Paralel Bilgisayarlar İçin Dağıtık Dinamik Yük Dengeleme

Muammer Akçay

DOKTORA TEZİ

Elektrik Elektronik Mühendisliği Anabilim Dalı

Mart 2008

Distributed Dynamic Load Balancing For Parallel Computers

Muammer Akçay

Ph.D. THESIS

Department of Electrical Electronics Engineering

March 2008

Paralel Bilgisayarlar İçin Dağıtık Dinamik Yük Dengeleme

Muammer Akçay

Eskişehir Osmangazi Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Elektrik Elektronik Mühendisliği Anabilim Dalı
Elektronik Bilim Dalında
DOKTORA TEZİ
Olarak Hazırlanmıştır

Danışman: Yrd.Doç. Dr. Nihat Adar

Mart 2008

Muammer AKÇAY' ın DOKTORA tezi olarak hazırladığı "Paralel Bilgisayarlar için Dağıtık Dinamik Yük Dengeleme" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

Üye : Yrd. Doç. Dr. Nihat ADAR .....

Üye : Prof. Dr. Atalay BARKANA .....

Üye : Yrd. Doç. Dr. Erol SEKE .....

Üye : Doç. Dr. Osman PARLAKTUNA .....

Üye : Yrd. Doç. Dr. Selçuk CANBEK .....

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

Prof. Dr. Abdurrahman KARAMANCIOĞLU

Enstitü Müdürü

ÖZET

Yüksek hesaplama kapasitesi olan heterojen küme bilgisayarlar, COTS ve/veya yüksek performanslı sunucular kullanılarak yapılan ekonomik paralel sistemlerdir. Paralel / dağıtık bilgisayarların birbirleriyle iletişiminde Internet protokolü üzerinde çalışan MPI ve PVM kütüphaneleri kullanılmaktadır. Gereken yazılımlar yüklendikten sonra uç bilgisayarlar yalnızca paralel / dağıtık / Grid bilgisayar sisteminin bir parçası olarak kullanılabilir. Küme veya Grid bilgisayarı oluşturan uç bilgisayarların hem sistemin bir parçası olarak kullanılabilmesi hem de bireysel olarak kullanılabilmesi çok önemlidir. Paralel, küme, Grid bilgisayar sistemlerinde herhangi bir uç bilgisayarın eklenmesi / sistemden çıkarılması kolay değildir. Bu çalışmada Web servis tabanlı Grid sistem mimarisi geliştirilmiştir. Geliştirilen mimari simülatör ile test edilmiş Microsoft .NET ortamında gerçeklenmiştir. Geliştirilen mimari ile uç bilgisayarlar Grid sistemine kolayca dahil olabilirler ve istedikleri zaman da ayrılabilirler. Uç bilgisayarlar için kayıt ve takip mekanizması yoktur. Uç bilgisayarlar sisteme dahil olduklarında yapılacak iş varsa alıp çalıştırırlar. Uç bilgisayarlar sistemden herhangi bir anda ayrılabilirler ve kendilerine daha önceden atanan işler diğer uç bilgisayarlar tarafından tamamlanır. Kullanıcılar tarafından verilen işler tek bir sunucudan değil de sisteme dahil herhangi bir uç bilgisayardan verilebilir. Bütün kullanıcılar sisteme iş verebilir. Farklı iş çizelgeleme yöntemleri desteklenmektedir. Geliştirilen mimarinin diğer sistemlerle entegrasyonu vardır. Birden fazla ağda çalışabilmektedir. Geliştirilen sistemde donanım, işletim sistemi, platform kısıtı olmadığından heterojen yapılar desteklenmektedir.

Anahtar Kelimeler: Paralel, küme bilgisayarlar, dağıtık sistemler, Grid hesaplama yük dengeleme, Web servis

SUMMARY

Heterogeneous Cluster Computers with high computational power are economic Parallel systems by using COTS and/or high performance server computers. MPI and PVM libraries on the Internet use to communicate parallel / distributed computers among themselves. Node computers which are loaded with required software can only used to be a part of parallel / distributed / grid system. It is important that node computers which are a part of cluster or grid computers can be used for both individual and a part of the system. It is not easy adding or removing any arbitrary node computer in parallel, cluster, grid computer systems. In this study, Web service based grid system architecture is developed. Developed architecture is tested by simulator and realized on Microsoft .NET environment. Node computers with developed architecture can be easily included grid system and leave at any time. There is no registration and monitoring mechanism for node computers. If there is a job to be executed when node computers are included to the system, they will take the job and execute it. Node computers can leave at any time from the system and those previously assigned jobs are completed on the rest of the computers. Although jobs giving by users cannot be submitted from a single server, they can be given from any node computers. All users can submit a job. Different job scheduling mechanisms are supported. Developed architecture has integration with other systems. It works on more than one network. Since there is no hardware, operating system and platform constraint, heterogeneous architectures are supported.

