ilgili çalışmalar bulunmamaktadır. Bu çalışmada önerilen algoritmalar, gerçek-zamanlı çoklu rotadan veri dağıtımı probleminin çözümü için literatürde yapılan ilk çalışmalar olmaktadır.

4.2. Gerçek-Zamanlı Tekli Rotadan Veri Dağıtımı

Bu bölümde öncelikle problem biçimsel olarak tanımlanacak, problemin çözümünde uygulanan tek aşamalı ve iki aşamalı yaklaşımlar tanıtılacak ve bu yaklaşımlara uygun keşifsel algoritmalar sunulacaktır. Daha sonra ise, bu algoritmaların, DGridSim simülatörü ile yapılan performans ölçümlerinin sonuçları gösterilecektir.

4.2.1. Problem Tanımı

Veri Grid sistemi ağ yapısı, yönsüz bir grafik $G = (V, E)$ ile belirtilmektedir. $V = \{v_1, ..., v_n\}$ sistem üzerindeki heterojen makineleri, $E = \{e_1, ..., e_m\}$ ise iki makineyi bağlayan linkleri göstermektedir. Makineler kısıtlı depolama alanı bulunan depolama elemanları veya yönlendirici olabilirler. Her link $e_l \in E$ için bant genişliği $c_l > 0$ ve gecikme $\delta_l > 0$ olsun.

Sistem üzerinde gerçek-zamanlı veri transferi isteklerinin karşılanması için veri dağıtımı çizelgelemelerini oluşturan bir veri dağıtımı algoritması vardır. Bölüm 2'de önerilen Veri Grid sistemi modelinde, veri dağıtımı algoritması veri yönetim servisinde gerçeklenmektedir. Veri yönetim servisi, veri dağıtımı algoritması tarafından oluşturulan çizelgelemeler için, rezervasyon servisine rezervasyon istekleri göndererek ilgili kaynakların veri transferi süresince rezervasyonunu sağlamaktadır. Bu varsayımlar, literatürdeki benzer çalışmalar ile uyumludur [4, 53, 59, 67, 83, 136].

$R = \{r_1, ..., r_k\}$ gerçek-zamanlı veri transferi istekleri olsun. Her istek $r_i \in R$, $<s_i, t_i, f_i, d_i>$ dörtlüsü ile modellenir. Bu dörtlüde, s_i kaynak makinesini (veri depolama elemanı), t_i hedef makinesini (işlem elemanı veya veri depolama elemanı), f_i istekte bulunulan veriyi ve d_i ise isteğin başarılı şekilde karşılanmış sayılması için transferin tamamlanması gereken son zamanı belirtmektedir. Ayrıca, $P_i = \{p_1, ..., p_{l_i}\}$, $r_i \in R$ isteğini karşılayabilecek rotalar kümesi ve $l_i > 0$'de rota sayısı olsun.

Tanımlama 1: *Rota*, içinde bir linkin en fazla bir defa kullanıldığı sınırlı sayıda linkin bir araya gelmesi ile oluşan bir gruptur.

Tanımlama 2: İstek $r_i \epsilon R$, eğer istekte bulunulan veri f_i, kaynak makine s_i'den hedef makine t_i'ye, iki makine arasında bulunan rotaların birisinden, d_i son zamanından önce transfer edilir ise, *karşılanabilir*dir.

Tanımlama 3: İstek $r_i \epsilon R$ için $p_j \epsilon P_i$ rotası, ancak ve ancak, $p_j \epsilon P_i$ rotasının üzerindeki linklerin kullanılabilir bant genişliği, istek $r_i \epsilon R$ için [*ŞimdikiZaman*, d_i] zaman aralığında gerekli en az bant genişliğinden büyük ise *uygun* bir rotadır. İstek $r_i \epsilon R$ için gerekli en az bant genişliği, $\beta_i = |f_i|/d_i$ şeklinde hesaplanır ve $|f_i|$, f_i verisinin büyüklüğünü ifade eder.

Literatürde çalışılmış olan en-iyi-girişim veri dağıtımı problemleri, tüm veri transferi isteklerinin en az zamanda tamamlanması, veri transferleri için tanımlanmış olan bir maliyet fonksiyonunun en aza indirilmesi veya tüm linklerin kullanım verimliliklerinin en yükseğe çıkarılması gibi hedefler tanımlamışlardır. Gerçek-zamanlı veri transferleri üzerine olan bu çalışmada kullanılan performans kriteri, Tanımlama 4'te biçimsel olarak belirtilmiştir.

Tanımlama 4: Veri Grid sistemi $G = (V, E)$ ve gerçek-zamanlı veri transferi istekleri R verilmiş olsun. Gerçek-zamanlı tekli rotadan veri dağıtımı problemi (Real Time Unsplittable Data Dissemination Problem - RTU/DDP), her veri transferinin tek rotadan gönderildiği durumda karşılanan veri transferi sayısını en yükseğe çıkarmaya çalışır.

$$\text{RTU/DDP: azami} \sum_{r_i \in R} \sum_{p_j \in P_i} x_{ij}$$

$$\sum_{p_j \in P_i} x_{ij} \leq 1, \text{ tüm } r_i \epsilon R \text{ için}$$

$$\sum_{r_i \in R} \sum_{p_j \in P_i} \xi_{lij} \pi_{ij} x_{ij} \leq c_l, \text{ tüm } e_l \epsilon E \text{ için}$$

$$x_{ij} = \{0, 1\}, \text{ tüm } r_i \epsilon R \text{ ve } p_j \epsilon P_i \text{ için}$$

Eğer link $e_l \epsilon E$, istek $r_i \epsilon R$'yi karşılayan rota $p_j \epsilon P_i$ üzerinde ise ξ_{lij} 1, aksi durumda ise 0'dır. İstek $r_i \epsilon R$'nin rota $p_j \epsilon P_i$ üzerinden karşılanması için

$$\pi_{ij} = \frac{|f_i|}{d_i - \sum_{e_l \in p_j} \delta_l}$$

büyüklüğünde akışa ihtiyaç duyulur.

Teorem 1: RTU/DDP, NP-zor bir problemdir.

İspat: Bölünemez Akış Problemi (Unsplittable Flow Problem - UFP), NP-zor bir problemdir [137] ve polinom zamanda RTU/DDP problemine dönüştürülebilir. UFP'de, RTU/DDP'den farklı olarak her istek $r_i \epsilon R$, $<s_i, t_i, \pi_i, \omega_i>$ dörtlüsü ile modellenir. Bu dörtlüde $\pi_i > 0$ gerekli bant genişliğini, $\omega_i > 0$ de $r_i \epsilon R$ isteğinin gerçekleşmesinden elde edilen kârı gösterir. UFP, link gecikmelerini ve istek için belirlenen son zamanları dikkate almaz.

$$\text{UFP: } azami \sum_{r_i \in R} \sum_{p_j \in P_i} \omega_i x_{ij}$$

$$\sum_{p_j \in P_i} x_{ij} \leq 1, \text{ tüm } r_i \epsilon R \text{ için}$$

$$\sum_{r_i \in R} \sum_{p_j \in P_i} \xi_{lij} \pi_i x_{ij} \leq c_l, \text{ tüm } e_l \epsilon E \text{ için}$$

$$x_{ij} \epsilon \{0,1\} \text{ tüm } r_i \epsilon R \text{ ve } p_j \epsilon P_i \text{ için}$$

UFP ve RTU/DDP problemi, aşağıdaki koşullar sağlandığında eşit olur:

1. Tüm $r_i \epsilon R$ için $\omega_i = 1$.
2. Tüm $r_i \epsilon R$ için $\pi_i = \pi_{ij} = |f_i|/d_i$. Tüm $e_l \epsilon E$ için link gecikmeleri, $\delta_l = 0$.

4.2.2. Tek Aşamalı Yaklaşım

Tekli rotadan veri dağıtımı ile ilgili literatürde, Bölüm 4.1'de de anlatıldığı gibi, genellikle en kısa yol bazlı algoritmalar önerilmiştir. Bu alt bölümde SP_MinHop, SP_MinDelay, SP_MinMin ve SP_MinCon olmak üzere dört algoritma tanıtılacaktır.

4.2.2.1. Tek aşamalı en az atlamalı

Baz çalışmalarda da kullanılan bu algoritma, Bölüm 2.4.2'de anlatılmıştır ve Şekil 4.1'de gösterilmiştir. Tek aşamalı en az atlamalı (Single Pass Minimum Hop – SP_MinHop) algoritma, gelen istekleri sıra ile değerlendirmekte ve tek bir döngü içinde gerekli tüm işlemleri gerçekleştirmektedir.

SP_MinHop algoritması
// Giriş: *G(V,E)* ve istek kümesi *R*
// Çıkış: *KarşılanabilirİstekListesi*

for each istek *r* **in** *R*
GerekliBantGenişliği = $|f_r|/d_r$;
tempG = *G*;
tempG grafiğinden *GerekliBantGenişliği*'ne sahip olmayan linkleri sil;
r.Hedef'ten Dijkstra en az atlamalı rota bulma algoritması ile tüm veri depolama elemanlarına olan uzaklıkları (rota üzerindeki link sayısı) bul;
r.Kaynak = en yakın veri depolama elemanı;
r.Rota = (*r.Kaynak*, *r.Hedef*) arasında *tempG* üzerinde en az atlamalı (en kısa) rota;
r.RotaÜzerindeKullanılanBantGenişliği = $|f_r|/(d_r -$ *RotaÜzerindekiLinklerinGecikmeToplamı*);
if (*r.Rota* ≠ *null*) **ve** (*r.RotaÜzerindeKullanılanBantGenişliği* ≤ *RotaÜzerindeLinklerinEnAzKullanılabilirBantGenişliği*) **then**
G üzerindeki linklerden *r.Rota* üzerindeki olanların kullanılan bant genişliklerini arttır;
KarşılanabilirİstekListesi += <*r*, *r.Kaynak*, *r.Hedef*, *r.Rota*, *r.RotaÜzerindeKullanılanBantGenişliği*>;
end
end for
return *KarşılanabilirİstekListesi*;
end program.

Şekil 4.1. SP_MinHop algoritması

Algoritma, öncelikle veri transferi isteğinin sağlanabilmesi için gerekli olan en az bant genişliğini (*GerekliBantGenişliği*) hesaplar. Ağ grafiğinden, *GerekliBantGenişliği*'ne sahip olmayan linkleri sildikten sonra, Dijkstra'nın en az atlamalı rota bulma algoritmasını kullanarak, veri transferi isteğinde bulunan sitedeki veri depolama elemanı (*HedefDepolamaElemanı*) veya işlem elemanı

(*SeçilenİşlemElemanı*) *r.Hedef* olmak üzere, diğer veri depolama elemanlarına olan uzaklıkları bulunur. Veri transferine kaynak olabilecek veri depolama elemanları arasından *r.Hedef* depolama elemanına en yakın depolama elemanı *r.Kaynak* olarak seçilir. Her istek için *RotaÜzerindeKullanılanBantGenişliği* hesaplanır. Hesaplama sırasında, rota üzerindeki linklerin gecikme değerleri toplamını ifade eden *RotaÜzerindekiLinklerinGecikmeToplamı* de dikkate alınır. Veri transferinin karşılanabilmesi için en az bir rota bulunmuş olması ve bulunan rotanın uygun olması gerekir. Rotanın uygun sayılabilmesi için, rota üzerindeki linklerin arasından en az kullanılabilir bant genişliğine sahip olan linkin kullanılabilir bant genişliği, *RotaÜzerindeKullanılanBantGenişliği* değerinden büyük veya eşit olmalıdır. Eğer veri transferi isteği karşılanabilir ise, ağ kaynaklarındaki kullanılan bant genişlikleri arttırılır ve istek *KarşılanabilirİstekListesi*'ne ilave edilir.

En az atlamalı yol algoritmaları O(|V||E|)'de çalışır. Dolayısı ile SP_MinHop algoritması O(|V||E||R|)'de çalışır.

4.2.2.2. Tek aşamalı en az gecikmeli

Tek aşamalı en az gecikmeli (Single Pass Minimum Delay – SP_MinDelay) algoritmasının SP_MinHop'tan farkı, Dijkstra'nın linklere 1 değeri vererek bulduğu en az atlamalı rotalar yerine, link gecikmeleri değerlerini vererek bulduğu en az gecikmeli rotaları kullanmasıdır. SP_MinDelay algoritması da SP_MinHop gibi O(|V||E||R|)'de çalışır.

4.2.2.3. Tek aşamalı MinMin

Tek aşamalı MinMin (Single Pass MinMin – SP_MinMin) algoritmasında öncelik en az atlamalı rotalara verilir. Eğer farklı rotalardaki atlama sayısı eşit çıkar ise, en az gecikmeli olan rota tercih edilir.

En az atlamalı yol problemi O(|V||E|)'de çözülür. SP_MinMin algoritmasında, SP_MinHop algoritmasına ilave olarak, rota üzerindeki tüm düğümler için gecikme değerleri hesaplanır. Bu işlem, en kötü senaryo için |E|

aşamada yapılır. Bu durumda, bir istek için gerekli rota $O(|V||E|^2)$'de bulunur ve SP_MinMin algoritması $O(|V||E|^2|R|)$'de çalışır.

4.2.2.4. Tek aşamalı MinCon

Tek aşamalı MinCon (Single Pass MinCon – SP_MinCon) algoritmasında, linklerin kullanım oranları dengelenmeye çalışılmıştır. Algoritma, eğer bir veri transferi için bir link kullanılıyor ise, o linkin daha sonraki veri transferleri için kullanılmasının zorlaştırılması esasına dayanır. Öncelikle tüm linklere 1 değeri atanır. Eğer link, bir veri transferi için kullanılıyor ise, o linkin değeri 1 arttırılır. Böylece, o linkin, Dijkstra'nın en kısa yol algoritmasında kullanılması olasılığı azaltılmış olur. Algoritma, Şekil 4.2'de gösterilmiştir.

SP_MinCon algoritması
// Giriş: *G(V,E)* ve istek kümesi *R*
// Çıkış: *KarşılanabilirİstekListesi*

Veri Grid sistemindeki tüm linklere 1 ağırlık değeri atanır;
for each istek *r* **in** *R*
  GerekliBantGenişliği = $|f_r|/d_r$;
  tempG = *G*;
  tempG grafiğinden *GerekliBantGenişliği*'ne sahip olmayan linkleri sil;
  r.Hedef, link ağırlıklarına göre, Dijkstra en kısa yol bulma algoritması ile tüm veri depolama elemanlarına olan uzaklıkları bul;
  r.Kaynak = en yakın veri depolama elemanı;
  r.Rota = (*r.Kaynak*, *r.Hedef*) arasında *tempG* üzerinde en kısa rota;
  r.RotaÜzerindeKullanılanBantGenişliği = $|f_r|/(d_r -$ *RotaÜzerindekiLinklerinGecikmeToplamı*);
  if (*r.Rota* ≠ *null*) **ve** (*r.RotaÜzerindeKullanılanBantGenişliği* ≤ *RotaÜzerindeLinklerinKullanılabilirBantGenişliği*) **then**
    G üzerindeki linklerden *r.Rota* üzerindeki olanların kullanılan bant genişliklerini arttır;
    r.Rota üzerindeki tüm linklerin ağırlık değerlerini 1 arttır;
    KarşılanabilirİstekListesi += <*r*, *r.Kaynak*, *r.Hedef*, *r.Rota*, *r.RotaÜzerindeKullanılanBantGenişliği*>;
  end
end for
return *KarşılanabilirİstekListesi*;
end program.

Şekil 4.2. SP_MinCon algoritması

En az atlamalı yol algoritmasına ilave olarak, bulunan rota üzerindeki her linkin ağırlık değerleri değiştirilir. Bu işlem, en kötü senaryoda |E| aşamada yapılır. Bu durumda, bir istek için gerekli rota $O(|V||E|^2)$'de bulunur ve SP_MinCon algoritması $O(|V||E|^2|R|)$'de çalışır.

4.2.3. Çift Aşamalı Yaklaşım

Bu yaklaşımda öncelikle gerçek-zamanlı rota seçiminin nasıl yapılacağı anlatılacak, sonrasında ise bu rotaları da kullanarak hangi isteklerin karşılanacağını bulan gerçek-zamanlı istek seçimi tanıtılacaktır. Her iki aşama için de keşifsel algoritmalar önerilecektir.

4.2.3.1. Gerçek-zamanlı rota seçimi

Gerçek-zamanlı rota seçimi (GZRS) aşaması, istek seçimi aşaması için bir ön işlemdir. GZRS aşamasında, birçok keşifsel algoritma kullanılabilir. Bu çalışma kapsamında, aşağıdaki algoritmalar önerilmiş ve performansları incelenmiştir.

En az atlamalı uygun rota: En az atlamalı uygun rota (Minimum Hop Feasible Path First - MinHop/FPF) algoritması, Şekil 4.3'te gösterilmiştir.

MinHop/FPF algoritması, Bellman-Ford benzeri en kısa yol algoritmaları baz alınarak hazırlanmıştır. Algoritma, kaynak-hedef arasındaki en az atlamalı uygun rotalara öncelik tanır. Rota hesaplanırken, sadece gerekli bant genişliğine sahip olan linkler kullanılır. Bellman-Ford veya Dijkstra algoritmaları, en az atlamalı yolun da dâhil olduğu en kısa yol yaklaşımları için en çok kullanılan algoritmalardandır.