Keywords : Parallel, cluster computers, distributed systems, grid computing, load balancing, Web service

TEŞEKKÜR

Doktora çalışması süresince, gerek derslerimde ve gerekse tez çalışmalarında, bana danışmanlık ederek, beni yönlendiren, motive eden ve her türlü imkânı sağlayan danışmanım Yrd.Doç. Dr. Nihat Adar' a çok teşekkür ederim. Bana yardımcı olan herkese teşekkürler. Ayrıca bana her zaman destek olan **aileme** ve **eşim Berna'**ya da çok teşekkür ederim.

İÇİNDEKİLER

İÇİNDEKİLER (devam)

İÇİNDEKİLER (devam)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

BÖLÜM 1

GİRİŞ

Matematik, fizik, kimya gibi doğa bilimlerinde yüksek hesaplama gücü ve büyük verilerle çalışmayı gerektiren problemleri içeren konulara ilginin artması, bu problemleri çözebilecek uygulamaları çalıştırabilecek ortamlara ilginin artmasına ve yüksek hesaplama gücü olan sistemlerin kurulmasına sebep olmaktadır. Bu problemleri çözmede kullanılan sayısal hesaplama yöntemleri yüksek hesaplama gücü (işlemci kapasitesi) ve en kısa sürede sonuçların alınmasını istemektedir. Yüksek işlemci kapasitesi için süper/paralel bilgisayarlar kullanılmaktadır. Bunların donanım maliyetleri oldukça yüksektir, kurulması zaman alır ve bakımları maliyetlidir. Bilgisayarların yaygın kullanılmasının sonucu olarak masaüstü, iş istasyonu, sunucu, küme bilgisayarların kullanımları artmaktadır.

İnternetteki hızlı gelişim aynı yaklaşımın daha büyük ölçeklerde yapılmasına olanak sağlamaktadır. Son beş yıl içinde de çok daha büyük ölçekli sanal bilgisayarın (Grid) geliştirilmesi konusundaki çalışmalarda önemli mesafeler alınmaktadır (Chien, 2003). Ev ve iş yerlerindeki kişisel bilgisayarlar (PC'ler) kapasitelerinin çok altında (tipik olarak işlem gücünün onda biri) kullanılmaktadır. İlave olarak internete bağlı masaüstü bilgisayarların iş ya da evde kullanım zamanları dışında atıl beklemesi de bu konuda yapılan çalışmalara destekleyici yönde motivasyon oluşturmaktadır (Mutka, 1991).

Eldeki bilgisayar kaynaklarının etkin, kolay ve hızlı bir şekilde kullanımı Grid hesaplama yöntemi ile mümkün olmaktadır. Grid'i oluşturan bilgisayarlar heterojen (değişik hız, kapasite, model, vs) ve fiziksel olarak farklı yerlerde olabilirler (Foster, 2002). Başka bir ifade ile Grid'deki uç bilgisayarlar farklı yerel ağlara bağlıdır. Grid sistemini yönetecek yapılar önem arz etmektedir. Fiziksel olarak değişik coğrafi konumlarda olan, farklı zamanlarda atıl bekleyen işlemci kaynaklarını bir problemin bir parçasını çözmede kullanmak için kaynak yönetim mekanizmaları gereklidir (Krauter, 2002; Ibm). Uç bilgisayarda bir programın çalışabilmesi için gereken donanım ve yazılımın olması da gereklidir. Grid'i oluşturmak için kaynak alışverişi, kaynak tahsisi, servis kalitesi gibi alt seviye servisler gereklidir. Ayrıca uygulama geliştirme, kaynak

yönetimi, çizelgeleme gibi yüksek seviye servisler de gerekmektedir (Buyya, 2005).

Grid hesaplama beraberinde; heterojenlik, kaynak yönetimi, hata yönetimi, güvenilirlik, çizelgeleme ve güvenlik problemlerini de getirmektedir. Bu alanda yapılacak çalışmaların bu problemlere de bir çözüm önermesi gerekmektedir (Buyya, 2002). Grid sistemini oluşturmak için özel olarak geliştirilmiş yazılım araçlarının (Globus) uç bilgisayarlara yüklenmesi ve kullanılması gerekmektedir. Ancak son yıllarda gelişmekte olan web servis yazılımlarının daha önce geliştirilen yazılım araçlarının (Globus) yerini alması yönünde çalışmalar başlatılmaktadır. Hatta IBM, Microsoft, Platform, Sun, Avaki, Entropia, ve United Devices gibi firmalar bu konuda "*Open Grid Services Architecture* (OGSA)" (Foster 2002, OGSA) desteklemektedir. Bu çalışmaların bilgisayar işletim sistemlerine hazır yüklü Grid protokolleri olarak ortaya çıkması, Grid üzerindeki çalışmaları yaygınlaştıracak ve herkesin kolayca erişip kullanımına zemin hazırlayacaktır. Grid protokollerine taban oluşturacak yaklaşımların web servis yazılımları olacağı düşünülmektedir. Web servislerin Grid protokollerini oluşturmak üzere kullanımı konusunda yapılan pek az çalışma bulunmaktadır. Bu çalışmalardan Alchemi projesinde Globus Grid sisteminin bir alt sistemi olarak çalışan bir yapı, web servis yaklaşımı ile geliştirilmiştir (Luther 2005). Ancak geliştirilen yöntemde iş dağıtımı ve takibi merkezi bir yönetici tarafından yapılmaktadır. Yönetici modülü bu merkezi kontrol yardımıyla; her an sistem durumu, kullanıcı istekleri ve alt sistemin her bir kaynağının kullanımı hakkında detay bilgiye sahiptir. Bu da alt sistemin tıkanıklık noktasını oluşturmaktadır. Böylesi bir yapı oldukça yoğun bir ağ trafiğinin oluşmasına neden olmaktadır. Yönetici modülün olduğu noktada olası bir hata (donanım, yazılım veya ağ hatası) sistemi tamamen kullanılamaz hale getirecektir. Merkezi yöneticili sistemlerde yeni bir bilgisayarın sisteme eklenmesi veya çıkarılması yönetici modülüne kayıt yaptırma veya sildirmeyi gerektirmektedir.