En az atlamalı yol algoritmaları $O(|V||E|)$'de çalışır. Dolayısı ile MinHop/FPF (Şekil 4.3) algoritması $O(|V||E||R|)$'de çalışır.

MinHop/FPF algoritması

// Giriş: *G(V,E)* ve istek kümesi *R*
// Çıkış: *GZRS_Listesi*

for each istek *r* **in** *R*
GerekliBantGenişliği = $|f_r|/d_r$;
tempG = *G*;
tempG grafiğinden *GerekliBantGenişliği*'ne sahip olmayan linkleri sil;
r.Hedef'ten Dijkstra en az atlamalı rota bulma algoritması ile tüm veri depolama elemanlarına olan uzaklıkları (rota üzerindeki link sayısı) bul;
r.Kaynak = en yakın veri depolama elemanı;
r.Rota = (*r.Kaynak*, *r.Hedef*) arasında *tempG* üzerinde en kısa rota;
r.RotaÜzerindeKullanılanBantGenişliği = $|f_r|/(d_r -$ *RotaÜzerindekiLinklerinGecikmeToplamı*);
GZRS_Listesi += <*r*, *r.Kaynak*, *r.Hedef*, *r.Rota*, *r.RotaÜzerindeKullanılanBantGenişliği*>;
end for
return *GZRS_Listesi*;
end program.

Şekil 4.3. MinHop/FPF algoritması

MinDelay/FPF algoritması

// Giriş: *G(V,E)* ve istek kümesi *R*
// Çıkış: *GZRS_Listesi*

for each istek *r* **in** *R*
GerekliBantGenişliği = $|f_r|/d_r$;
tempG = *G*;
tempG grafiğinden *GerekliBantGenişliği*'ne sahip olmayan linkleri sil;
r.Kaynak'tan Dijkstra en kısa yol bulma algoritması ile tüm veri depolama elemanlarına en az link gecikmeli rotaları bul;
r.Hedef = en yakın veri depolama elemanı;
r.Rota = (*r.Kaynak*, *r.Hedef*) arasında *tempG* üzerinde en az link gecikmeli rota;
r.RotaÜzerindeKullanılanBantGenişliği = $|f_r|/(d_r -$ *RotaÜzerindekiLinklerinGecikmeToplamı*);
GZRS_Listesi += <*r*, *r.Kaynak*, *r.Hedef*, *r.Rota*, *r.RotaÜzerindeKullanılanBantGenişliği*>;
end for
return *GZRS_Listesi*;
end program.

Şekil 4.4. MinDelay/FPF algoritması

En az gecikmeli uygun rota: Şekil 4.4'te gösterilen en az gecikmeli uygun rota (Minimum Delay Feasible Path First - MinDelay/FPF) algoritması, öncelikle istek için gerekli olan bant genişliğine sahip olan linkleri belirler. Bu linkleri kullanarak, kaynak-hedef arasında en az toplam link gecikmesini sağlayacak olan rotaları bulur.

En kısa yol algoritmaları O(|V||E|)'de çalışır. Dolayısı ile MinDelay/FPF (Şekil 4.4) algoritması da MinHop/FPF algoritması gibi O(|V||E||R|)'de çalışır.

En az atlamalı en az gecikmeli uygun rota: Şekil 4.5'de gösterilen en az atlamalı en az gecikmeli uygun rota (Minimum Hop Minimum Delay Feasible Path First - MinMin/FPF) algoritması, önceliği en az atlamalı rotalara verir. İki rotanın atlama sayısı eşit ise, daha az toplam link gecikme değerine sahip olan rota tercih edilir.

MinMin/FPF algoritması
// Giriş: *G(V,E)* ve istek kümesi *R*
// Çıkış: *GZRS_Listesi*

for each istek *r* **in** *R*
GerekliBantGenişliği = $|f_r|/d_r$;
tempG = *G*;
tempG grafiğinden *GerekliBantGenişliği*'ne sahip olmayan linkleri sil;
r.Kaynak'tan Dijkstra en kısa yol bulma algoritması ile tüm veri depolama elemanlarına en az atlamalı en az gecikmeli rotaları bul;
r.Hedef = en yakın veri depolama elemanı;
r.Rota = (*r.Kaynak*, *r.Hedef*) arasında *tempG* üzerinde en az atlamalı en az link gecikmeli rota;
r.RotaÜzerindeKullanılanBantGenişliği = $|f_r|/(d_r -$ *RotaÜzerindekiLinklerinGecikmeToplamı*);
GZRS_Listesi += <*r*, *r.Kaynak*, *r.Hedef*, *r.Rota*, *r.RotaÜzerindeKullanılanBantGenişliği*>;
end for
return *GZRS_Listesi*;
end program.

Şekil 4.5. MinMin/FPF algoritması

MinHop/FPF algoritması O(|V||E|)'de çalışır. MinMin/FPF algoritmasında, MinHop/FPF'ye ilave olarak, her düğüm için rota gecikmesi hesap edilir. Bu işlem, en kötü senaryo için |E| aşamada yapılır. Bu durumda, bir istek için gerekli rota O(|V||E|^2)'de bulunur ve MinMin/FPF algoritması (Şekil 4.5) tüm rotaları O(|V||E|^2|R|)'de bulur.

En az çakışmalı uygun rota: MinHop/FPF, MinDelay/FPF ve MinMin/FPF algoritmaları ile bulunan rotalar, bazı linkleri daha çok, bazı linkleri de daha az kullanıyor olabilirler. Ayrıca, linklerin kapasiteleri de farklılık göstermektedir. Az bant genişliğine sahip linklerin bant genişliği kapasiteleri daha çabuk tükenir ve bu nedenle söz konusu linkler daha dikkatli kullanılmalıdırlar.

Önerilen en az çakışmalı uygun rota (Minimum Contention Feasible Path First - MinCon/FPF) algoritması, Şekil 4.6'da gösterilmiştir.

MinCon/FPF algoritmasında, en az ağırlıklı en az gecikmeli uygun rotanın bulunması için, öncelikle linklere ağırlık olarak 1 atanır. Her istek için bulunan rotaların üzerindeki linklerin ağırlıkları, *OrtalamaBantGenişliği*/*e.BantGenişliği* kadar arttırılır. Linklere ilave edilen ağırlık, linkin bant genişliği ile ters orantılıdır. Bu durumda, daha az bant genişliğine sahip olan linkler, daha fazla ağırlığa sahip olacaklardır ve bu linklerin daha sonraki rotalarda seçilme olasılıkları da azalmış olacaktır.

MinHop/FPF algoritması O(|V||E|)'de çalışır. MinCon/FPF algoritmasında, MinHop/FPF'ye ilave olarak, bulunan rota üzerindeki her düğümün ağırlık değerleri değiştirilir. Bu işlem, en kötü senaryoda |E| aşamada yapılır. Bu durumda, bir istek için gerekli rota O(|V||E|^2)'de bulunur ve MinCon/FPF algoritması (Şekil 4.6) tüm rotaları O(|V||E|^2|R|)'de bulur.

MinCon/FPF algoritması

// Giriş: *G(V,E)* ve istek kümesi *R*
// Çıkış: *GZRS_Listesi*

G'de bulunan tüm $e \in E$ için *e.Ağırlık* 1 olarak atanır;
LinklerinOrtalamaBantGenişliği = *G*'de bulunan tüm $e_l \in E$ için ortalama bant genişliği;
for each istek *r* **in** *R*
  GerekliBantGenişliği = $|f_r|/d_r$;
  tempG = *G*;
  tempG grafiğinden *GerekliBantGenişliği*'ne sahip olmayan linkleri sil;
  r.Kaynak'tan Dijkstra en kısa yol bulma algoritması ile tüm veri depolama elemanlarına olan en az ağırlıklı rotaları bul;
  r.Hedef = en yakın veri depolama elemanı;
  r.Rota = (*r.Kaynak*, *r.Hedef*) arasında *tempG* üzerinde en az ağırlıklı rota;
  r.RotaÜzerindeKullanılanBantGenişliği = $|f_r|/(d_r -$ *RotaÜzerindekiLinklerinGecikmeToplamı*);
  GZRS_Listesi += <*r*, *r.Kaynak*, *r.Hedef*, *r.Rota*, *r.RotaÜzerindeKullanılanBantGenişliği*>;
  for each link *e* **in** *r.Rota*
    e.Ağırlık += *LinklerinOrtalamaBantGenişliği*/*e.BantGenişliği*;
  end for
end for
return *GZRS_Listesi*;
end program.

Şekil 4.6. MinCon/FPF algoritması

4.2.3.2. Gerçek-zamanlı istek seçimi

Önceki bölümde belirtilen GZRS algoritmalarından biri kullanılarak her istek için bir rota bulunduğunu varsayan gerçek-zamanlı istek seçimi (GZİS), karşılanan istek sayısını en fazlaya çıkartacak ve link bant genişliği sınırlamalarını da sağlayacak şekilde isteklerin seçilmesi problemidir. GZİS, matematiksel olarak aşağıdaki şekilde tanımlanır.

Tanımlama 5: Veri Grid sistemi $G = (V, E)$, herbir $r_i \in R$ isteğinin karşılandığı rota $p_j \in P_i$ ve gerekli bant genişliği, π_{ij}, bilgilerinin bulunduğu gerçek-zamanlı veri transferi istekleri kümesi *R* verilmiş olsun. GZİS problemi karşılanan istek sayısını en fazlaya çıkartacak şekilde karşılanabilir istekleri seçer.

$$\text{GZİS: } azami \sum_{r_i \in R} x_i$$

$$\sum_{r_i \in R} \xi_{lij} \pi_{ij} x_i \leq c_l, \text{ tüm } e_l \epsilon E \text{ için}$$

$$x_i \epsilon \{0,1\} \text{ tüm } r_i \epsilon R \text{ için}$$

Formüllerde belirtilen x_i, istek $r_i \epsilon R$ karşılanabilir ise 1, karşılanamaz ise 0'dır.

Teorem 2: GZİS problemi NP-zor bir problemdir.

İspat: Çok boyutlu 0-1 sırt çantası problemi (Multidimensional 0-1 Knapsack Problem - MKP), çok bilinen bir NP-zor problemdir ve GZİS problemine polinom zamanda dönüşür.

$$\text{MKP: } azami \sum_{i=1}^{n} \omega_i x_i$$

$$\sum_{i=1}^{n} a_{il} x_i \leq c_j, 1 \leq l \leq m$$

$$x_i \epsilon \{0,1\}, 1 \leq i \leq n$$

Formüllerde belirtilen n sırt çantasına yerleştirilecek maddelerin kaç çeşit olduğunu, m ise kaç sırt çantasına yerleştirileceğini belirtir. Madde i çantaya konulduğunda $\omega_i > 0$ kadar kazanç sağlar. Çanta l, $c_l > 0$ kapasitesine sahiptir. Madde i, çanta l'ye konulursa a_{ij} (tüm $1 \leq l \leq m$ için $0 \leq a_{il} \leq c_l$) kadar kaynak kullanır. Madde i eğer çantalara konulacak ise $x_i = 1$, konulmayacak ise $x_i = 0$ olur. MKP ve GZİS problemleri aşağıdaki şartlar sağlanır ise eşit olur:

1. n: istek sayısı ve m: link sayısı olsun,
2. $\omega_i = 1$, tüm $1 \leq i \leq n$ için, ve
3. $a_{il} = \xi_{lij} \pi_{ij} = \xi_{lij} |f_i| / \delta_i$, tüm $r_i \epsilon R$ ve $p_j \epsilon P_i$ için

MKP, iyi bilinen bir tamsayı programlama problemidir. Dallandır ve sınırla (branch and bound) ve dinamik programlama yöntemleri ile optimal sonuçlar bulunabilir. Ancak söz konusu yaklaşımlar, sadece küçük ağ yapılarında ve az sayıda istek için sonuç üretmektedirler. Daha büyük problemler için keşifsel

yaklaşımlar uygulanmalıdır. Bu çalışma kapsamında gerçeklenen ve performans karşılaştırılması yapılan algoritmalar aşağıda anlatılmıştır:

Sıradan karşılanamayan akış ilk: Sıradan karşılanamayan akış ilk (In Order Unsatisfiable Flow First Algoritm - IO_UFF) algoritması Şekil 4.7'de gösterilmiştir. İstekler sıra ile değerlendirmeye alınır. Eğer istek kabul edildiğinde, kapasitesini aşacak olan en az bir link var ise, istek reddedilir. Aksi durumda isteğin karşılanacağı rotanın uygun rota olması kontrol edildikten sonra istek kabul edilir.

```
IO_UFF algoritması
  // Giriş: GZRS_Listesi
  // Çıkış: KarşılanabilirİstekListesi
  for each istek r in R
    r.RotaÜzerindeKullanılanBantGenişliği'ni, r.Rota üzerindeki linklere ilave
      et;
    if en az bir link kapasitesini aşmış ise then
      r. RotaÜzerindeKullanılanBantGenişliği'ni, isteğin kullanacağı linklerden
        çıkar;
    else
      if (r.RotaÜzerindeKullanılanBantGenişliği ≤
        RotaÜzerindeLinklerinKullanılabilirBantGenişliği) then
        KarşılanabilirİstekListesi += <r, r.Kaynak, r.Hedef, r.Rota,
          r.RotaÜzerindeKullanılanBantGenişliği>;
      end if
    end if
  end for
  return KarşılanabilirİstekListesi;
end program.
```

Şekil 4.7. IO_UFF algoritması

Bir istek için gerekli bant genişliklerinin, isteğin kullanacağı linklere ilave edilmesi ve kapasite aşım kontrolü, en kötü durumda O(|E|)'de yapılır. İşlem |R| defa tekrar edeceği için IO_UFF algoritması, O(|E||R|) zaman alır.

En fazla sayıda çıkan akış ilk: En fazla sayıda çıkan akış ilk (Maximum Number of Outgoing Flows First – MNOFF) algoritması Şekil 4.8'de gösterilmiştir. MNOFF, ana döngüde GZRS algoritmalarının sağlamış olduğu rotalara bağlı olarak bir çakışma grafiği oluşturur.

Çakışma grafiği çift taraflı (bipartite) grafiktir ve şu şekilde oluşturulur: Her bir link için, linki kullanan istekler GZRS algoritması tarafından bulunur. Tüm isteklerin, gerekli en az bant genişliği ihtiyaçları kadar bant genişliği kullandıkları varsayılır. Bu durumda bant genişliği kapasitesini aşan linklere *tıkanık link* adı verilir. Çakışma grafiğinin sağ tarafına her bir tıkanık link için bir düğüm; sol tarafına ise tıkanıklığa sebep olan her bir istek için bir düğüm yerleştirilir. Her bir istek ile o isteğin gerçekleşmesinde kullanılacak olan tıkanık linkler arasında bir bağlantı kurularak çift taraflı grafik tamamlanır.

```
MNOFF algoritması
  // Giriş: GZRS_Listesi
  // Çıkış: KarşılanabilirİstekListesi
  for each istek r in R
    r.durum = karşılanabilir;
    Çakışma grafiğini oluştur;
  end for
  while çakışma grafiğinde tıkanık link var
    r = çakışma grafiğinde en fazla sayıda çıkan akışlı istek;
    r.durum = karşılanamaz;
    Çakışma grafiğini güncelle;
  end while
  for each istek r in R
   if r.durum == karşılanabilir then
     if (r.RotaÜzerindeKullanılanBantGenişliği ≤
       RotaÜzerindeLinklerinKullanılabilirBantGenişliği) then
       KarşılanabilirİstekListesi += <r, r.Kaynak, r.Hedef, r.Rota,
         r.RotaÜzerindeKullanılanBantGenişliği>;
     end if
   end if
  end for
  return KarşılanabilirİstekListesi;
end program.
```

Şekil 4.8. MNOFF algoritması

Çakışma grafiğinde bulunan bir tıkanık link, bir veya birden fazla isteğin karşılanamayacağını gösterir. Dolayısı ile en az bir istek çakışma grafiğinden silinmelidir. MNOFF, her iterasyonda en fazla sayıda çıkan akışa sahip olan düğümü *karşılanamaz* olarak işaretler. Eşitlik durumunda, daha yüksek bant genişliği ihtiyacı olan istek işaretlenir. *Karşılanamaz* olarak işaretlenmemiş olan istekler çakışma grafiğinden silinerek, prosedür çakışma grafiğinde tıkanık link kalmayana kadar tekrar edilir. Sonrasında ise, *karşılanamaz* olarak işaretlenmiş olmayan tüm istekler için kullanılacak olan rotaların uygun rota olduğu kontrol edilir. Eğer rotalar uygun rota ise, istek *KarşılanabilirİstekListesi*'ne dâhil edilir.

Çakışma grafiğinin oluşturulması veya güncellenmesi O(|E||R|) zaman alır. En kötü durumda ana döngü |R| defa tekrar eder. Bu durumda MNOFF algoritması en kötü durumda $O(|E||R|^2)$ zaman alır.

En fazla miktarda çıkan akış ilk: Şekil 4.9'da gösterilen en fazla miktarda çıkan akış ilk (Maximum Outgoing Flows First – MOFF) algoritması, MNOFF algoritmasına benzemektedir.

MOFF algoritmasında, MNOFF algoritmasından farklı olarak, çakışma grafiğinde tıkanık link olduğu durumda, en fazla sayıda çıkan akışlı istek yerine en fazla miktarda çıkan akışlı isteğin karşılanamaz olarak işaretlenir. İstekler için; çıkan akış miktarları, çıkan istek sayısının gerekli olan en az bant genişliği ile çarpılması sonucunda bulunur. MOFF algoritmasının zaman karmaşıklığı, MNOFF algoritmasının zaman karmaşıklığına benzer şekilde $O(|E||R|^2)$'dir.

```
MOFF algoritması
 // Giriş: GZRS_Listesi
 // Çıkış: KarşılanabilirİstekListesi
 for each istek r in R
  r.durum = karşılanabilir;
 Çakışma grafiğini oluştur;
 while çakışma grafiğinde tıkanık link var
  r = çakışma grafiğinde en fazla miktarda çıkan akışlı istek;
   r.durum = karşılanamaz;
   Çakışma grafiğini güncelle;
 end while
 for each istek r in R
  if r.durum == karşılanabilir then
    if (r.RotaÜzerindeKullanılanBantGenişliği ≤
      RotaÜzerindeLinklerinKullanılabilirBantGenişliği) then
      KarşılanabilirİstekListesi += <r, r.Kaynak, r.Hedef, r.Rota,
        r.RotaÜzerindeKullanılanBantGenişliği>;
    end if
  end if
 end for
 return KarşılanabilirİstekListesi;
end program.
```

Şekil 4.9. MOFF algoritması

Genetik algoritma: Genetik algoritmalar (Genetic Algorithm - GA), bir çok karmaşık programın keşifsel çözümünde başarı ile uygulanmaktadır. GZİS probleminin keşifsel çözümlemesinde de, Şekil 4.10'da ana hatları gösterilen, genetik algoritma kullanılabilir.

Genetik algoritmada kullanılan kromozomlar, *İstekSayısı* uzunluğundadırlar. Eğer istek karşılanabilir ise ilgili gen 1, karşılanamaz ise 0 değerini alır. Genetik algoritmada gerekli olan ilk çözüm uzayı oluşturulurken, öncelikle istekler rassal bir sırayla dizilir. Sıradaki ilk istek alınır. İsteğin kullanacağı linklerde, isteğin gereksinim duyduğu bant genişliği ilave edilir. Eğer kapasitesini aşan link yoksa, istek karşılanabilir olarak işaretlenir. En az bir link kapasitesini aşana kadar sıradaki işlere aynı işlem uygulanır. Bu işlem sonucunda bir adet çözüm oluşturulmuş olur. Bu işlem *ÇözümUzayıBüyüklüğü* defa tekrar edilerek, ilk çözüm uzayı oluşturulur.

```
program GA
  // Giriş: GZRS_Listesi
  // Çıktı: KarşılanabilirİstekListesi

  HerRotaİçinGerekliBantGenişlikleriniBul()
  İlkÇözümleriOluştur(ÇözümUzayıBüyüklüğü)

  for i:1 to DöngüSayısı
    Selection
    CrossOver(OranCO)
    Mutation(OranM)
    CalculateFitnesses(YeniÇözümler)
    ReplaceLeastFit(YeniÇözümlerinSayısı)
  end for

  for each istek r in R
    if r.durum == karşılanabilir
      if (r.RotaÜzerindeKullanılanBantGenişliği ≤
        RotaÜzerindeLinklerinKullanılabilirBantGenişliği) then
        KarşılanabilirİstekListesi += <r, r.Kaynak, r.Hedef, r.Rota,
          r.RotaÜzerindeKullanılanBantGenişliği>;
      end if
    end if
  end for
  return KarşılanabilirİstekListesi;
end program.
```

Şekil 4.10. GA algoritması

Rulet tekerliği (roulette wheel) metodu, genetik algoritmalarda çok kullanılan bir seçme (selection) operasyonudur ve bu çalışmada da benimsenmiştir. CrossOver operasyonu için, *OranCO* olasılığı ile, iki çözüm seçilir. Çözüm kromozomları üzerinde rassal olarak iki nokta belirlenir ve karşılıklı olarak değiştirilir. Mutasyon operasyonunda ise, *OranM* olasılığı ile, rassal olarak bir çözüm ve çözüm üzerinde rassal bir gen seçilir. Genin değeri 1 ise 0, 0 ise 1 yapılır. CrossOver ve Mutation operasyonları sonunda elde edilen kromozomlar, eğer ilk çözüm uzayını oluşturan en kötü çözümlerden daha iyi iseler, çözüm uzayına dâhil edilirler. Kötü olan çözümler de çözüm uzayından çıkarılırlar.

Çözüm için *fayda fonksiyonu* değeri hesaplanırken, çözümdeki isteklerin karşılanması durumunda, herhangi bir linkin bant genişliği kapasitesinin aşılıp

aşılmadığı da incelenir. Eğer en az bir link bant genişliği kapasitesini aşmış ise, fayda fonksiyonu = 0 olur. Eğer hiç bir linkin bant genişliği kapasitesi aşılmamış ise, fayda fonksiyonunun değeri, çözümde bulunan karşılanabilir istek sayısıdır.

4.2.4. Tekli Rotadan Veri Dağıtımı Simülasyon Sonuçları

Simülasyon çalışmalarında, bu çalışmanın 3. Bölüm'ünde anlatılan DGridSim simülatörü kullanılmıştır. Simülasyonlar sırasında, rassal veri erişimi düzeninde, 3. Bölüm'de iyi performans gösteren Rand iş çizelgeleme algoritması; geometrik ve zipf veri erişim düzenlerinde ise MMwDP iş çizelgeleme algoritması kullanılmaktadır. Veri kopyalama algoritması olarak çekme tabanlı MRD algoritması, veri yenileme algoritması olarak LRU algoritması kullanılmaktadır. Bu algoritmalar 5. Bölüm'de detaylandırılacaktır. Simülasyonlarda, 3. Bölüm'de kullanılan parametreler kullanılmıştır.

Bölüm 3'te sunulan temel çalışmalarda kullanılan algoritmaların gösterdikleri en iyi performans sonuçları, Çizelge 4.1'de bulunan SP_MinHop satırında belirtilmiştir. Çizelgede, Grid sistemine arz edilen 10000 adet son zamanlı işin, Bölüm 3'teki simülasyon ortamından, sadece veri dağıtımı algoritması değiştirilerek elde edilen performans değişimleri gösterilmiştir. Performans değerleri, 10000 son zamanlı işten başarı ile tamamlananların yüzdesini göstermektedir.

Çizelge 4.1. Tek aşamalı yaklaşımdaki veri dağıtımı algoritmalarının, sistemin gerçek-zaman performansına etkileri

Veri dağıtımı algoritması	**Rassal**		**Geometrik**		**Zipf**	
	Fed.	**Hiye.**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**
SP_MinHop (Bölüm 3'deki sonuçlar)	61	56	65	96	60	66
SP_MinDelay	62	57	68	97	63	68
SP_MinMin	62	57	66	97	62	67
SP_MinCon	61	56	67	96	61	67

Çizelge 4.1'deki sonuçlar, rassal, geometrik ve zipf veri erişim düzenleri ile federatif ve hiyerarşik veri organizasyon modeline sahip sistemlerin tamamında SP_MinDelay tek rotadan tek aşamalı veri dağıtımı algoritmasının en iyi performansa sahip olduğunu göstermektedir.

10000 gerçek-zamanlı işin simülasyonu için gerekli ortalama zamanlar ise Çizelge 4.2'de gösterilmiştir. Elde edilen sonuçlara göre SP_MinDelay veri dağıtımı algoritması, diğer algoritmalara göre daha kısa zamanda sonuçlanmaktadır. SP_MinHop algoritması, yakın sonuçlar göstermektedir.

Çizelge 4.2. Tek aşamalı yaklaşımdaki veri dağıtımı algoritmalarının ortalama çalışma zamanları

Veri dağıtımı algoritması	**Rassal**		**Geometrik**		**Zipf**	
	Fed.	**Hiye.**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**
SP_MinHop (Bölüm 3'deki sonuçlar)	156	684	523	490	572	452
SP_MinDelay	148	645	519	476	565	437
SP_MinMin	180	872	562	486	602	480
SP_MinCon	184	902	544	503	614	484

İki aşamalı veri dağıtımı algoritmalarının performans ölçümleri ise, Çizelge 4.3'te gösterilmiştir. İlk aşamada hangi rotaların kullanılacağını seçen GZRS algoritmaları, ikinci aşamada ise hangi isteklerin karşılanacağını seçen GZİS algoritmaları bulunmaktadır.

Çizelge 4.3. İki aşamalı yaklaşımdaki veri dağıtımı algoritmalarının, sistemin gerçek-zaman performansına etkileri

Veri dağıtımı algoritmaları		**Rassal**		**Geometrik**		**Zipf**	
GZRS	**GZİS**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**
MinHop/FPF	IO_UFF	42	41	50	78	48	44
MinHop/FPF	MNOFF	53	48	60	86	55	55
MinHop/FPF	MOFF	57	51	62	88	57	62
MinHop/FPF	GA	46	47	54	81	50	48
MinDelay/FPF	IO_UFF	45	43	55	81	50	46
MinDelay/FPF	MNOFF	56	50	61	88	60	58
MinDelay/FPF	MOFF	61	53	65	93	62	65
MinDelay/FPF	GA	50	49	59	84	54	51
MinMin/FPF	IO_UFF	43	44	52	78	48	45
MinMin/FPF	MNOFF	53	50	60	87	59	58
MinMin/FPF	MOFF	58	52	63	91	62	64
MinMin/FPF	GA	49	49	57	83	53	51
MinCon/FPF	IO_UFF	45	42	53	80	49	46
MinCon/FPF	MNOFF	55	49	60	87	60	57
MinCon/FPF	MOFF	60	52	64	92	62	65
MinCon/FPF	GA	48	49	57	84	52	50

Çizelge 4.3'te de gösterildiği gibi, rassal, geometrik ve zipf veri erişim düzenlerinde federatif ve hiyerarşik veri organizasyonu modellerinde gerçek-zamanlı rota seçimi algoritmalarından MinDelay/FPF, genellikle, en başarılı sonuçları verirken; onu MinCon/FPF, MinMin/FPF ve MinHop/FPF algoritmaları takip etmektedir. Son üç algoritma, rota seçiminde atlama sayısına daha fazla önem vermektedirler. Sonuçlar göstermektedir ki, gerçek-zaman performansları açısından, rota seçiminde linklerin gecikmeleri mutlaka dikkate alınmalıdır.

Gerçek-zamanlı istek seçimi algoritmalarından ise MOFF, diğer algoritmalara göre daha başarılı sonuçlar göstermektedir. MOFF algoritmasını, sırası ile, MNOFF, GA ve IO_UFF algoritmaları takip etmektedir.

Çizelge 4.4. İki aşamalı yaklaşımdaki veri dağıtımı algoritmalarının ortalama çalışma zamanları

Veri dağıtımı algoritmaları		**Rassal**		**Geometrik**		**Zipf**	
GZRS	**GZİS**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**
MinHop/FPF	IO_UFF	583	892	748	713	898	883
MinHop/FPF	MNOFF	498	832	743	720	845	804
MinHop/FPF	MOFF	458	783	694	680	795	774
MinHop/FPF	GA	1232	1675	1564	1467	1756	1676
MinDelay/FPF	IO_UFF	559	862	752	784	876	845
MinDelay/FPF	MNOFF	418	794	702	689	805	767
MinDelay/FPF	MOFF	390	761	679	658	769	735
MinDelay/FPF	GA	1192	1580	1378	1326	1542	1567
MinMin/FPF	IO_UFF	595	895	779	695	845	794
MinMin/FPF	MNOFF	490	805	724	673	787	757
MinMin/FPF	MOFF	453	760	671	648	764	736
MinMin/FPF	GA	1189	1583	1490	1389	1698	1684
MinCon/FPF	IO_UFF	591	893	753	712	902	872
MinCon/FPF	MNOFF	487	807	750	725	820	793
MinCon/FPF	MOFF	433	753	686	672	784	740
MinCon/FPF	GA	1342	1456	1459	1402	1808	1589

Çizelge 4.4'te gösterilen ortalama simülasyon zamanlarına da bakıldığında, MinDelay/FPF algoritması kullanılarak yapılan simülasyonlar, diğer GZRS algoritmalarına göre daha az zaman almıştır. GZİS algoritmaları arasından da en

iyi performansı MOFF almıştır. IO_UFF algoritmasında, işlerin başarı ile tamamlanma oranı düşüktür. IO_UFF'ta işler, daha uzun süre çizelgeleme sırasında kaldığı için simülasyonun tamamlanması için geçen zaman fazladır.

İki aşamalı yaklaşımda en başarılı sonuçları veren MinDelay/FPF rota seçimi ile MOFF istek seçimi algoritmaları birlikte kullanıldığında bile, SP_MinDelay tek aşamalı yaklaşımında elde edilen başarı oranları yakalanamamaktadır. Ayrıca, SP_MinDelay kullanılarak yapılan simülasyonlar da daha az zaman almaktadır. Dolayısı ile, tekli rotadan veri dağıtımı algoritmaları arasında yapılacak en iyi seçim SP_MinDelay olarak ortaya çıkmaktadır.

4.3. Gerçek-Zamanlı Çoklu Rotadan Veri Dağıtımı

4.3.1. Problem Tanımı

Önceki bölümde Gerçek-Zamanlı Tekli Rotadan Veri Dağıtımı Problemi (RTU/DDP) tanımlanmıştır. Problemin NP-zor olduğunun gösterilmesi amacı ile, RTU/DDP probleminin biçimsel formülasyonu da sunulmuş ve problemin NP-zor olduğu kanıtlanmıştır. Bu bölümde, RTU/DDP probleminin genelleştirilmiş hali olan Gerçek-Zamanlı Çoklu Rotadan Veri Dağıtımı Problemi tanımlanacak ve problemin çözümü için keşifsel algoritmalar önerilecektir.

Tanımlama 6: Veri Grid sistemi $G = (V, E)$ ve gerçek-zamanlı veri transferi istekleri R verilmiş olsun, gerçek-zamanlı çoklu rotadan veri dağıtımı problemi (Real-Time Splittable Data Dissemination Problem - RTS/DDP), veri transferlerinin k rotadan gönderildiği durumda, karşılanan veri transferi sayısını en yükseğe çıkarmaya çalışır.

$$\text{RTS/DDP: azami} \sum_{r_i \in R} \sum_{p_j \in P_i} x_{ij}$$

$$\sum_{p_j \in P_i} x_{ij} = \{0,1\}, \text{ tüm } r_i \epsilon R \text{ için}$$

$$\sum_{p_j \in P_i} \lceil x_{ij} \rceil \leq k, \text{ tüm } r_i \epsilon R \text{ için}$$

$$\sum_{r_i \in R} \sum_{p_j \in P_i} \xi_{lji} \pi_{ij} x_{ij} \leq c_l, \text{ tüm } e_l \epsilon E \text{ için}$$

$$0 \leq x_{ij} \leq 1, \text{ tüm } r_i \epsilon R \text{ ve } p_j \epsilon P_i \text{ için}$$

Eğer link $e_l \epsilon E$, istek $r_i \epsilon R$'yi karşılayan rota $p_j \epsilon P_i$ üzerinde ise ξ_{tji} 1, aksi durumda ise 0'dır. İstek $r_i \epsilon R$'nin rota $p_j \epsilon P_i$ üzerinden karşılanması için

$$\pi_{ij} = \frac{|f_i|}{d_i - \sum_{e_l \in p_j} \delta_l}$$

büyüklüğünde akışa ihtiyaç duyulur.

Teorem 3: RTS/DDP, NP-zor bir problemdir.

İspat: k=1 iken RTS/DDP, RTU/DDP'ye dönüşmektedir.

Gerçek-zamanlı tekli rotadan veri dağıtımı probleminde en iyi sonuçları, gerçeklenmesi de kolay olan SP_MinDelay algoritması vermiştir. Gerçek-zamanlı çoklu rotadan veri dağıtımı problemi için keşifsel algoritmalar önerilirken SP_MinDelay algoritması baz alınacaktır. RTS/DDP probleminin çözümü için iki temel problemin çözüme kavuşturulması gerekmektedir. Birinci problem, veri transferlerinin gerçekleştirileceği rotaların seçimi; ikinci problem ise, hangi rotaya ne kadar akışın paylaştırılacağıdır.

4.3.2. *k*-Adet Gerçek-Zamanlı Rota Seçimi

RTS/DDP probleminin çözümü için öncelikle verinin gönderileceği k adet rotanın bulunması gereklidir. Her bir rotanın bulunması sırasında, GZRS

algoritmalarından herhangi biri kullanılabilir. Bu bölümde, RTU/DDP'nin çözümünde, en iyi performansı gösteren en az gecikme tabanlı rota bulma algoritmaları kullanılacaktır. Bu çalışmada *k*-adet gerçek-zamanlı rota seçimi *(k*GZRS) için iki keşifsel algoritma önerilmiştir: *k*-en kısa yol, *k*-en kısa ayrık yol.

4.3.2.1. *k*-en kısa yol

Şekil 4.11'de gösterilen *k*-en kısa yol (k-Shortest Path - *k*SP) algoritması ile, her istek için gerekli olan *k* adet kısa yol bulunur. Bu rotalar, her istek *r* için *r.Rotalar* veri yapısında tutulurlar. Ayrıca her *r.Rotalar*'daki her rota için, isteğin sadece o rota üzerinde dağıtılması durumunda gerekli olacak bant genişliği olan *r.RotalarÜzerindeKullanılanBantGenişlikleri* değeri hesaplanır. Söz konusu değer hesaplanırken, isteğin o rota üzerinde karşılanması durumunda oluşacak toplam link gecikmeleri *RotaÜzerindekiLinklerinGecikmeToplamı* değeri kullanılır.