Grid sistemi dağıtık ve heterojen bilgisayarlardan oluştuğundan, elde edilecek performansı arttırmak için kaynakların verimli ve etkili kullanılması gereklidir. Grid'in verimli kullanılması için eldeki bütün kaynaklar (bilgisayarlar, işlemciler, hesaplama kabiliyetleri) kullanıcılar tarafından kullanılmalıdır. Grid oluşturmak için harcanan parayı en iyi kullanmak için bunu bir yatırım olarak görmek ve en fazla karı elde etmek için çalıştırmak gereklidir. Grid hesaplama gücü Grid kaynak sağlayıcılar tarafından karşılanmakta ve kullanıcılar tarafından kullanılmaktadır. Kullanıcılar ayrıca bir iş

yaptırmak için bir bedel (ücret) ödemektedirler. Genelde Grid kaynak sağlayıcılar ve kullanıcılar zaman için de değişmektedir. Hesaplama gücünü sağlayanlar en fazla paraya satmak isterler. Alan kullanıcılar ise bu gücü en ucuza almak isterler.

Grid kaynak yönetimi, elde bulunan kaynakları kullanıcıların verdiği işler için nasıl kullanacağını çözerken, hem kaynak sağlayıcıların hem de kullanıcıların ölçütlerini yerine getirmeye çalışır. Grid kaynaklarının kullanılabilirliği her an değişmektedir. Kullanıcılar değişik zamanlarda kaynak kullanım taleplerinde bulunabilir. Kaynak sağlayıcılar ellerindeki bütün kaynakları en çok para verene satarak en çok kar elde etmeye çalışırlar. Kullanıcılar ise işlerini yaptırabilecek kaynak sağlayıcılarını en ucuza alarak en kısa sürede işlerin tamamlanmasını veya işlerin istenen tamamlanma süresinde (deadline) bitmesini isterler.

Kaynakların etkili yönetimi ve uygulamaların çizelgelenmesi (hangi sırada ve nerede yapılacağı) Grid ekonomisi ile yapılmaktadır (Buyya, 2005). Literatürdeki ekonomik modeller: *commodity market models*, *posted price models*, *bargaining models*, *tendering*, veya *contract-net models*, *auction models*, *bid-based proportional resource sharing models*, *cooperative bartering models*, *monopoly* ve *oligopoly*'dir.

Son zamanlarda yapılan çalışmalar ajan tabanlı modeller üzerinde yapılmaktadır. Ajanlar yapılacak işler ile işlemciler arasındaki bağlantıyı sağlar. Her bir işin farklı şartlardaki çalıştırılmaları için değişik ajanlar tanımlanır. Bu çalışma şartları işlerin değişik işlemcilerde çalıştırılması olabileceği gibi, değişik zamanlarda aynı veya farklı işlemcilerin kullanılması da olabilir. Böylece bir işi yapabilmek için birden çok seçenek olacaktır. Her seçenek beraberinde kullanıcıya farklı maliyet, kaynak sağlayıcıya farklı kar (kazanç) getirecektir. Bunlar arasından birine karar vermek gerekmektedir. Bunun için pazarlık (*Bargaining*) modelleri geliştirilmektedir. Pazarlık modelleri temelde iki seçenek arasından birine karar verir.

Grid kaynak sağlayıcılar, kullanıcılar, hesaplama gücü, kullanıcının ödeme gücü pazarı (*market*) oluşturmaktadır. Bazı modeller pazar şartlarının değişmediğini kabul ederek çözüm üretirler. Bazıları ise zamanla pazar şartlarının değiştiğini kabul etmektedirler.

Paralel bilgisayar sistemindeki işlemcilerden herhangi bir veya birkaçının beklendiği gibi çalışmaması (arızalanması) ve çalışmayan işlemcinin işlerinin başka işlemciler tarafından yapılmaması durumunda bu sistem hataya duyarlı değil (*fault-*

tolerant değil) denir.