Bu çalışmada $rota_i$'den olası tüm sapmaları bulmak için şu işlemler yapılmıştır:

1. $rota_i$'de bulunan link sayısı *n* olsun.
2. j=1'den *n*'e kadar aşağıdaki işlemi tekrar et.
3. i'ninci en az gecikmeli yolun j'inci linkini sil ve kalan linkler ile en az gecikmeli yolu bul. Eğer yol bulunmuş ise, yolu olası bir sapma olarak kaydet. Silinen linki tekrar ilave et.

Her en az gecikmeli yolun bulunması O(|V||E|) zaman alır. En kötü durumda $rota_i$'de |E| adet link bulunur. Dolayısı ile tüm olası sapmalar O(|V||E|^2)'de bulunur. Bu durumda *k*SP algoritması O(k|V||E|^2)'de çalışır.

***k*SP algoritması**
// Giriş: *G(V,E),* istek kümesi *R*, her istek için bulunacak rota sayısı *k*
// Çıkış: *kGZRS_Listesi*

for each istek *r* **in** *R*
GerekliBantGenişliği = $|f_r|/d_r$;
tempG = *G*;
tempG'den *GerekliBantGenişliği*'ne sahip olmayan linkleri sil;
tempG'de kalan linkler ile *r.Hedef*'ten Dijkstra en kısa yol algoritmasını kullanarak diğer veri depolama elemanlarına en az gecikmeli yolları bul;
r.Kaynak = en yakın veri depolama elemanı;
r.Rotalar += (*r.Kaynak*, *r.Hedef*) arasında *tempG* üzerinde en az ağırlıklı rota;
r.RotalarÜzerindeKullanılanBantGenişlikleri += <$|f_r|/(d_r$ – *RotaÜzerindekiLinklerinGecikmeToplamı*)>;
for i=1:*k*-1
B = boş küme;
$rota_i$'den tüm sapmaları oluştur ve B kümesine ilave et;
r.Rotalar += $rota_{i+1}$ = B kümesindeki rotalar arasında en kısa yol;
r.RotalarÜzerindeKullanılanBantGenişlikleri += $|f_r|/(d_r$ – *RotaÜzerindekiLinklerinGecikmeToplamı*);
end for
kGZRS_Listesi += <*r*, *r.Kaynak*, *r.Hedef*, *r.Rotalar,* *r.RotalarÜzerindeKullanılanBantGenişlikleri*>;
end for
return *kGZRS_Listesi*;
end program.

Şekil 4.11. *k*SP algoritması

4.3.2.2. *k*-en kısa ayrık yol

Önceki bölümde tanıtılan *k*-en kısa yol algoritması kullanılan rotalardaki link adedini düşürse de, bazı linklerin diğerlerine göre daha fazla kullanıldığı durumlar olabilir. Linklerin kullanım oranlarının mümkün olduğu kadar dengeli olabilmesi için *k*-en kısa ayrık yol (*k*-shortest Disjoint Paths - *k*DP) yaklaşımı kullanılabilir. Bu yaklaşım, özellikle ağ yapısındaki bozulmalarda, sistemi çalışır hale getirme aşamasında kullanılır [138]. *k*DP algoritması, Şekil 4.12'de gösterilmiştir.

Şekilde gösterildiği üzere, k adet rotayı bulmak için, öncelikle en kısa rota bulunur. Rota üzerindeki linkler, ağ topolojisinden silinir. Geri kalan linkler ile en kısa rota bulunmaya çalışılır. İşlem, bulunan rota üzerindeki linklerin silinmesi ve

geri kalan linkler kullanılarak rotaların bulunması şeklinde, *k* adet rota bulunana kadar devam eder. Her istek *r* için bulunan *r.Rotalar* için *r.RotalarÜzerindeKullanılanBantGenişlikleri* de veri yapısında saklanır.

En az gecikmeli yol algoritması O(|V||E|)'de çalışır. Bu durumda *k*DP algoritması O(k|V||E|)'de çalışır.

***k*DP algoritması**
// Giriş: *G(V,E),* istek kümesi *R*, her istek için bulunacak rota sayısı *k*
// Çıkış: *kGZRS_Listesi*

for each istek *r* **in** *R*
r.GerekliBantGenişliği = $|f_r|/d_r$;
tempG = *G*;
tempG'den *GerekliBantGenişliği*'ne sahip olmayan linkleri sil;
tempG'de kalan linkler ile *r.Hedef*'ten Dijkstra en kısa yol algoritmasını kullanarak diğer veri depolama elemanlarına en az gecikmeli yolları bul;
r.Kaynak = en yakın veri depolama elemanı;
r.Rotalar += (*r.Kaynak*, *r.Hedef*) arasında *tempG* üzerinde en az ağırlıklı rota;
r.RotalarÜzerindeKullanılanBantGenişlikleri += $|f_r|/(d_r -$ *RotaÜzerindekiLinklerinGecikmeToplamı*);
for i=1:*k*-1
$rota_i$'de bulunan tüm linkleri *tempG*'den sil;
r.Rotalar += $rota_{i+1}$= *tempG*'yi kullanarak Dijkstra en kısa yol algoritmasını kullanarak en az gecikmeli yol;
r.RotalarÜzerindeKullanılanBantGenişlikleri += $|f_r|/(d_r -$ *RotaÜzerindekiLinklerinGecikmeToplamı*);
end for
kGZRS_Listesi += <*r*, *r.Kaynak*, *r.Hedef*, *r.Rotalar,* *r.RotalarÜzerindeKullanılanBantGenişlikleri*>;
end for
return *kGZRS_Listesi*;
end program.

Şekil 4.12. *k*DP algoritması

4.3.3. Gerçek-Zamanlı Akış Paylaşımı

Verinin gönderileceği rotalar ve isteğin sadece o rota üzerinde karşılanması durumunda gerekli olacak olan bant genişlikleri bulunduktan sonra, veri transferi isteğinin başarılı biçimde gerçekleştirilebilmesi için hangi rotaya ne kadarlık akış tahsis edilmesi gerektiği problemi çözülmelidir. *k*GZRS algoritmalarından biri

kullanılarak her istek için *k*-adet rota bulunduğunu varsayan Gerçek-Zamanlı Akış Paylaşımı (GZAP), karşılanan istek sayısını en fazlaya çıkartacak şekilde akışların tahsis edilmesi problemidir.

4.3.3.1. Eşit bölüşümlü çoklu rotadan dağıtım

Eşit bölüşümlü çoklu rotadan dağıtım (Equal Share Multi Path - ESMP) algoritması Şekil 4.13'te gösterilmiştir. ESMP, en fazla *k* rotayı kullanarak isteğin karşılanması için, öncelikle sıradaki veri transferi isteğini tek rotadan dağıtmaya çalışır. Tek rotadan gönderim başarısız ise, eşit miktarda veri transferi yapan iki rotadan sağlamaya çalışır. Bu durumda, *kGZRS_Listesi*'nde bulunan ve *r.Rotalar* veri yapısında saklanan her rota için *r.RotaOranlar* değerleri 0.5 olur. Rota üzerinde kullanılan bant genişliği, ilgili rota için belirtilmiş olan *r.RotalarÜzerindeKullanılanBantGenişlikleri* değeri ile *r.RotaOranlar* değerlerinin çarpılması neticesinde bulunur. İsteğin, iki rota üzerinden karşılanamaması durumunda, üç rota üzerinden karşılanıp karşılanmadığı kontrol edilir. Bu işleme, en fazla *k* rotaya kadar tekrar edilir. Eğer *k* rotada da gönderim başarılı olmaz ise istek *karşılanamaz* olarak işaretlenir.

```
ESMP algoritması
  // Giriş: kGZRS_Listesi
  // Çıkış: KarşılanabilirİstekListesi
for each istek r in R
    r.durum = karşılanamaz;
    i=1;
    while i≤k
      Herbir rotadan VeriBüyüklüğü/i'yi son zamanından önce göndermek için
        gerekli bant genişliğini bul;
      Herbir rotadaki linklerin kullanılan bant genişliğini arttır;
      if linklerden herhangi birisi kapasitesini aşmış ise then
        Linklerde arttırılan tüm bant genişliklerini azalt;
        i=i+1;
        continue;
      else
        if (r.RotalarÜzerindeKullanılanBantGenişlikleri ≤
          RotaÜzerindeLinklerinKullanılabilirBantGenişliği) then
          KarşılanabilirİstekListesi += <r, r.Kaynak, r.Hedef, r.Rotalar,
            r.RotalarÜzerindeKullanılanBantGenişlikleri>;
          break;
        end if
      end if
    end while
  end for
  return KarşılanabilirİstekListesi;
end program.
```

Şekil 4.13. ESMP algoritması

4.3.3.1. Dengeli bölüşümlü çoklu rotadan dağıtım

ESMP algoritmasında, her rotadan eşit miktarda verinin transfer edildiği varsayılmakta idi. Ancak, daha fazla kullanılabilir bant genişliğine sahip olan rotalardan daha fazla miktarda verinin gönderilmesi daha verimli olacaktır. Şekil 4.14'te gösterilen dengeli bölüşümlü çoklu rotadan dağıtım (Balanced Share Multi Path - BSMP) algoritmasında ise, rotalardan gönderilecek olan verinin büyüklüğü, kullanılabilir bant genişlikleri ile doğru orantılı olarak paylaştırılmıştır.

```
BSMP algoritması
  // Giriş: kGZRS_Listesi
  // Çıkış: KarşılanabilirİstekListesi
for each istek r in R
   r.durum = karşılanamaz;
   i=1;
   while i≤k
    Her j (j<i) rotası için KBW_j değerlerini, kullanılabilir bant genişlikleri, bul;
    i rotadaki kullanılabilir bant genişliklerinin toplamı olan TKBW değerini bul;
    Herbir rotadan KBW_j*VeriBüyüklüğü/ TKBW'yi son zamanından önce
      göndermek için gerekli bant genişliğini bul;
    Herbir rotadaki linklerin kullanılan bant genişliğini arttır;
    if linklerden herhangi birisi kapasitesini aşmış ise then
      Linklerde arttırılan tüm bant genişliklerini azalt;
      i=i+1;
      continue;
    else
      if (r.RotalarÜzerindeKullanılanBantGenişlikleri ≤
        RotaÜzerindeLinklerinKullanılabilirBantGenişliği) then
        KarşılanabilirİstekListesi += <r, r.Kaynak, r.Hedef, r.Rotalar,
          r.RotalarÜzerindeKullanılanBantGenişlikleri>;
        break;
      end if
    end if
   end while
  end for
  return KarşılanabilirİstekListesi;
end program.
```

Şekil 4.14. BSMP algoritması

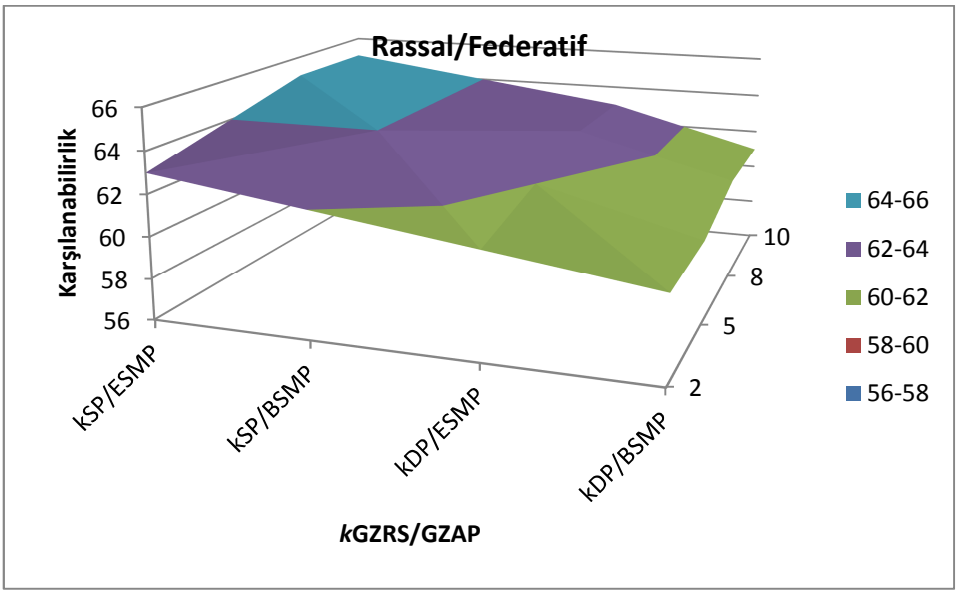
4.3.4. Çoklu Rotadan Veri Dağıtımı Simülasyon Sonuçları

Şekil 4.15-4.20'de, çoklu rotadan veri dağıtımı yapıldığı durumda, veri dağıtımı algoritmalarının, gerçek-zaman performansı üzerindeki etkisi gösterilmiştir. Karşılaştırmalarda, çoklu rotadan veri gönderiminin, tekli rotadan veri gönderimine göre performans artışları gösterilmiştir.

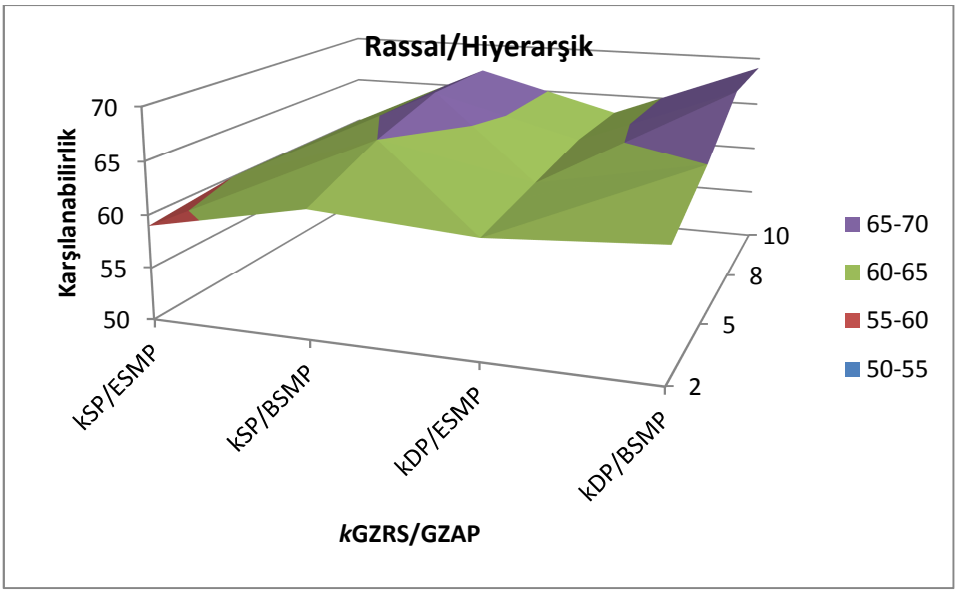
Şekil 4.15, 4.17 ve 4.19'da gösterildiği gibi, federatif veri organizasyon modeline uyan sistemlerde çoklu rotadan gerçek-zamanlı rota seçimi algoritmalarından *k*-en kısa yol algoritması, *k*-en kısa ayrık yol algoritmasına göre daha iyi performans göstermektedir. *k*GZİS algoritmalarından eşit bölüşümlü

çoklu rotadan gönderim, dengeli bölüşümlü çoklu rotadan gönderim algoritmasına göre daha başarılı sonuçlar vermektedir. Çoklu rotadan gönderimde, federatif modele sahip bir sistemde (*k*SP, ESMP) ikilisi, Bölüm 3'teki baz sonuçlara göre, tekli rotadan veri gönderimine göre yaklaşık % 4'e varan iyileştirme sağlayabilmektedir.

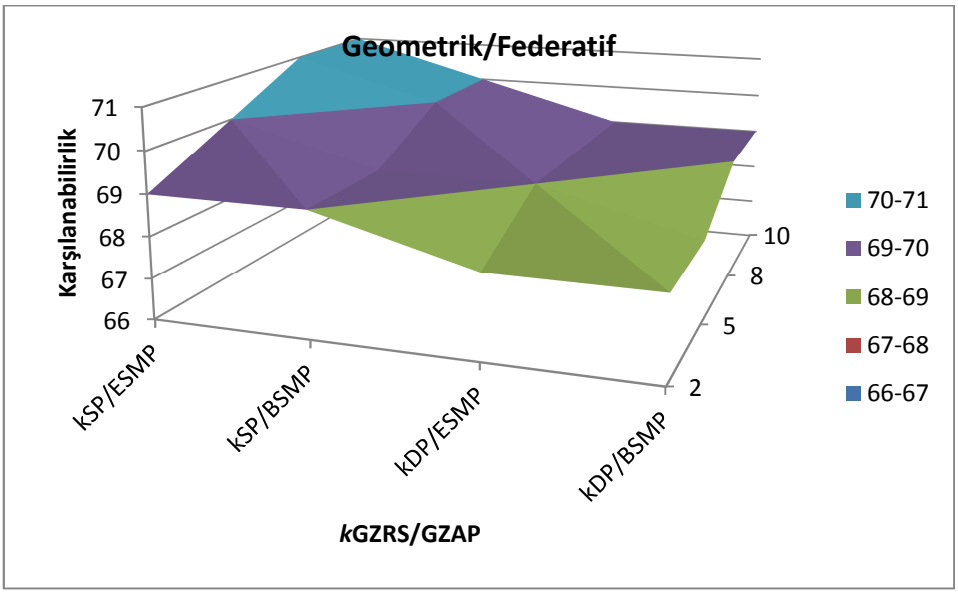
Şekil 4.16, 4.18 ve 4.20'de gösterildiği gibi, hiyerarşik veri organizasyon modeline uyan sistemlerde çoklu rotadan gerçek-zamanlı rota seçimi algoritmalarından *k*DP algoritması, *k*SP algoritmasına göre daha iyi performans göstermektedir. *k*GZİS algoritmalarından da dengeli bölüşümlü çoklu rotadan gönderim, eşit bölüşümlü çoklu rotadan gönderim algoritmasına göre daha başarılı sonuçlar vermektedir.