Yük dengeleme algoritmaları statik ve dinamik yük dengeleme olarak ikiye ayrılır. Statik yük dengelemede, işler yük dengeleme en iyi olacak şekilde işlemcilere yürütülmeden önce dağıtılır. Her işlemci kendisine verilen işleri yürütür. İş verildikten sonra işlemcinin görevi tamamlayacağı beklenir. Bu işler yürütülürken işlerin bir işlemciden bir diğerine aktarılması söz konusu değildir. İşlerin dağıtılması işlemi bir defa yapılır. Her işlemci hangi işi yapacağını önceden bilir. Statik yük dengeleme yapabilmek için işin ne kadar sürede biteceğinin işe başlamadan önce bilinmesi gereklidir. Her işlemci beklendiği gibi işleri yürütüp belirtilen süre sonunda verilen işleri tamamlarsa her şey yolunda olacaktır. Ancak, verilen işler sebebi ne olursa olsun işlemcilerde istenen zamanda tamamlanamazsa dengeli bir yük dağılımı olmayacaktır. Bu durumda bazı işlemciler çalışırken bazıları da boşta bekleyecektir.

Statik yük dengelemenin aksine dağıtık sistemlerden esinlenerek paralel bilgisayar sistemine herhangi bir anda yeni görevler eklenebilirse buna dinamik yük dengeleme denir. Bu durumda yük dengeleme problemi, boşta bekleyen işlemci kalmayacak şekilde işleri mümkün olduğu kadar eşit dağıtmak ve verilen işleri en kısa sürede tamamlamaktır. Dinamik yük dengeleme merkezi ve merkezi olmayan olarak ikiye ayrılır. Merkezi dinamik yük dengelemede yapılacak işler tek bir merkezden verilir. Burada usta-çırak (*Master-Slave*) yapısı vardır. Dinamik yük dengelemedeki önerilen çözümlerin çoğu merkezi olmayan algoritmalardır (Heiss, 1995).

Buraya kadar Grid, dağıtık, paralel sistemler kısaca anlatılmış, bundan önce yapılan çalışmalar özetlenmiştir. Bir sonraki bölümde problemin tanımı yapılacaktır.

1.1 Önerilen Yaklaşım

Bu çalışmada yük dengeleme dinamik olarak yapılacaktır. Ayrıca her işlemci kendi iş yükünü takip edecek ve gerekirse ek iş alabilecektir.

Küme bilgisayar mimarisi birbirinden bağımsız bilgisayarları ağ sayesinde bir büyük paralel bilgisayar ya da sanal bir süper bilgisayar gibi kullanılmasını sağlar. Geleneksel küme bilgisayarlar yerel ağlarda çalışan sanal paralel bilgisayarlar olarak kullanılmaktadır. Bu bilgisayarların üzerinde çalıştırılacak paralel program, geliştirme araçları PVM (Pvm), MPI (Mpi) ile geliştirilerek kullanılmaktadır.

Bu çalışmada tanımlı bir iş grubunu WEB servisleri (Bray, 1998; Brittenham, 2001; Chamberlin, 2001; Christensen, 2001; Humphrey, 2005a; Humprey, 2005b; Wasson, 2005; Web1, 2004; WSDL) kullanarak yerel ağa bağlı heterojen bilgisayarlara (PC, sunucu, Workstation, vb.) dağıtarak çalıştırması için bir model geliştirildi. Tanımlanan modelde (daha önce geliştirilen merkezi yönetim ve dağıtım modellerinden farklı) işler herhangi bir uç bilgisayar üzerinden sisteme verilebilmektedir. Böylece Grid kullanıcılarının kolaylıkla sisteme iş vermesi mümkün olabilmektedir. Yine önceki sistemlerden farklı olarak sisteme yeni bilgisayarların dahil edilmesi veya çıkarılması gibi süreçlerin kolaylaşması için işlerin dağıtımı, çalıştırılması ve takibi dağıtık karar mekanizması ile gerçekleştirilmektedir. Böylece sisteme ek bilgisayarların girmesi/çıkması ve bunların model sayesinde standart COTS bilgisayarların hem bireysel kullanıcısı hem de paralel bir sistemin parçası olarak kullanılması mümkün olmaktadır. Bu tip sistemlerde en önemli kısıtlardan birisi de sistemin bir parçası olan ve kullanılmakta olan bir bilgisayarda veya bilgisayarlarda hata oluşması durumudur. Bu durumda sisteme verilen işin tamamlanabilmesi önemlidir. Tanımlanan modelde yerel ağa bağlı COTS bilgisayarlara verilen tüm işlerin o işi yapabilecek en az bir bilgisayar olması halinde tamamlanacağı garanti edilmektedir. Tüm bunların ötesinde böyle bir sistemin yaygın kullanımı için tasarlanan model günümüzde neredeyse tüm COTS bilgisayarlarda desteklenen bir ara yüz olan WEB servis mimarisi ile çalışmaktadır. Bu çözüm mimarisi ile işletim sisteminden bağımsız heterojen bilgisayarlar arasında bilgi değişimi mümkün olabilmektedir. Yerel ağa bağlı heterojen (işletim sistemi ve/veya donanım) bilgisayarlar üzerinde çalışan, onları paralel sistemin parçası olarak kullanıp iş dağıtımı yapan, hataya (bir bilgisayarın hata yapması) dayanıklı web servisler kullanılarak modellenen çalışma, Web servis tabanlı Grid sistemi, WGS, olarak adlandırılmaktadır.