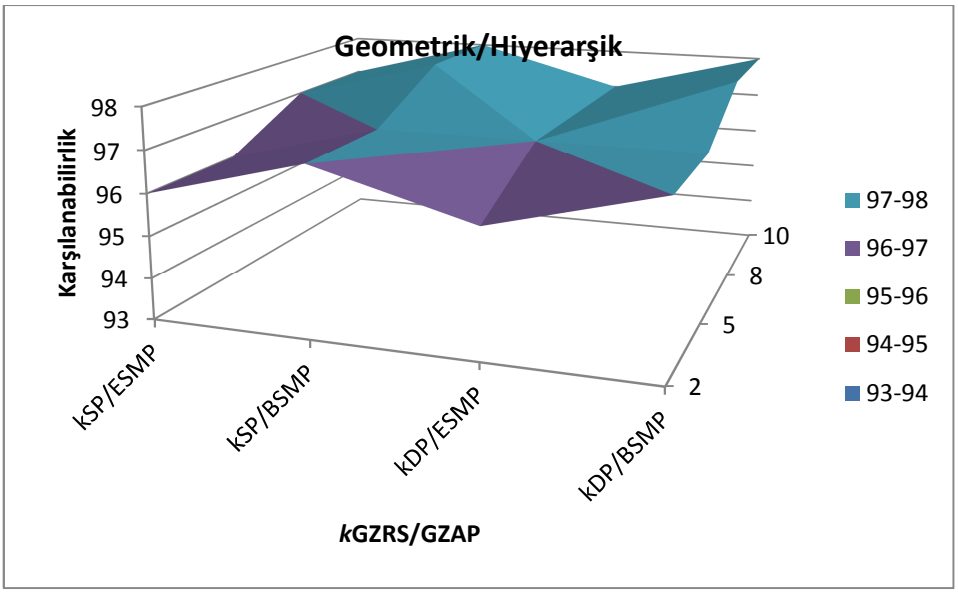
Şekil 4.15. Gerçek-zamanlı çoklu rotadan veri dağıtımı algoritmalarının, rassal veri erişimi düzeni ve federatif veri organizasyon modeline uygun Veri Grid sisteminin gerçek-zaman performansına etkileri



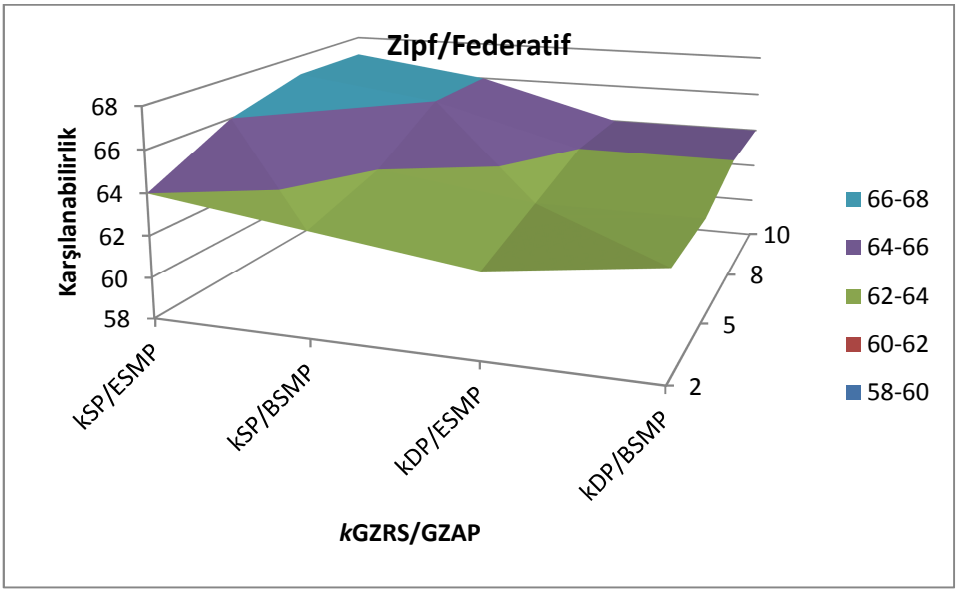
Şekil 4.16. Gerçek-zamanlı çoklu rotadan veri dağıtımı algoritmalarının, rassal veri erişimi düzeni ve hiyerarşik veri organizasyon modeline uygun Veri Grid sisteminin gerçek-zaman performansına etkileri



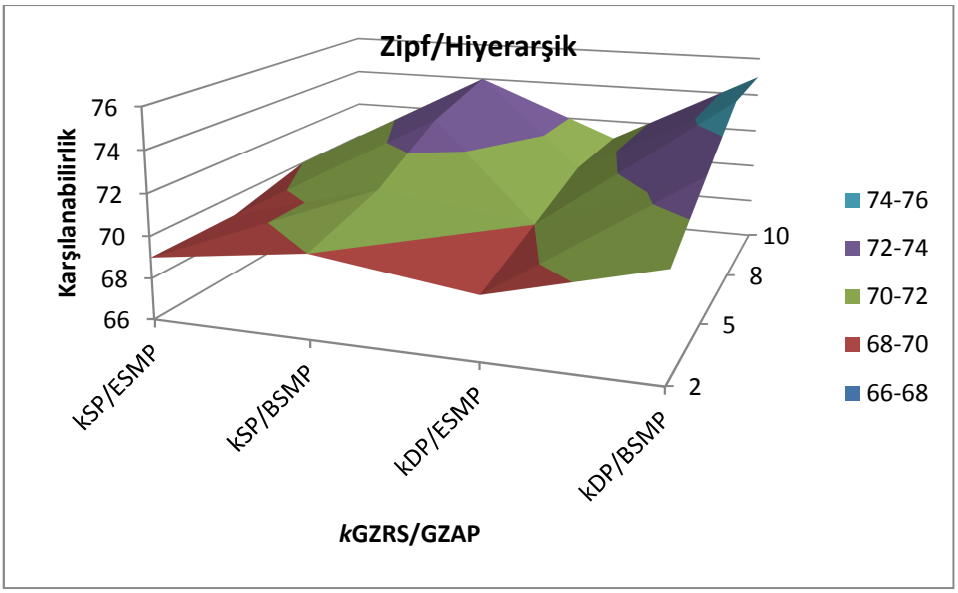
Şekil 4.17. Gerçek-zamanlı çoklu rotadan veri dağıtımı algoritmalarının, geometrik veri erişimi düzeni ve federatif veri organizasyon modeline uygun Veri Grid sisteminin gerçek-zaman performansına etkileri



Şekil 4.18. Gerçek-zamanlı çoklu rotadan veri dağıtımı algoritmalarının, geometrik veri erişimi düzeni ve hiyerarşik veri organizasyon modeline uygun Veri Grid sisteminin gerçek-zaman performansına etkileri



Şekil 4.19. Gerçek-zamanlı çoklu rotadan veri dağıtımı algoritmalarının, zipf veri erişimi düzeni ve federatif veri organizasyon modeline uygun Veri Grid sisteminin gerçek-zaman performansına etkileri



Şekil 4.20. Gerçek-zamanlı çoklu rotadan veri dağıtımı algoritmalarının, zipf veri erişimi düzeni ve hiyerarşik veri organizasyon modeline uygun Veri Grid sisteminin gerçek-zaman performansına etkileri

Çoklu rotadan gönderimde, hiyerarşik modele sahip bir sistemde (*k*DP, BSMP) ikilisi, Bölüm 3'teki baz sonuçlara göre, tekli rotadan veri gönderimine göre yaklaşık % 12'ye varan iyileştirme sağlayabilmektedir. Hiyerarşik modelde sağlanabilecek iyileştirmeler, federatif modele göre daha yüksektir. Bunun temel sebebi Hiyerarşi-0 sitesine bağlı linklerdeki tıkanıklıkların büyük ölçüde azaltılmasıdır.

5. VERİ KOPYALAMA

Veri Grid sistemlerine arz edilen işler, çok büyük miktarlardaki veriye ihtiyaç duyabilirler. İşlerin ihtiyaç duyduğu veriler, genellikle, farklı Grid siteleri üzerinde dağıtılmış vaziyettedirler. Diğer taraftan, verilerin, kendilerine ihtiyaç duyan uygulamalara yakın olması, Veri Grid sisteminin performansını önemli ölçüde arttırır. Verilerin farklı sitelerde kopyalanmasının, linklerdeki bant genişliği ve depolama elemanlarındaki veri depolama alanı gereksinimlerini önemli ölçüde azalttığına dair bir çok çalışma bulunmaktadır [49-52, 54, 56-58]. Son zamanlarda yapılmış olan bir çalışma [53], uygulamaların Veri Grid sisteminde son zamanlarından önce tamamlanmasında, veri kopyalama algoritmalarının olumlu etkisinin olduğunu göstermiştir.

Veri kopyalama algoritmaları temel olarak iki soru üzerinde çözüm üretmeye çalışırlar: Hangi veriler kopyalanmalıdır? Kopyalanacak veriler hangi siteye veya sitelere kopyalanmalıdır? Bu süreç sırasında da ihtiyaç duyulan veri depolama alanı ve ağ üzerine binecek ek yükü de en aza indirmeye çalışırlar.

Veri kopyalama algoritması tarafından kopyalanmasına karar verilen verileri alacak yeterli depolama alanı, hedef Grid sitesinin veri depolama elemanları üzerinde bulunmayabilir. Bu durumda, bir veya birden fazla verinin, depolama elemanından silinmesi gerekir. Hangi verilerin silinmesi gerektiğine de *veri yenileme algoritma*ları karar verir.

5.1. İlgili Çalışmalar

Literatürdeki veri kopyalama algoritmaları iki gruba ayrılabilirler: çekme tabanlı algoritmalar [50, 51, 54], itme tabanlı algoritmalar [49, 52, 56-58]. Çekme tabanlı modelde, kendi yerel veri depolama elemanlarında istekte bulunulan veri bulunmayan site, veriyi yerel olarak kopyalayıp kopyalamayacağına ve hangi kaynak siteden kopyalanacağına karar verir. İtme tabanlı modelde ise, siteler kendi yerel depolama elemanlarında bulunan verilerden hangilerini, ne zaman ve hangi hedef sitelerde kopyalaması gerektiğine karar verirler. Çekme ve itme

tabanlı veri kopyalama algoritmaları da ayrıca dağıtık [49-51, 54, 56] ve merkezi [52, 57, 58] olarak ikiye ayrılabilir.

[50]'de, LFU ile birlikte iki adet ekonomi bazlı algoritma olmak üzere üç farklı veri kopyalama yaklaşımı incelenmiştir. Tüm bu yaklaşımlarda, tüm sitelerin veri kopyalama algoritmaları çalıştırmalarını gerektiren, dağıtık bir yapı öngörülmüştür. Bir veri transferi isteği, yerel olarak Kopya Yöneticisi tarafından sağlanamıyor ise; LFU, istekte bulunulan veriyi, uzak Grid sitesinden kopyalar ve eğer gerekiyor ise, son zamanlarda en az sıklıkla kullanılan veriyi, depolama elemanından siler. Ekonomik yaklaşımlar ise, depolama elemanındaki tüm verilere ve istekte bulunulan veriye bir değer tahmin fonksiyonu yardımı ile değer takdir ederler. Eğer istekte bulunulan veri, depolama elemanındaki en az değerli veriden daha değerli ise, veri kopyalanır. Eğer Grid sistemi üzerinde, istekte bulunulan veri birden fazla sitede mevcut ise, ekonomik yaklaşımlar, bir müzayede protokolü ile, hangi sitedeki verinin kopyalanacağına karar verirler. LFU algoritması ise, böyle bir durumda, en az erişim sıklığı değerine sahip olan veriyi transfer eder. Bu üç algoritmanın dışında, [51]'de, depolama elemanında en uzun zamandır kullanılmayan veriyi silen LRU algoritması da tanımlanmıştır. [51]'de, LRU ve LFU stratejilerinin, ekonomik modeli kullanan yaklaşımlara göre yüzde 20-30 daha iyi performans sağladıkları gösterilmiştir.

[54]'te, başka bir çekme tabanlı veri kopyalama algoritması tanıtılmıştır. Bu çalışmada, Grid sistemi üzerindeki veriler, ağaç ve halka topolojilerinin karışımı olan hibrit bir yapıda dağıtılmıştır. Her sitede bulunan bir Kopya Yöneticisi, sitedeki tüm veriler için, ekonomik modellerde olduğu gibi bir maliyet fonksiyonu hesaplar. Veriler, hesaplanan veri transfer maliyetinde bir iyileşmeye neden olacak ise siteye kopyalanır.

[49]'da üç farklı itme tabanlı dağıtık veri kopyalama algoritması tanıtılmıştır: DataRandom, DataLeastLoaded ve DataRandLeastLoaded. Bu algoritmalar, Grid sistemi üzerindeki tüm sitelerde bulunan bir çizelgeleyici üzerinde çalışırlar. DataRandom algoritması, sitede bulunan çok istekte bulunulan verileri, Grid sistemi üzerinde bulunan diğer bir siteye rassal olarak gönderir. DataLeastLoaded algoritması, en az yüklü olan siteye; DataRandLeastLoaded algoritması ise, en az yüklü olan siteler arasında rassal bir siteye verileri gönderir.

[49]'da bu algoritmaların, farklı iş çizelgeleme algoritmaları ile birlikte çalışırken, yakın sonuçlar verdiği rapor edilmiştir. [56]'da aynı araştırmacılar, beş farklı itme tabanlı dağıtık veri kopyalama algoritması daha önermişlerdir.

[57]'de, Basit Aşağıdan Yukarı (Simple Bottom-Up) ve Birleştirilmiş Aşağıdan Yukarı (Aggregate Bottom-Up) algoritmaları tanıtılmıştır. Her iki algoritma da itme tabanlı merkezi veri dağıtımı algoritmalarıdır. Grid sistemi üzerinde bulunan Dinamik Kopyalama Çizelgeleyicisi bileşeni, bu algoritmaları belirli aralıklar ile çalıştırır, kopyalanması gereken verileri ve kopyalanacağı siteleri seçer. Daha sonrasında ise, Grid sitelerinde bulunan Yerel Kopya Yöneticileri, kopyalama işlemini yürütürler. Birleştirilmiş Aşağıdan Yukarı algoritmasının, Basit Aşağıdan Yukarı algoritmasına göre daha iyi sonuçlar verdiği rapor edilmiştir. [52]'de, aynı araştırmacılar, CDR_RTPlace ve CDR_SMPlace isimli iki farklı itme tabanlı merkezi veri kopyalama algoritması daha önermişlerdir. Farklı iş çizelgeleme algoritmaları ile yaptıkları testler sonucunda, CDR_RTPlace algoritmasının CDR_SMPlace algoritmasına gore, işlerin tamamlanma süreleri açısından daha iyi sonuçlar gösterdiğini belirtmişlerdir.

[58]'de, ortanca (*p*-median) bazlı dinamik itme tabanlı merkezi bir veri kopyalama algoritması önerilmiştir. Çalışmada belirtilen *p*-median modelinde, istekte bulunan siteler ile veriyi barındıran siteler arasındaki toplam mesafeyi en aza indirecek *p* tane kopyanın yerleştirileceği siteler bulunur. Çalışmadaki *p*-median problemi, NP-zor bir problem olduğu için, gevşetilmiş Langrange yöntemi (Lagrangean Relaxation) teknikleri kullanılarak bir kopya yerleştirme algoritması sunulmuştur.

Bahsi geçen tüm veri kopyalama algoritmalarında, veri istekleri için ortalama karşılama zamanı, ortalama bant genişliği kullanımı, kopyalama frekansı gibi en-iyi-girişim performans ölçütleri kullanılmıştır. Belirtilen sonuçlar, en-iyi-girişim uygulamalar için yeterli olsalar bile, gerçek-zamanlı uygulamalar için yeterli değillerdir. Bu bölümde, gerçek-zaman gereksinimlerine duyarlı veri kopyalama algoritmaları geliştirilecek ve bu algoritmaların gerçek-zaman performansları karşılaştırılacaktır.

5.2. Veri Kopyalama Algoritmaları

2. Bölüm'de anlatılan Veri Grid sistemi modelinde, Veri Grid sistemleri için veri kopyalama modeli olarak, çekme tabanlı, itme tabanlı-dağıtık ve itme tabanlı-merkezi olmak üzere üç farklı modelden birisi kullanılabilmektedir. Her bir modele uygun olarak, hem federatif hem de hiyerarşik veri organizasyon modelinde kullanılabilecek veri kopyalama algoritmaları geliştirilmiş ve bu algoritmalar aşağıdaki bölümlerde sunulmuştur.

5.2.1. Çekme Tabanlı Veri Kopyalama

Çekme tabanlı veri kopyalama için En Fazla İstek Alan Veri (Most Requested Data – MRD) algoritması Şekil 5.1'de gösterildiği gibi gerçeklenmiştir. MRD algoritması çevrim içi çalışır ve 2. Bölüm'de anlatılan LDRS servisi bünyesinde gerçeklenir. LDRS servisi, ait olduğu sitenin diğer sitelerden istemiş olduğu her veri için, veriye yapılan istek sayısı, veriye yapılan isteklerin son zamanı erişim istatistiklerini takip eder. LDRS, LDMS'den *VERİ_İSTEĞİ_GELDİ* := [*TransferEdilecekVeriListesi* = *İşVeriListesi*, *İstekSonZamanı* = *TransferSonZamanı*] mesajı alır. LDRS, *İşVeriListesi* ve *TransferSonZamanı* bilgilerini, çekme tabanlı bir veri kopyalama algoritması olan MRD'ye iletir. MRD'den de VeriKopyalamaListesi'ni alır. LDRS, LDMS'ye *VERİ_KOPYALAMA_CEVAP* := [*KopyalanacakVeriListesi* = *VeriKopyalamaListesi*] biçiminde bir mesaj ile cevap verir.

MRD algoritmasının federatif veya hiyerarşik veri organizasyonu modelleri için farkı, veri kopyalama isteği oluşturulurken kaynak sitenin seçiminde ortaya çıkmaktadır. Söz konusu seçimi DMS servisi, veri dağıtımı algoritmalarını çalıştırarak yapar. Federatif modelde, Veri Grid sistemi üzerindeki herhangi bir site kaynak olabilirken; hiyerarşik modelde, sadece veri isteğinde bulunan sitenin Hiyerarşi-1 ebeveyn sitesi veya Hiyerarşi-0 site kaynak olarak seçilebilir. Her iki modelde de hedef site, veri kopyalama isteğinde bulunan sitedir.

MRD algoritması
// Giriş: *İşVeriListesi* ve *TransferSonZamanı*
// Çıkış: *VeriKopyalamaListesi*
// Her veri, v_i, için site içinden yapılan istek sayıları, n_i, LDRS'de birer
// değişkende tutulmaktadır

VeriKopyalamaListesi = *Boş*;
İşVeriListesi kümesindeki verilerin istek sayılarını bir arttır;
İşVeriListesi kümesindeki verilerin, ortalama istek son zamanlarını güncelle;
if istek sayısı, n_i, belirlenmiş bir eşik değeri aşan veri v_i var **then**
  VT1: verilerin toplam istek sayısına göre büyükten küçüğe sıralandığı tablo;
  VT2: verilerin ortalama *TransferSonZamanı*'na göre küçükten büyüğe
    sıralandığı tablo;
  $VeriAğırlıklıDeğer_i$ = a_1x(v_i'ın VT1'deki sırası)+a_2x(v_i'ın VT2'deki sırası);
  v_k = en büyük $VeriAğırlıklıDeğer_i$'e sahip olan veri v_i;
end if
if v_k sitede yok **then**
  VeriKopyalamaListesi += v_k;
  n_k = 0;
end if
return *VeriKopyalamaListesi*;
end program.

Şekil 5.1. MRD algoritması

5.2.2. İtme Tabanlı Dağıtık Veri Kopyalama

İtme tabanlı dağıtık veri kopyalama algoritmaları, 2. Bölüm'de anlatılan ve her sitede yer alan LDRS servisi tarafından gerçeklenirler. LDRS, kendisinin bulunduğu sitenin bünyesindeki veriler için, diğer sitelerden yapılan isteklerin istatistiklerini tutar. Belirli periyotlar ile, kopyalanacak veriyi ve kopyalanması için verinin kopyalanması için seçilen siteyi hesaplar.

Bu işlem için, LDRS, DMS'den gelen, *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = **UzakVeriKilitBilgisi*] veya *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = *#UzakVeriKilitBilgisi*] mesajlarını alır. *KopyalanacakVeri* = *VeriKilitBilgisi.Veri* ve *VeriTransferiSonZamanı* = *VeriKilitBilgisi.KilitBitişZamanı* bilgilerini göndererek itme tabanlı dağıtık bir veri kopyalama algoritmasını çalıştırır. Algoritmadan *VeriKopyalamaListesi* = <*TransferEdilecekVeriListesi, Hedef*, *TransferSonZamanı*> bilgilerini alır. LDRS, DM'ye *UZAK_VERİ_TRANSFERİ_İSTEĞİ* := [*TransferEdilecekVeriListesi* = v_k,

Hedef = s_m, *TransferSonZamanı* = *VeriKopyalamaTransferiSonZamanı*] biçiminde bir mesaj gönderir. İtme tabanlı dağıtık veri kopyalama stratejisinde, DM'ye sadece bir veri için uzak veri isteği gönderilir. İtme tabanlı dağıtık bir veri kopyalama algoritması olan ve çevrim dışı çalışan Gerçek-Zamanlı Dağıtık Ağırlıklı (Real-Time Distributed Weighted – RT_DIW) algoritması Şekil 5.2'de gösterilmiştir.

RT_DIW algoritması
// Giriş: *KopyalanacakVeri*, *VeriTransferiSonZamanı*
// Çıkış: *VeriKopyalamaListesi*
// Parametre: $0 \le a_1 \le 1$; $0 \le a_2 = 1-a_1 \le 1$; $0 \le b_1 \le 1$; $0 \le b_2 = 1-b_1 \le 1$
// Kaynak sitedeki her veri, v_i, için diğer sitelerden, s_j, yapılan istek sayıları,
// n_{ij}, ve toplam istek son zamanları, sz_{ij}, ve toplam yapılan istek sayısı, tis_i,
// LDRS'de değişkenlerde tutulmaktadır.

VeriKopyalamaListesi = *Boş*;
KopyalanacakVeri kümesindeki veriler için tis_i, n_{ij}, sz_{ij} değerlerini güncelle;
if toplam istek sayısı, tis_i, belirlenmiş bir eşik değeri aşan veri yok **then**
  return *VeriKopyalamaListesi*;
end if
VT1: verilerin toplam istek sayısına göre büyükten küçüğe sıralandığı tablo;
VT2: verilerin ortalama son zamana (son zamanların toplamı/toplam istek sayısı) göre küçükten büyüğe sıralandığı tablo;
for veri v_i **in** *KopyalanacakVeri*
  $VeriAğırlıklıDeğer_i = a_1$x($v_i$'ın VT1'deki sırası)+$a_2$x($v_i$'ın VT2'deki sırası);
end for
v_k = en büyük $VeriAğırlıklıDeğer_i$'e sahip olan veri v_i;
ST1: sitelerin, v_k'ya yapmış oldukları istek sayılarına göre büyükten küçüğe sıralandığı tablo;
ST2: sitelerin, v_k'ya yapmış oldukları isteklerin son zamanlarının ortalamalarına göre küçükten büyüğe sıralandığı tablo;
for site s_i **in** *potansiyelHedefSiteler*
  $SiteAğırlıklıDeğer_j = b_1$x($s_j$'ın ST1'deki sırası)+$b_2$x($s_j$'ın ST2'deki sırası);
end for
s_m = en büyük $SiteAğırlıklıDeğer_j$'e sahip olan site s_j;
VeriKopyalamaTransferiSonZamanı = ŞimdikiZaman +
  VeriBüyüklüğü/*VeriKopyalamaTransferHızı*;
VeriKopyalamaListesi = < *TransferEdilecekVeriListesi* = v_k, *Hedef* = s_m,
  TransferSonZamanı = *VeriKopyalamaTransferiSonZamanı*>;
VeriKopyalamaTablosu'nu güncelle;
return *VeriKopyalamaListesi*;
end program.

Şekil 5.2. RT_DIW algoritması

RT_DIW algoritması, öncelikle kopyalanacak veriyi, sonrasında ise kopyalanacağı hedef siteyi seçer. Veri kopyalama isteğinin kaynağı, RT_DIW algoritmasını çalıştıran LDRS'nin bulunduğu sitedir. RT_DIW algoritmasının federatif ve hiyerarşik veri organizasyon modellerindeki farkı, veri kopyalama isteği oluşturulurken, hedef sitenin seçiminde kullanılan *potansiyelHedefSiteler*'de ortaya çıkmaktadır. Federatif modelde, Veri Grid sistemi içerisindeki herhangi bir site hedef olabilirken; hiyerarşik modelde, sadece kaynak sitenin çocukları hedef site olabilirler.

Algoritmada belirtilen *TransferHızı*, Veri Grid sistemi üzerinde bulunan linklerin en az bant genişliğinden küçük bir hızdır ve kullanıcı tarafından sağlanan bir parametre kullanılarak bulunur.

5.2.1. İtme Tabanlı Merkezi Veri Kopyalama

İtme tabanlı merkezi veri kopyalama algoritmaları, 2. Bölüm'de anlatılan DRS servisi bünyesinde gerçeklenirler. DRS servisi, veri kopyalama isteklerini direk olarak DM servisine gönderir. Bu işlem için, DRS, DMS'den gelen, *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = **UzakVeriKilitBilgisi*] veya *VERİ_KİLİT_ONAY* := [*VeriKilitBilgisi* = #*UzakVeriKilitBilgisi*] mesajlarını alır. *KopyalanacakVeri* = *VeriKilitBilgisi.Veri* ve *VeriTransferiSonZamanı* = *VeriKilitBilgisi.KilitBitişZamanı* bilgilerini göndererek itme tabanlı merkezi bir veri kopyalama algoritmasını çalıştırır. Algoritmadan *VeriKopyalamaListesi* = <*TransferEdilecekVeriListesi, Hedef*, *TransferSonZamanı*> bilgilerini alır. DRS, *TransferEdilecekVeriListesi*'ndaki her veri için DM'ye *UZAK_VERİ_TRANSFERİ_İSTEĞİ* := [*TransferEdilecekVeriListesi* = v_k, *Hedef* = s_m, *TransferSonZamanı* = *VeriKopyalamaTransferiSonZamanı*] biçiminde mesajlar gönderir.

Şekil 5.3'de, çevrim dışı çalışan itme tabanlı merkezi bir veri kopyalama algoritması Gerçek-Zamanlı Merkezi Ağırlıklı (Real-Time Centralized Weighted - RT_CENT) algoritması sunulmuştur. DRS servisi, periyodik olarak RT_CENT algoritmasını çalıştırır ve kopyalama sürecini başlatır.

RT_CENT algoritması

// Giriş: *KopyalanacakVeri*, *VeriTransferiSonZamanı*
// Çıkış: yok
// Parametre: $0 \leq a_1 \leq 1$; $0 \leq a_2 = 1-a_1 \leq 1$; $0 \leq b_1 \leq 1$; $0 \leq b_2 = 1-b_1 \leq 1$
// Veri Grid sistemindeki tüm site s_i'larda bulunan tüm veri, v_{ij}, için diğer
// sitelerden, s_k, yapılan istek sayıları, n_{ijk}, ve toplam istek son zamanları, sz_{ijk},
// toplam yapılan istek sayısı, tis_{ij}, değerleri DRS'de tutulmaktadır.

KopyalanacakVeri kümesindeki veriler için tis_{ij}, n_{ijk}, sz_{ijk} değerlerini güncelle;
VT1: verilerin toplam istek sayısına göre büyükten küçüğe sıralandığı tablo;
VT2: verilerin verilerin ortalama son zamana (son zamanların toplamı/toplam; istek sayısı) göre küçükten büyüğe sıralandığı tablo;
Tüm veri, v_{ij}, için $VeriAğırlıklıDeğeri_{ij}$ hesapla;
$VeriAğırlıklıDeğeri_{ij} = a_1$ x (v_{ij}'ın VT1'deki sırası) + a_2 x (v_{ij}'ın VT2'deki sırası);
Verileri, $VeriAğırlıklıDeğeri_{ij}$ değerlerine gore büyükten küçüğe dizerek *VeriKopyalamaTablosu*'nu oluştur;
for each v_{ij} **in** *VeriKopyalamaTablosu*
  if $VeriAğırlıklıDeğeri_{ij} < a_1$ x (tüm verilere yapılan toplam erişim sayısı/en az bir sefer erişim yapılan verilerin sayısı) + a_2 x (tüm verilere yapılan toplam ortalama son zaman/en az bir sefer erişim yapılan verilerin sayısı) **then**
    VeriKopyalamaTablosu'ndan v_{ij}'ı sil;
  end if
end for
VeriKoyalamaTablosu'ndaki *EnKüçükVeriAğırlıklıDeğeri* bul;
for each v_{ij} **in** *VeriKopyalamaTablosu*
  $v_k = v_{ij}$; //v_m kopyalanacak olan veri;
  ST1: sitelerin, v_k'ye yapmış oldukları istek sayılarına göre büyükten küçüğe sıralandığı tablo;
  ST2: sitelerin, v_k'yeyapmış oldukları isteklerin son zamanlarının ortalamalarına göre küçükten büyüğe sıralandığı tablo;
  for site s_l **in** *potansiyelHedefSiteler*
    $SiteAğırlıklıDeğer_l = b_1$x($s_l$'ın ST1'deki sırası)+$b_2$x($s_l$'ın ST2'deki sırası);
  end for
  s_m = en büyük $SiteAğırlıklıDeğer_l$'e sahip olan site s_l;
  VeriKopyalamaTransferiSonZamanı = ŞimdikiZaman + VeriBüyüklüğü/TransferHızı;
  VeriKopyalamaListesi = <*TransferEdilecekVeriListesi* = v_k, *Hedef* = s_m, *TransferSonZamanı = VeriKopyalamaTransferiSonZamanı*>;
  VeriKopyalamaTablosu'nu güncelle;
end for
return *VeriKopyalamaListesi*;
end program.

Şekil 5.3. RT_CENT algoritması

RT_CENT algoritması da RT_DIW algoritması gibi, öncelikle kopyalanacak verileri, sonrasında ise verilerin kopyalanacağı hedef siteleri seçer. Veri kopyalama isteğinde, kopyalanacak veri v_{ij}'ı içinde barındıran site s_i kaynak sitedir. Verinin kopyalanması için, DM'ye bir veri transferi isteği gönderir ve *VeriKopyalamaTablosu*'ndaki ilgili verinin *VeriAğırlıklıDeğeri*'ni, *EnKüçükVeriAğırlıklıDeğeri* kadar azaltır. Eğer ilgili verinin *VeriAğırlıklıDeğeri*, *EnKüçükVeriAğırlıklıDeğeri*'nden daha küçük ise, veri *VeriKopyalamaTablosu*'ndan silinir.

Federatif veri organizasyonu modelinde *VeriKopyalamaTablosu* oluşturulurken tüm sitelerin, tüm diğer sitelere olan veri istekleri dikkate alınır. Algoritmada yer alan *potansitelHedefSiteler*'de, Veri Grid sistemi içerisindeki tüm sitelerdir. Hiyerarşik veri organizasyonu modelinde ise, *potansiyelerHedefSiteler*, ancak kaynak sitenin çocukları olabilirler. VeriKopyalamaTabloları da, Hiyerarşi-0 ve Hiyerarşi-1 siteler için ayrı ayrı oluşturulurlar. Hiyerarşi-0 sitenin veri kopyalama tablosu, bu siteye sadece Hiyerarşi-1 sitelerden yapılmış olan veri istekleri gözetilerek oluşturulur. Her bir Hiyerarşi-1 sitenin veri kopyalama tablosu ise, bu siteye bu sitenin Hiyerarşi-0 çocuk sitelerinden yapılmış olan veri istekleri gözetilerek oluşturulur.

5.3. Simülasyon Sonuçları ve Yorumlar

Simülasyon çalışmalarında, bu çalışmanın 3. Bölüm'ünde anlatılan DGridSim simülatörü kullanılmıştır. Simülasyonlar sırasında, rassal veri erişim modelinde, Bölüm 3'te iyi performans gösterdiği belirtilen Rand iş çizelgeleme algoritması, geometrik ve zipf veri erişim modellerinde isemodelde ise iyi performans gösteren MMwDP algoritması kullanılacaktır. Simülasyonlarda, 3. Bölüm'de arz edilen sonuçlar ile karşılaştırma yapılabilmesi için söz konusu bölümde kullanılan simülasyon parametreleri kullanılmıştır. Ayrıca, Bölüm 3'te de veri dağıtımı algoritması olarak kullanılan SP_MinHop algoritması kullanılmaktadır.

Çizelge 5.1'de farklı veri kopyalama algoritmaların, sistemin gerçek zaman performansı üzerindeki etkisi gösterilmiştir. Çizelgenin ilk satırında, veri kopyalama ve yenilemenin olmadığı baz durum belirtilmiştir. LRU veri yenileme

algoritmasının kullanıldığı durumda, federatif ve hiyerarşik veri organizasyonu modellerinde en iyi sonuçları itme tabanlı merkezi veri kopyalama algoritması göstermektedir. Bu durumda, veri kopyalama yapılırken, sistemi bir bütün olarak değerlendirmenin önemini vurgulamaktadır. Geometrik veri erişim düzeni modelinde, gerçek-zamanlı işlerin ihtiyaç duyduğu veriler, sürekli olarak küçük bir veri öbeğinden seçilmektedir. Bu sebeple, veri kopyalamanın gerçek-zaman performansı üzerindeki etkisi büyük olmaktadır. Zipf veri organizasyon modelinde ise, ihtiyaç duyulan veriler, daha büyük bir veri öbeği içerisinden seçilmektedir. İşlerin ihtiyaç duyduğu verilerden en az bir tanesinin, çok kullanılmayan verilerin arasından seçilme ihtimali fazladır. Bu durum sistem performansını da olumsuz etkilemektedir.