Bu çalışmada önerilen yaklaşım ile elde edilen katkılar şu şekilde listelenebilir:

- *Modülerlik:* Sisteme bağlı uç bilgisayarlar herhangi bir anda sisteme dahil olabilir veya sistemden ayrılabilirler. Sistemin dinamik bir yapısı vardır. Herhangi bir anda sistemde kimlerin bulunduğunun kaydını tutan merkezi bir birim yoktur.
- İşler sisteme herhangi bir uç bilgisayar üzerinden gelebilir. Sistem dışından

verilecek işler Dış Erişim bilgisayarı üzerinden verilebilmektedir.

- *Hataya Duyarlılık:* Sistemde çalışan ve verilen işleri yapabilecek en az bir bilgisayar olduğu sürece verilen tüm işler tamamlanacaktır.
- *Dağıtık yük dengeleme:* Sisteme dahil olan her bir uç bilgisayar kendi iş yükünü takip edecektir. İş yükü belirli bir eşik değerinin altında ise daha fazla iş talebinde bulunacaktır. İş yükü fazla olan ise yükünü başkasına verecektir. Bu yaklaşım ile dağıtık yük dengeleme yapılmıştır.

Tezin ilerleyen bölümlerinde şu konular yer almaktadır. Bölüm 2'de dağıtık bilgisayar sistemleri verilmektedir. Bölüm 3 önerilen yaklaşımla ilgili heterojen küme bilgisayarların web servislerle modelleme detaylarını içermektedir. Önerilen modelin uygulanması ve elde edilen sonuçlar bölüm 4'de verilmektedir. Son olarak, bölüm 5'te yapılan tez çalışması ile ilgili genel sonuçlar tartışılmaktadır.

BÖLÜM 2

DAĞITIK BİLGİSAYAR SİSTEMLERİ

Bu bölümde ilk olarak açık Grid servis mimarisi (*Open Grid Service Architecture*, OGSA) (Foster, 2002; Li 2005; Cunha 2006) detaylı açıklanacaktır. Daha sonra dağıtık sistemin genel özellikleri verilecektir. Dağıtık sistemlerden küme, Alchemi (Yang 2006), Digipede (Luther, 2005), JISGA (Huang, 2003), High Availability sistemler (Trivede, 2006) ve dağıtık sistemin özellikleri ile tartışılacaktır. Daha sonra ideal dağıtık sistemin özellikleri özetlenecektir. OGSA Grid sistemlerine çerçeve oluşturduğu için bir sonraki bölümde detaylı anlatılacaktır.

2.1 OGSA

OGSA Web servis tanımlama dili (*Web Service Description Language*, WSDL) ara yüzleri ile tanımlanır. OGSA karmaşık dağıtık sistemlerin oluşturulması ve birleştirilmesi için gerekli yöntemleri, yaşam boyu yönetimi, yönetim değişimi ve bildirmeyi tanımlar. OGSA'da her şey sanaldır.

Bundan sonra sırasıyla OGSA Servis modeli, geçici servis oluşturma, servis ömür yönetimi, referans ve *handle* yönetimi, servis verileri ve veri bulma bildirme ve yönetim değiştirme alt bölümlerinde OGSA ile ilgili detaylı bilgi verilecektir.

2.1.1 OGSA Servis Modeli

OGSA'nın temel dayanağı her şeyin servis ile temsil edilmesidir. Hesaplama kaynakları, depolama kaynakları, ağlar, programlar, veri tabanlarının tamamı servislerdir. Web servisi tanımlanan kurallara uyar ve sürekli yönetim amacı için standart ara yüzleri destekler.

Web servisler önerdikleri yeteneklerle karakterize edilir. Bir web servisi bir veya daha fazla ara yüzü uygular. Her bir ara yüz tanımlı mesaj dizilerinin değişimi ile çağrılan işlemler kümesini tanımlar. Grid servis ara yüzleri WSDL' deki *portType* ile uyumludur. Grid servisin sürümü ile ilgili ek bilgi servisin *serviceType*'ında belirtilir.

Grid service olayı (*Grid service instance*) bir Grid servisin belirli bir örneklemesini göstermektedir.

Dağıtık sistemler bileşen hatasına yatkındır. Dağıtık sistemlerde hiç kimse gönderilen mesajın ulaştığını garanti edemez. OGSA'da servisler dinamik olarak oluşturulur ve ortadan kaldırılır. Servisler açıkça veya servislere bazı sistem hatalarının (işletim sistemin çökmesi, network hatası) sonucu olarak ulaşılamadığında ortadan kaldırılmaktadır. Dinamik olarak oluşturulan servisleri birbirinden ayırt etmek için GSH (*Grid Service Handle*) ile her bir Grid servis olayı için tek bir isim atanmaktadır. Bir sonraki bölümde servis oluşturma açıklanacaktır.