Çizelge 5.1. Veri kopyalama algoritmalarının performansa etkisi

Veri kopyalama algoritması		**Rassal**		**Geometrik**		**Zipf**	
		Fed.	**Hiye.**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**
MRD	a1=0.0 a2=1.0	62	57	66	96	61	66
	a1=0.5 a2=0.5	61	56	65	96	60	66
	a1=1.0 a2=0.0	61	55	65	95	60	66
RT_DIW	a1=b1=0.0 a2=b2=1.0	60	56	64	96	60	66
	a1=b1=0.5 a2=b2=0.5	59	55	64	96	59	66
	a1=b1=1.0 a2=b2=0.0	59	55	64	96	59	66
RT_CENT	a1=b1=0.0 a2=b2=1.0	65	59	70	97	64	68
	a1=b1=0.5 a2=b2=0.5	64	57	69	97	62	67
	a1=b1=1.0 a2=b2=0.0	64	56	67	96	62	67

Veri kopyalama algoritmalarında kullanılan a1 ve b1 parametreleri, istek sayısının etkisini belirtmekte; a2 ve b2 parametreleri ise istekler için ortalama son zamanın etkisini belirtmektedir. Çizelge 5.1'de de gösterildiği gibi, ortalama son

zamanın daha fazla etkili olduğu durumlarda, Veri Grid sistemlerinin performansları daha iyi olmaktadır.

Çizelge 5.2'de veri kopyalama algoritmalarının ortalama simülasyon sonuçları sunulmuştur. Bu sonuçlara göre, her ne kadar en iyi simülasyon zamanları, çekme tabanlı MRD algoritması tarafından sağlansa da, algoritmalar arasında belirgin bir zaman farkı bulunmamaktadır.

Çizelge 5.2. Veri kopyalama algoritmalarının ortalama çalışma zamanları

Veri kopyalama algoritması		**Rassal**		**Geometrik**		**Zipf**	
		Fed.	**Hiye.**	**Fed.**	**Hiye.**	**Fed.**	**Hiye.**
MRD	a1=0.0 a2=1.0	153	678	518	487	571	450
	a1=0.5 a2=0.5	156	684	523	490	572	452
	a1=1.0 a2=0.0	162	690	526	496	572	457
RT_DIW	a1=b1=0.0 a2=b2=1.0	163	732	625	527	590	490
	a1=b1=0.5 a2=b2=0.5	167	741	625	532	591	495
	a1=b1=1.0 a2=b2=0.0	172	748	629	534	587	507
RT_CENT	a1=b1=0.0 a2=b2=1.0	155	687	538	510	581	465
	a1=b1=0.5 a2=b2=0.5	160	693	547	513	584	471
	a1=b1=1.0 a2=b2=0.0	160	705	556	524	582	482

6. SONUÇLAR

Bu çalışma kapsamında, literatürde mevcut Grid simülatörleri incelenmiştir ve mevcut simülatörlerin simülasyon altyapılarını destekleyebilecek bir model üzerinde durulmuştur. Yapılan çalışmaların sonucunda ortaya çıkan model tanıtılmıştır. Önerilen modelde, Veri Grid sisteminin parçalarını oluşturan servisler ve servisler arası etkileşimler iyi tanımlanmıştır. Bu model çerçevesi, dört temel esas olan iş çizelgeleme, veri kopyalama, veri dağıtımı ve ön rezervasyon sistemlerini içermektedir. Ayrıca önerilen modelin, Grid sistemleri sınıflandırmasındaki yeri de incelenmiştir.

Tez kapsamında önerilen Veri Grid sistemi modelini gerçekleyen DGridSim simülatörü geliştirilmiştir. DGridSim'in, literatürdeki benzer simülatörlere göre önemli bazı avantajları bulunmaktadır. Bu avantajlardan bazıları şunlardır:

1. Çevrim içi ve dışı çalışabilen algoritmaları desteklemektedir.
2. Farklı Veri Grid sistem modellerini destekleyebilecek şekilde tasarlanmıştır.
3. Tüm işlem, veri depolama ve ağ kaynakları için önceden rezervasyon özelliğini desteklemektedir.
4. İş çizelgeleme, veri dağıtımı ve veri kopyalama algoritmaları, ön rezervasyon sistemini kullanabilmektedir. Literatürde, ön rezervasyon sistemini kullanarak iş çizelgeleme, veri dağıtımı ve veri kopyalama algoritmalarının simülasyonunu yapabilecek yetenekte diğer bir simülatör bulunmamaktadır.
5. Görsel arayüzü kullanılarak simülasyonu yapılmak istenen Veri Grid sisteminin tüm özellikleri kolayca belirlenebilmektedir.
6. DGridSim, üzerinde çalışmış olduğu bilgisayar sisteminde bulunan tüm çekirdekleri kullanarak simülasyon yapabilmektedir.
7. Son derece modüler ve esnek bir yapıda tasarlanmış olup, isteyen araştırmacılar farklı iş çizelgeleme, veri dağıtımı ve veri kopyalama algoritmalarını rahatlıkla ekleyebilirler.
8. DGridSim üzerinde mevcut durumda gerçeklenmiş ve araştırmacıların kendi kodlarını da yazmasına yardımcı olacak çok sayıda iş çizelgeleme,

veri dağıtımı, veri kopyalama ve veri yenileme algoritmaları bulunmaktadır.

9. Araştırmaların devamında DGridSim simülatörünün yeni sürümleri gerçeklenmeye devam edilecektir. Yeni sürümlerle birlikte, yeni sistem modellerinin, iş çizelgeleme, veri çizelgeleme, veri kopyalama ve veri yenileme algoritmalarının DGridSim tarafından desteklenmesi planlanmaktadır.

DGridSim simülatörü üzerinde beş adet iş çizelgeleme algoritmasının (Rand, EDF, MCTF, MCTFwDP, MMwDP) gerçek-zaman performansları karşılaştırılmıştır. Bu testlerde veri dağıtımı algoritması olarak SP_MinHop algoritması, veri kopyalama algoritması olarak çekme tabanlı MRD algoritması ve veri yenileme algoritması olarak da LRU veri yenileme algoritması kullanılmıştır. Bu testlerin sonucuna göre rassal veri erişimi modeline uyan isteklerde Rand iş çizelgeleme algoritmasının, geometrik ve zipf modeline uyan isteklerde ise MMwDP iş çizelgeleme algoritmalarının en iyi performansı gösterdikleri sonucuna varılmıştır.

Ön rezervasyon sisteminin olduğu bir Veri Grid sistemindeki veri dağıtımı problemi üzerine çalışmalar yapılmıştır. Öncelikle UFP'nin genel hali olan RTU/DDP üzerinde durulmuştur. Bu probleme biçimsel bir tanımlama getirilmiş ve iki farklı yaklaşım ile çözülmeye çalışılmıştır. Birinci yaklaşımda veri transferi isteği için uygun rotanın bulunması ile kaynakların rezerve edilmesi tek aşamada yapılmıştır. İkinci yaklaşımda ise çözüm, iki alt problemin çözümüne dönüştürülmüştür: Gerçek-Zamanlı Rota Seçimi (GZRS) ve Gerçek-Zamanlı İstek Seçimi (GZİS). Bu iki alt problem için ayrı ayrı keşifsel algoritmalar önerilmiştir. İki yaklaşım için de önerilen algoritmalar test edilmiştir. GZRS algoritması olarak MinDepay/FPF, GZİS algoritması olarak da MOFF, diğer algoritmalara göre daha iyi sonuçlar vermektedir. Ancak, problemin çözümünde birinci yaklaşımın ikinci yaklaşıma göre daha iyi sonuçlar verdiği görülmüştür.

RTU/DDP'nin genelleştirilmiş hali olan RTS/DDP üzerinde çalışmalar yapılmıştır. Bu problemde, bir istek birden fazla rotadan eşzamanlı olarak gönderilen akışlar ile karşılanmaktadır. Performans ölçümleri, bir isteğin birden fazla rotadan eşzamanlı karşılanmasının, linklerin bant genişliğinin daha dengeli

ve verimli kullanılmasını sağladığı, tıkanıklığa neden olan link sayısını ve istek sayısını azalttığı, dolayısı ile sistemin gerçek-zaman performansını arttırdığını göstermektedir. Farklı rotaların bulunması için iki farklı algoritma önerilmiştir. Performans ölçümlerine göre *k*-en kısa yol algoritması (*k*SP), *k*-ayrık yol algoritmasına (*k*DP) göre daha iyi sonuçlar vermektedir. Farklı rotalar üzerinden eşit miktarda veri transferi yapılmasının (ESMP); rotaların kullanılabilir bant genişlikleri ile orantılı miktarda veri transferi yapılmasını öngören dengeli bölüşümlü veri transferine (BSMP) göre de daha iyi sonuçlar verdiği gözlemlenmiştir.

Veri kopyalama, verilerin uygulamaları yapacak sitelere daha yakın olmaları hedeflenmektedir. Bu sayede, ortalama veri transferi süreleri ile link kullanım gereksinimlerinin azalacağı; dolayısı ile sistemin gerçek-zaman performansının artacağı öngörülmektedir. Simülasyon sonuçları da, beklentiler ile örtüşmektedir.

DGridSim üzerinde, veri kopyalama için çekme tabanlı, itme tabanlı dağıtık ve itme tabanlı merkezi olmak üzere üç farklı model önerilmiş ve her model için de örnek veri kopyalama algoritmaları gerçeklenmiştir. Çekme tabanlı veri kopyalama için MRD, itme tabanlı dağıtık için RT_DIW, itme tabanlı merkezi için ise RT_CENT veri kopyalama algortmaları önerilmiştir. Gerçeklenmiş olan algoritmalar arasında yapılan test sonuçlarına gore, en iyi sonuçları, LRU veri yenileme algoritmasını kullanan RT_CENT algoritması göstermiştir.

Araştırmanın devamında, literatürdeki farklı Veri Grid sistem modellerini, iş çizelgeleme, veri dağıtımı, veri kopyalama ve veri yenileme algoritmalarını da dâhil ederek DGridSim simülatörünün kullanışlılığını, kullanılabilirliğini, kapsamını ve bilinirliliğini arttırmak için çalışmalar devam edecektir. Çoklu rotadan veri dağıtımı ve veri kopyalama algoritmalarının, gerçek zaman kriterlerine biçimsel problem tanımlamaları ve yeni keşifsel algoritmalar önerilmesi ve simülasyon çalışmaları yapılması planlanmaktadır.

Mevcut sürümünde DGridSim'de işlerin istekte bulundukları veriler, rassal, geometrik ve zipf dağılımlarına uygun olarak seçilmektedirler. Literatürde tanımlanmış diğer dağılımların seçilmesi durumunda, algoritmaların gerçek-zaman performanslarında karşılaşılan performans değişiklikleri de test edilecektir.

Bu çalışmada önerilen model, sadece bağımsız işler (independent tasks) iş modelini desteklemektedir. Önerilen modelin, iş torbası (bag of tasks) veya iş akışları (workflows) gibi iş modellerini de desteklemesi için çalışmalar yapılacaktır.

Ayrıca, rezervasyon modeli de güncellenecektir. DGridSim'in mevcut sürümünde ağ ve depolama kaynakları zaman-paylaşımlı (time-shared) çalışabilmektedirler. Ancak işlem elemanları, sadece uzay-paylaşımlı (space-shared) olarak çalıştırılabilmektedir. Benzeri sınırlamaların da kaldırıldığı yeni sürümlerin yapılması planlanmaktadır.

KAYNAKLAR

[1] Foster, I. ve Kesselman, C., The Grid: Blueprint for a new computing infrastructure, Morgan Kaufmann Publishers, San Francisco, 1999.

[2] Grimshaw, A.S. ve Wulf, W.A., "The legion vision of a worldwide virtual computer," Communications of the ACM, No: 40(1), 39-45, 1997.

[3] Foster I. ve Kesselman C., "Globus: A Metacomputing Infrastructure Toolkit," International Journal of Supercomputer Applications, No: 11(2), 115-128, 1997.

[4] Buyya, Abramson, ve Giddy, "Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid," Proc. of 4th International Conference and Exhibition on High Performance Computing in Asia Pacific Region (HPC ASIA 2000), IEEE Computer Society Press, 14–17, 2000.

[5] Casanova, H., Obertelli, G., Berman, F. ve Wolski, R., "The AppLeS parameter sweep template: User-level middleware for the grid," Proc. of IEEE Supercomputing Conference, Dallas, USA, 2000.

[6] Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M. ve Werthimer, D., "SETI@home: an experiment in public-resource computing," Communications of the ACM, No: 45(11), 56–61, 2002.

[7] Frey, J., Tannenbaum, T., Foster, I., Livny, M. ve Tuecke, S., "Condor-G: A computation management agent for multi-institutional grids," Proc. of 10th Symposium on High Performance Distributed Computing, 2001.

[8] Allcock, W., Bresnahan, J., Bunn, J., Hegde, S., Insley, J., Kettimuthu, R., Newman, H., Ravot, S., Rimovsky, T., Steenberg, C. ve Winkler, L., "Grid-enabled Particle Physics Event Analysis: Experiences Using a 10 Gb, High-Latency Network for a High-Energy Physics Application," Future Generation Computer Systems, No: 19, 983-997, 2003.

[9] The DataGrid Project, http://eu-datagrid.web.cern.ch/eu-datagrid/.

[10] Allcock, B., Foster, I., Nefedova, V., Chervenak, A., Deelman, E., Kesselman, C., Lee, J., Sim, A., Shoshani, A., Drach, B. ve Williams, D., "High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies," Supercomputing, 2001.

[11] Cui, Y., Moore, R., Olsen, K., Chourasia, A., Maechling, P., Minster, B., Day, S., Hu, Y., Zhu, J., Majumdar, A. ve Jordan, T., "Enabling Very-Large Scale Earthquake Simulations on Parallel Machines," International Conference on Computational Science, 2007.

[12] International Virtual Observatory Alliance, http://www.ivoa.net.

[13] BioGrid, http://www.biogrid.jp/.

[14] Moore, R., Rajaseka, A. ve Wan, M., "Data Grids, Digital Libraries and Persistent Archives: An integrated approach to publishing, sharing and archiving datas," Proc. of the IEEE (Special Issue on Grid Computing), No: 93(3), 2005.

[15] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. ve Tuecke, S., "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," Journal of Network and Computer Applications, No: 23(3), 187-200, 2000.

[16] GridPP, http://www.gridpp.ac.uk/.

[17] Earth System Grid, http://www.earthsystemgrid.org/.

[18] Worldwide LHC Computing Grid, http://lcg.web.cern.ch/lcg/

[19] International Virtual Data Grid Laboratory, http://www.ivdgl.org/.

[20] Enabling Grids for E-Science, http://www.eu-egee.org/.

[21] Network for Earthquake Engineering Simulation, http://it.nees.org/.

[22] Diagnostic Mammography National Database, http://www.ediamond.ox.ac.uk/.

[23] Lefebure, V. ve Wildish, T., The Spring 2002 DAQ TDR Production, CMS Internal Note, Geneva, Switzerland, 2005.

[24] OGSA v1.5, http://www.gridforum.org/documents/GFD.80.pdf.

[25] Foster, I., Kesselman, C., Nick, J.M. ve Tuecke, S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

[26] RFC 2212: Specification of Guranteed Quality of Service.

[27] Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. ve Tuecke, S., "Secure, efficient data transport and replica management for high-performance data-intensive computing," IEEE Mass Storage Conference, 2001.

[28] Chandra, P., Fisher, A. ve Kosak, C., "Darwin: Customizable Resource Management for Value-Added Network Services," Proc. of the 6th Int'l Conf. On Network Protocols, IEEE,1988.

[29] Dalheimer, M., Pfreundt, F.J. ve Merz, P., "Calana – A General-purpose Agent-based Grid Scheduler," Proc. of the 14th IEEE Int'l Symposium on High Performance Distributed Computing (HPDC-14), 2005.

[30] VIOLA – Vertically Integrated Optical Testbed for Large Application in DFN, http://www.viola-testbed.de/.

[31] Weissman, J., Gallop: The benefits of wide-area computing for parallel processing, Technical report, University of Texas at San Antonio, 1997.

[32] Cao, J., Jarvis, S.A., Saini, S. ve Nudd, G.R., "GridFlow: Workflow Management for Grid Computing," Proc. of the 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan, 2003.

[33] Litzkow, M., Livny, M., Mutka, M., "Condor - a hunter of idle workstations," Proc. 8th Intl Conf. on Distributed Computing Systems, 1988.

[34] The GridWay Project, http://www.gridway.org, 2006.

[35] Almond, J. ve Snelling, D., "UNICORE: Uniform Access to Supercomputing as an Element of Electronic Commerce," Future Generation Computer Systems, 1999.

[36] Cooper, K., "New Grid Scheduling and Rescheduling Methods in the GrADS Project", NSF Next Generation Software Workshop, International Parallel and Distributed Processing Symposium, 2004.

[37] Nakada, H., Sato, M. ve Sekiguchi, S., "Design and Implementation of Ninf: towards a Global Computing Infrastructure," Future Generation Computing Systems, 1999.

[38] Casanova, H. ve Dongarra, J., "NetSolve: A Network Server for Solving Computational Science Problems," Intl. Journal of Supercomputing Applications and High Performance Computing, No: 11(3), 1997.

[39] The DRMAA Working Group, http://www.drmaa.org, 2006.

[40] Dijsktra, E.W., "A Note on Two Problems in ConnexionWith Graphs," NumerischeMathematik, 1959.

[41] Bellman, R., "On a Routing Problem," Quarterly of Applied Mathematics, No: 16, 87-90, 1958.

[42] RFC 1058: Routing Information Protocol.

[43] RFC 2328: OSPF Version 2.

[44] RFC 4271: A Border Gateway Protocol 4.

[45] Callon, R., Jeffords, J., Sandick, H., Halpern, J.M., Issues and Approaches for Integrated PNNI, ATM Forum 96-0355, 1996.

[46] RFC 2676: QoS Routing Mechanisms and OSPF Extensions.

[47] Bahk, S. ve Zarki, M., "Dynamic Multi-path Routing and How it Compares with other Dynamic Routing Algorithms for High Speed Wide Area Networks," ACM Computer Communications Review, No: 22(4), 53-64, 1992.

[48] Skutella, M., "Approximating the Single-Source Unsplittable Min-Cost Problem," Mathematical Programming, 2002.

[49] Ranganathan, K., Foster, I., "Simulation studies of computation and data scheduling algorithms for data grids," Journal of Grid Computing, No: 1(1), 2003.

[50] Camaron, D.G., Millar, A.P., Nicholson, C., Schiaffino, R.C., Zini, F. ve Stockinger, K., "Analysis of Scheduling and Replica Optimisation Strategies for Data Grids using OptorSim," Journal of Grid Computing, No: 2(1), 57-69, 2004.

[51] Nicholson, C., Camaron, D.G., Doyle, A.T., Millar, A.P. ve Stockinger, K., "Dynamic Data Replication in LCG 2008," UK e-Science All Hands Conference, 2006.

[52] Tang, M., Lee, B.S., Tang, X., Yeo, C.K., "The Impact of Data Replication on Job Scheduling Performance in the Data Grid," Future Generation Computer Systems, No: 22, 254-268, 2006.

[53] Doğan, A., "A Study on Performance of Dynamic File Replication Algorithms for Real-time File Access in Data Grids," Future Generation Computer Systems, No: 25, 829-839, 2009.

[54] Lamehamedi, H., Shentu, Z., Szymanski, B., Deelman, E., "Simulation of Dynamic Data Replication Strategies in Data Grids," Proc. of 12th Heterogeneous Computing Workshop (HCW2003), IEEE-CS Press, 2003.

[55] Desprez, F. ve Vernois, A., Simultaneous Scheduling of Replication and Computation for Data-Intensive Applications on the Grid, Technical Report, RR2005-01, INRIA, 2005.

[56] Ranganathan, K., Foster, I., "Identifying dynamic replication strategies for a high performance data grid," Lecture Notes In Computer Science, No: 2242, 75-86, 2001.

[57] Tang, M., Lee, B.S., Yeo, C.K. ve Tang, X., "Dynamic replication algorithms for the multi-tier data grid," Future Generation Computer Systems, No: 21, 775-790, 2005.

[58] Rahman, M.R., Barker, K. ve Alhaji, R., "Replica placement design with static optimality and dynamic maintainability," IEEE Int'l Symposium on Cluster Computing and the Grid, 2006.

[59] Foster, I., Kesselman, C., Lee, C., Lindell, R., Nahrstedt, K. ve Roy, A., "A Distributed Resource Management Architecture That Supports Advance Reservation and Coallocation," International Workshop on Quality of Service, 27-36, 1999.

[60] Maui scheduler, http://www.clusterresources.com/products/maui-cluster-scheduler.php/.

[61] Sulistio, A., Cibej, U., Prasad, S., Buyya, R., "GarQ: An Efficient Scheduling Data Structure for Advance Reservations of Grid Resources," Int'l Journal of Parallel, Emergent and Distributed Systems, No: 24(1), 1-19, 2009.

[62] Open Grid Forum, http://www.gridforum.org/.

[63] RFC 1633: Integrated Services in the Internet Architecture: an Overview.

[64] RFC 2205: Resource Reservation Protocol (RSVP).

[65] Resende, M. ve Ribeiro, C., "Greedy Randomized Adaptive Search Procedures", State-of-the-art Handbook in Metaheuristics, Kluwer Academic Publishers, 2002.

[66] The Knowledge Grid Lab, http://dns2.icar.cnr.it/kgrid/.

[67] Andretto, P., Borgia, S., Dorigo, A., Gianelle, A. ve Mordacchini, M., "Practical approaches to Grid workload & resource management in the EGEE Project," CHEP 2004, 2005.

[68] Chervenak, A., Deelman, E., Foster, I., Guy, L., Hoschek, W., Iamnitchi, A., Kesselman, C., Kunst, P., Ripeanu, M., Schwartzkopf, B., Stockinger, H., Stockinger, K. ve Tierney, B., "Giggle: A framework for constructing scalable replica location services," Proc. of Supercomputing (SC'2002), 2002.

[69] Nakada, H., Takefusa, A., Ookubo, K., Ku-doh, T., Tanaka, Y. ve Sekiguchi, S, "An Advance Reservation-Based Computation Resource Manager for Global Scheduling," Proc. of 3rd Workshop on Grid Computing and Applications, 3–14, 2007.

[70] Thain, D., Basney, J., Son, S.C. ve Livny, M., "The Kangaroo Approach to Data Movement on the Grid," Proc. of the 10th IEEE Symposium on High Performance Distributed Computing, IEEE CS Press, 2001.

[71] Foster, I., Kesselman, C. ve Tuecke, S., "The Nexus approach to integrating multithreading and communication," Journal of Parallel and Distributed Computing, No: 37, 70-82, 1996.

[72] Kunszt, P., Laure, E., Stockinger, H. ve Stockinger, K., "Replica Management with Reptor," Proc. of 5th International Conference on Parallel Processing and Applied Mathematics, 2003.

[73] Samar, A. ve Stockinger, H., "Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication," Proc. of the IASTED Int'l Conference on Applied Informatics (AI'2001), 2001.

[74] Venugopal, S., Buyya, R. ve Ramamohanarao, K., "A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing," ACM Computing Surveys, No: 38, 2006.

[75] Pacini, F., Job Description Language How To, 2003, http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0142-02.pdf.

[76] Terzis, Nikoloudakis, Wang ve Zhang, "Irl-sim: A general purpose packet level network simulator," Proc. of the 33rd Annual Simulation Symposium, 2000.

[77] Kim, J.K., "Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment," Journal of Parallel Distributed Computing, No: 67, 154-169, 2007.

[78] The Grid'5000 project, https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home.

[79] NS Network Simulator, http://www.mash.cs.berkeley.edu/ns.

[80] Grangfer, G.R., Worthington, B.L. ve Patt, Y.N., The DiskSim Simulation Environment, Reference Manual Version 2.0, Univ. of Michigan, 1999.

[81] Bell, W.H., Cameron, D.G., Capozza, L., Millar, A.P., Stockinger, K. ve Zini, F., "OptorSim - A grid simulator for studying dynamic data replication strategies," International Journal of High Performance Computing Applications, No: 17 (4), 2003.

[82] Buyya, R. ve Murshed, M., "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," Journal of Concurrency and Computation: Practice and Experience, 1–32, Wiley Press, 2002.

[83] Casanova, H., Legrand, A. ve Marchal, L., "Scheduling distributed applications: the simgrid simulation framework," Proc. of the 3rd IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03), 138–144, 2003.

[84] Casanova, H., "SimGrid: A toolkit for the simulation of application scheduling," Proc. of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'01), 2001.

[85] Song, H., Liu, X. ve Jakobsen, D., "The Microgrid: a scientific tool for modeling computational grids," IEEE Supercomputing, 2000.

[86] Liu, X. ve Chien, A., "Realistic large-scale online network simulation," Proc. of IEEE Supercomputing 2004, 2004.

[87] Takefusa, Matsuoka, Nakada, Aida ve Nagashima, "Overview of a performance evaluation system for global computing scheduling algorithms," Proc. of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC'8), 1999.

[88] Takefusa, A., Tatebe, O., Matsuoka, S. ve Morita, Y., "Performance analysis of scheduling and replication algorithms on grid datafarm architecture for high energy physics applications," Proc. of the 12th IEEE international Symposium on High Performance Distributed Computing (HPDC'12), IEEE CS Press, 2003.

[89] Dumitrescu, C.L. ve Foster, I., "Gangsim: A simulator for grid scheduling studies," Cluster Computing and Grid, 2005.

[90] Bagchi, S., "Simulation of grid computing infrastructure: challenges and solutions," Proc. of the 2005 Winter Simulation Conference, 2005.

[91] User's Guide: CSIM20 Simulation Engine (C++ Version), http://www.mesquite.com.

[92] Rockmore, A.J., BADD Functional Description, internal DARPA memo, 1996.

[93] Hea, T., Stankovic, J.A., Lub, C. ve Abdelzahera, T., "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," International Conference on Distributed Computing Systems, 2003.

[94] Massoulié, L., Roberts, J.W., "Bandwidth sharing and admission control for elastic traffic," Telecommunication Systems, No: 15(1-2), 185-201, 2000.

[95] Naghshineh, M., "Distributed call admission control in mobile/wireless networks," IEEE Journal on Selected Areas in Communications, No: 14(4), 711-717, 1996.

[96] Chen, J., "Two Bandwidth Packing Algorithms for a Centralized Wireless Network and Their Average-case Analysis," Fifth Int'l Conference on Information, Communications and Signal Processing, 2005.

[97] Ma, G.H.K. ve Zomaya, A.Y., "An efficient channel allocation scheme for cellular network using maximum channel packing," Wireless Communications and Mobile Computing, No: 4, 683–692, 2004.

[98] Gallager, R.G., "A Minimum Delay Routing Algorithm Using Distributed Computation," IEEE/ACM Transactions on Networking, No: 1, 130-141, 1977.

[99] Vutukury, S. ve Garicia-Luna-Aceves, J.J., "MDVA: A Distance Vector Multipath Routing Protocol," Proc. of IEEE INFOCOM'2001, 2001.

[100] Wang, Y. ve Wang, Z., "Explicit Routing Algorithms for Internet Traffic Engineering," Proc. of IEEE ICCCN'99, 1999.

[101] Ahuja, R.K., Magnanti, T.L., Orlin, J.B., Network Flows, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[102] Korkmaz, T. ve Krunz, M., "Multi-constrained optimal path selection", 20th Annual Joint Conference of the IEEE Computer and Communications Societies, No: 2, 834–843, 2001.

[103] Shin, D.W., Chong, E.K.P. ve Siegel, H.J., "A multiconstraint QoS routing scheme using the depth-first search method", IEEE Workshop on High Performance Switching and Routing, 385–389, 2001.

[104] Resende, R.A., Oliveira, R.M., Padua, F.J.L., Yamakami, A., Bonathi, I.S. ve Lavelha, A.C., "An algorithm for quality-of-service unicast routing in data communication networks", 13th International Conference on Telecommunications, 2006.

[105] Jung, E.S., Li, Y., Ranka, S. ve Sahni S., “An evaluation of in-advance bandwidth scheduling algorithms for connection-oriented networks,” International Symposium on Parallel Architectures, Algorithms and Networks, 2008.

[106] Liu, L. ve Feng, G., “Simulated annealing based multi constrained QoS routing in mobile ad hoc networks,” International Journal of Wireless Personal Communications, No: 41(3), 393–405, 2007.

[107] Cabrera, G. ve Toledo, C., "A Tabu Search Algorithm with a Probabilistic Neighbor Selection Criterion for Capacitated Multicommodity Network Flow Problem," 2010 International Conference on Technologies and Applications of Artificial Intelligence, 2010.

[108] Yuan, Y. ve Wang, D., "Multi-Objective Path Selection Model and Algorithm for Emergency Evacuation," IEEE International Conference on Automation and Logistic, 340-344, 2007.

[109] Yen, Y.S., Chang, R.S. ve Chao, H.C., “Flooding limited for multi-constrained quality-of-service routing protocol in mobile ad hoc networks”, IET Communications, No: 2(7), 971–981, 2008.

[110] Dai, F.S. ve Liu, A.J., “A multi-constrained quality-of-service routing algorithm based on vector converting”, 5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009.

[111] Wookahn, C. ve Ramakrishna, R.S., “A genetic algorithm for shortest path routing problem and the sizing of populations”, IEEE Transactions on Evolutionary Computing, No: 6, 566–579, 2002.

[112] Munetomo, M., Takai, Y., Sato, Y., “A migration scheme for the genetic adaptive routing algorithm”, IEEE International Conference on Systems, Man and Cybernatics, No: 3, 2774–2779, 1998.

[113] Munetomo, M., Yamaguchi, N., Akama, K. ve Sato, Y., “Empirical investigations on the genetic adaptive algorithm in the Internet”, 2001 Congress of Evolutionary Computing, No: 2, 1236–1243, 2001.

[114] Sinclair, M.C., “Minimum cost routing and wavelength allocation using genetic algorithm / heuristic hybrid approach”, 6th IEE Conference on Telecommunications, 62-71, 1998.

[115] Shimamoto, N., Hiramatsu, A. ve Yamasaki, K., “A dynamic routing control based on a genetic algorithm”, IEEE Conference in Neural Networks, 1123-1128, 1993.

[116] Hamdan, M. ve El-Hawary, M.E., “Hopfield-Genetic approach for solving the routing problem in computer networks”, Canadian Conference on Electrical and Computer Engineering, No: 2, 823–827, 2002.

[117] Yu, Z., Xin, Z. ve Qingwei, Y., "An Effective Genetic Algorithm for QoS-Based Routing Optimization Problem," 1st International Conference on Information Science and Engineering (ICISE), 2009.

[118] Leung, Y., Li, G. ve Xu, Z.B., “A genetic algorithm for the multiple destination routing problems,” IEEE Trans. Evol. Comput., No: 2, 150-161, 1998.

[119] Xiawei, Z., Changjia, C. ve Gang, Z., “A genetic algorithm for multicasting routing problem,” Proc. Int. Conf. Communication Technology (WCC-ICCT 2000), 1248–1253, 2000.

[120] Zhang, Q. ve Leung, Y.W., “An orthogonal genetic algorithm for multimedia multicast routing,” IEEE Trans. Evol. Comput., No: 3, 53–62, 1999.

[121] Guerin, R. ve Orda, A., “QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms,” IEEE/ACM Trans. Net, 1999.

[122] Ma, Q. ve Steenkiste, P., “Quality of Service Routing with Performance Guarantees,” Int’l. IFIP Wksp. Quality of Service, 1997.

[123] Goyal, P. ve Hjalmtysson, G., “QoS Routing for Best-Effort Flows,” Proc. Int’l. Wksp. Network and Operating System Support for Digital Audio and Video (NOSSDAV), 1999.

[124] Wang, Z. ve Crowcroft, J., “Quality-of-Service Routing for Supporting Multimedia Applications,” IEEE JSAC, 1996.

[125] Grimmell, W. ve Rao, N., “On source-based route computation for quickest paths under dynamic bandwidth constraints,” International Journal on Foundations of Computer Science,No: 14(3), 503-523, 2003.

[126] Yang, Y., Muppala, J.K. ve Chanson, S.T., "Quality of service routing algorithms for bandwidth-delay constrained applications," Ninth International Conference on Network Protocols, 2001.

[127] Chen, S. ve Nahrstedt, K., "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions," IEEE Network, No: 12(6), 1998.

[128] Korkmaz, T. ve Krunz, M., "Multi-Constrained Optimal Path Selection," Proc. Conf. Comp. Commun., (IEEE INFOCOM'01), 2001.

[129] RFC 2991: Multipath Issues in Unicast and Multicast Next-Hop Selection.

[130] Vutukury, S. ve Garicia-Luna-Aceves, J.J., "A Traffic Engineering Approach based on Minimum-Delay Routing," Proc. IEEE ICCCN'2000, 2000.

[131] Awduche, D., Malcolm, J., Agogbua, J. ve O'Dell, M., "Requirements for Traffic Engineering Over MPLS," Proc. of IETF, 1999.

[132] Rao, N.S.V. ve Batsell, S.G., "QoS Routing Via Multiple Paths Using Bandwidth Reservation," Proc. IEEE INFOCOM'98, 1998.

[133] Martens, M. ve Skutella, M., "Flows On Few Paths: Algorithms and Lower Bounds," Algorithms - ESA 2004, Lecture Notes in Computer Science, 2004.

[134] Koch, R., Skutella, M. ve Spenke, I., "Approximation and Complexity of k-splittable Flows," WAOA, Lecture Notes in Computer Science, 2005.

[135] Koch, R. ve Spenke, I., "Complexity and Approximability of k-splittable Flows," Theoretical Computer Science, 2006.

[136] Eltayeb, M., Doğan, A. ve Özgüner, F., "Concurrent Scheduling: Efficient Heuristics for Online Large-Scale Data Transfers in Distributed Real-Time Environments," IEEE Transactions on Parallel and Distributed Computing, No: 17(11), 1348- 1359, 2006.

[137] Kleinberg, J., Approximation Algorithms for Disjoint Paths Problems, Ph.D Thesis, Dept. Of EECS, MIT, 1996.

[138] Dunn, D.A., Grover, W.D. ve MacGregor, M.H., "Comparison of kshortest paths and maximum flow routing for network facility restoration," *IEEE Journal on Selected Areas in Communications*, No: 12(1), 88–99, 1994.