2.1.2 Geçici Servis Oluşturma

OGSA yeni Grid servis olaylarını oluşturan ara yüzleri gerçekleyen Grid servisler sınıfını tanımlar. Yeni servis olaylarını oluşturan ara yüzler *Factory* olarak adlandırılmaktadır. *Factory* ara yüzünün *CreateService* işlemi istenen Grid servisi oluşturmaktadır. *Factory* ara yüzü GSH (*Grid Service Handle*) ve yeni servisin başlangıç GSR (*Grid Service Reference*)'sini geri göndermektedir. *Factory* ara yüzü servisin nasıl oluşturulduğunu açıkça belirtmez. *Factory* ara yüzünün uygulandığı ortamlar (.NET veya J2EE gibi) yeni servis oluşturmak ve yönetmek için standart yöntemleri sağlamaktadır. Oluşturulan servisin yaşam süresi bir sonraki bölümde açıklanmaktadır.

2.1.3 Servis Ömür Yönetimi

Normal çalışma şartlarında, geçici servis olayı belirli bir işi yapmak için oluşturulmaktadır. Bu işin tamamlanmasıyla, istemci isteği ile veya başka bir servisin isteği ile servis sonlandırılmaktadır. Dağıtık sistemlerde bileşenler hata yapabilmekte ve mesajlar kaybolabilmektedir. Beklenen açık sonlandırılma isteğini servis hiçbir zaman göremediğinde, kaynakların belirsiz bir şekilde tüketimine sebep olmaktadır.

Grid servis olayları belirli bir yaşam ömrü ile oluşturulduğu yerlerde OGSA *soft state* yaklaşımı ile bu problemi adreslemektedir. Başlangıç ömrü belirli bir kullanıcı veya onun yerine davranan bir Grid servis için belirli bir süre uzatılabilmektedir.

İstemcinin isteği ile yeniden onay alınmadan bu süre biterse, servisi bulunduran ortam veya servis olayı, servis olayını sonlandırmakta ve ilgili kaynakları serbest bırakabilmektedir.

Grid servis ömür yönetiminin istenen iki özelliği:

1. İstemci Grid servisin ne zaman sonlanacağını bilir veya tayin eder. İstemci bu bilgi ile güvenilir bir şekilde web servisin ne zaman sonlanacağını, kaynakların ne zaman kurtarılacağını hatta sistem hatalarını (sunucu, ağ, istemci) bilir. İstemci tam olarak son durum isteğinin ne kadar süreceğini servisten veya servis ömrü uzatma isteğinden bilir. İstemci bir sistem hatası oluştuğunu bilirse sonlandırma zamanından sonra servisle temasa geçmeye gerek kalmaz. Bu sonlandırma süresinden sonra ilgili kaynaklar serbest bırakılmaktadır. Ancak bu istemciyi takip eden varsa bu işlem yapılmamaktadır. Özetle, kaynak yönetimi gürbüz sonlandırmayı ve hata bulmayı yetkin kılmaktadır.
2. Kontrol dışı bir sistem hatası olsa bile, sunan ortam kaynak tüketimini sınırlamayı garantilemektedir. Servis sonlanma zamanına ulaşılınca, sunan ortam verdiği ilgili tüm kaynakları geri istemektedir.

Soft state ömür yönetimi *SetTerminationTime* işlemi ile gerekli *GridService* ara yüzü içinde yapılmaktadır. *SetTerminationTime* işlemi yeni servis olayı için başlangıç ömrü müzakeresi (istemcinin en az ve en çok kabul edilebilir başlangıç değerleri arasından seçilir), ömür uzatma isteği (istemcinin en az ve en çok kabul edilebilir yeni ömür uzatma değerleri arasından seçilir), ve servisin ömrü bittiğinde servis olaylarını toplama için gereken işlemleri tanımlamaktadır. İstemcinin ömür uzatma istekleri zorunlu değildir. Servis gelen istekleri cevaplamada kendi kurallarını uygulayabilir. *SetTerminationTime* işleminde NTP (*Network Time Protocol*) kullanılarak en çok 10 milisaniyeler derecesinde saatleri senkronize edilmektedir. Bu da ömür yönetimi için yeterlidir.

2.1.4 Referans ve Handle Yönetimi

OGSA'da *HandleMap* ara yüzü ile yapılan işlem GSH' yi alarak geçerli GSR' yi geri döndürmektedir. Bu haritalandırma işlemi ulaşım kontrollüdür ve reddedilebilir. *HandleMap* ara yüzünün gerçeklenmesi var olan Grid servis olaylarını ve sonlandığı bilinen fakat referansları geri döndürülmeyen olayları takip etmek isteyebilir. Bununla beraber geçerli GSR' ye sahip olma Grid servis olayına bağlantı kurmayı garanti etmemektedir.

2.1.5 Servis Verileri ve Servis Bulma

Her Grid servis olayı tip ve yaşam süresi bilgisini içeren servis veri seti ile tanımlanmaktadır. Grid servis ara yüzü zorunlu olarak servis veri sorma ve erişme için standart WSDL işlemi (*FindServiceData*) tanımlamaktadır.

Grid servis *Registry* olarak adlandırılan servis bulmayı desteklemektedir. *Registry* ara yüzü GSH' yi kaydetmek için ve *GridService* ara yüzünün *FindServiceData* işlemi kayıtlı GSH hakkında bilgileri almada kullanılmaktadır.

2.1.6 Bildirme

OGSA bildirme çerçevesi istemcilerin bildirilen belirli bir mesajı kaydetmeye (*NotificationSource*) ve bu bildirilerin asenkron tek-yönlü dağıtımını (*NotificationSink*) desteklemektedir. Bir servis bildiri mesajlarının üyeliğini desteklemek isterse, üyelikleri yönetebilmek için bildiri mesajlarının dağıtımında kullanılan *NotificationSink* ara yüzünü gerçekleştirmek zorundadır. Alıcı periyodik olarak *keepalive* mesajları göndererek bildiri alma isteğini gönderirken bildiri mesajları kaynaktan alıcıya doğru akmaktadır. Eğer güvenilir dağıtım istenirse, bu servis için uygun protokol bağlama gerçeklenebilmektedir.

2.1.7 Yönetim Değiştirme

Grid servisleri bulma ve yönetim değişikliklerini desteklemek için, Grid servis ara yüzleri tek ve küresel isimlendirilmektedir. WSDL'de ara yüz p*ortType*'ın *qname* ile global ve tek olarak isimlendirilir. Grid servisin tanımında yapılan herhangi bir değişiklik (ara yüzün değiştirilmesi veya işlemlere anlamsal önemli değişiklikleri yapmak için) yeni ara yüz isimlerine (yeni *portTypes*, *ServiceTypes* gibi) yansıtılmalıdır.

Özetle OGSA da her şey Grid servis ile tanımlanmaktadır. Servis ömür yönetimi, karakterlerin bulunması, bildirim gibi amaçlar için WSDL kullanılarak ifade edilen sözleşmelere Grid servis uymaktadır. Grid servis gerçeklemeleri var olan teknoloji altyapılarını veya yerli imkânların bir araya getirilmesini hedef almaktadır. Grid servisleri oluşturma, kaydetme, bulma ara yüzleri değişik sanal organizasyonları oluşturmak için ayarlanabilmektedir. OGSA'da tüm çevre elemanları sanaldır.

2.2 Dağıtık Sistemin Özellikleri

Bu bölümde dağıtık sistemin genel özellikleri maddeler halinde aşağıda açıklanmıştır.

- Dağıtık sistemlere verilen problemin *grain size* (*fine, medium*, *coarse grain*)

Dağıtık sistemlerdeki *grain size* hangi seviyede paralellik yapacağının bir ölçüsüdür. *Fine-grain* (küçük taneli) *instruction* seviyesinde verilen işlerin paralel çalıştırılmasıdır. Paralel bilgisayarlarda *fine-grain* paralellik görülmektedir. *Coarse-grain* (büyük taneli) program seviyesinde paralelliktir. *Coarse-grain* paralellikte aynı anda farklı programlar farklı işlemcilerde çalışmaktadır. Grid sistemlerinde *coarse-grain* paralellik daha uygun olmaktadır. Bir yerel ağda çalışan sistemlerde (küme bilgisayarlar) *medium-grain* paralellik uygulanmaktadır. *Medium grain* paralellik *fine-grain*'deki *instruction* ile *coarse-grain*'deki program seviyesi arasında bir paralelliktir. *Medium grain* paralellik subroutine, fonksiyon, bir *instruction* grubunun aynı anda çalışmasıdır.

- İşleri dağıtma yöntemi (merkezi, dağıtık)

Dağıtık sistemlerde kullanıcılar tarafından sisteme verilen işlerin çalıştırılacakları işlemcilere dağıtma şeklini ifade etmektedir. Merkezi iş dağıtmada işler tek bir işlemciye (sunucuya) verilmektedir (Digipede, Alchemi, Küme bilgisayar). Sunucu üzerindeki iş dağıtma mekanizması işleri çalıştırılacakları işlemcilere göndermektedir. Merkezi yapılarda verilen işlerin hangi aşamada oldukları, tamamlanıp tamamlanmadığı gibi iş durumları da merkezde (sunucuda) takip edilmektedir. Dağıtık sisteme verilen işler önerdiğimiz çalışmada olduğu gibi birden fazla noktadan da dağıtık olarak işlemcilere dağıtılabilmektedir. İşler işlemcilere herhangi bir işlemci üzerinden verilebilmektedir. İşlerin çalışma anındaki aşamaları takip edilmemektedir.

- Dağıtık sistemlerin gerçekleştirme şekli (MPI, PVM, ajan tabanlı, web servis)

Önerilen yaklaşımların gerçekleme şekilleri farklılık göstermektedir. Küme yapılarda işlemciler arasındaki iletişim MPI / PVM kütüphaneleri ile yapılmaktadır. Bazı uygulamalarda ajan tabanlı gerçekleme yapılmıştır. Sisteme verilen işlerin alınması, dağıtılması, yürütülmesi ajanlar ile yapılmaktadır (Digipede). Bazı uygulamalarda verilen işler altyapıya uygun sınıf kodlarına çevrilmekte, daha sonra çalıştırılacak işlemcilere verilmektedir (Alchemi, JISGA). Alchemi de dış ortamlara erişim için web servisleri kullanılmıştır. Önerdiğimiz model ile işlerin alınması, dağıtılması, işlemciler arası iletişim web servislerle gerçeklenmiştir.

- Hataya duyarlılık (*Fault Tolerance*)

Sistemi oluşturan işlemcilerden bir veya birkaçında hata oluştuğunda sisteme verilen işlerin nerede ve nasıl tamamlandığı incelenmektedir. Paralel ve dağıtık sistemlere verilen işlerin tamamlanması istenmektedir. Sistemde oluşabilecek herhangi bir hatanın işlerin tamamlanmasına engel olması istenmemektedir. Bu yüzden sisteme dahil işlemcilerin durumları, işlerin öngörülen zamanda tamamlanıp tamamlanmadığı kontrol edilmelidir. Sisteme dahil herhangi bir birimde yada işlemcide hata olduğunda bunun giderilmesi, varsa yapmakta olduğu görev ve işlerin başkaları tarafından yapılması sağlanmalıdır. *High Availability* sistemler bu konuda hatalara daha duyarlıdır. Olası bir hata durumunda, hatalı birim (güç kaynağı, hard disk, işlemci, network, vs.)

yerine bir başka işlemcinin o işi yapması sağlanmaktadır. Dağıtık ve Grid sistemler genelde esnek yapılardır. Oluşabilecek bir hatada başka alt sistemlerin hatalı birimin işlerini üstlenebilecek kontrol, takip ve yeniden iş atama mekanizmalarının oluşturulması gerekmektedir. İşler merkezi birim tarafından dağıtılıyorsa, aynı zamanda bu merkezi birim işlerin takibini de yapmaktadır.

- İşlerin sisteme verilme şekli (tek bir noktadan veya çoklu noktadan)

Kullanıcılar sisteme iş verebilmek için merkezi yapılarda sunucuya bağlanıp iş vermektedirler. Diğer bazı uygulamalarda kullanıcılar herhangi bir işlemci üzerinden sisteme iş verebilmektedir.

- Diğer sistemlerle bütünleşmesi, ortak kullanımı

Kullanılan sistemin diğer sistemlere bağlantısının olup olmadığı, başka sistemlere iş verme ve iş almanın araştırılmasıdır. Alchemi'de dış erişim birimindeki web servisler aracılığı ile diğer Grid sistemlerine erişilmektedir. Önerilen yapı ile web servisler kullanılarak diğer dağıtık sistemlerle çalışılması desteklenmektedir.

- Uç (düğüm, node) bilgisayarların tahsisli veya isteğe bağlı kullanılabilmesi

Küme bilgisayarı oluşturan işlemciler sadece bu amaç için kullanılabilir. Uç bilgisayar küme yapıya bir kez eklendikten sonra kolayca ayrılması mümkün değildir. Eğer işlemci üyesi olduğu sistemden ayrılmak istiyorsa yapının yeniden kurulması gerekmektedir. Bu şekilde bağlı olan işlemciler tahsisli (*dedicated*) olarak çalışmaktadır. Bazı sistemlerde işlemciler istedikleri zaman sisteme dahil olabilir, istedikleri, zamanda ayrılabilmektedir. Bazı uygulamalarda (Alchemi) işlemciler ekran koruyucuya bağlı olarak veya boş oldukları zamanlar önceden tanımlanarak sisteme dahil olabilmektedir. İşlemcilerin sisteme kolayca eklenip çıkabilmesi istenen bir özelliktir. İşlemcilerin isteğe bağlı olarak çalışmasında ayrılan işlemciye atanan işlerin diğer işlemciler tarafından nasıl tamamlanacağının tanımlanması gerekmektedir.

- İşlerin düğümlere (node'lara) verilme şekli (*push*: sunucu veriyor; *pull*: düğümler talepte bulunuyor)

Sisteme verilen işlerin çalıştırılacakları işlemcilere ne şekilde atanacaklarını tanımlar. İşler bir sunucu tarafından çalıştırılacakları işlemcilere verilebilir (*push*). Uç bilgisayarlar önceden kendi özelliklerini, ne kadar iş çalıştırabileceklerini merkezi iş dağıtma birimine bildirirler. Merkezi iş yönetim birimi işlemcilerin özelliklerine göre işleri dağıtır. *Pull* iş alma şeklinde işlemciler kendileri iş alır. Bazı dağıtık yapılarda işlemciler ajanlar veya programlar aracılığı ile kendileri iş talebinde bulunmaktadır (Digipede).

- Desteklediği işletim sistemleri ve platformlar

Bugüne kadar geliştirilen, önerilen modeller değişik platformlarda çalışmaktadırlar. Kullanılan işletim sistemleri Unix, Linux, Microsoft ve desteklenen platformlar Windows NT, Unix, J2EE, Microsoft .NET dir.

- Heterojen yapıların desteklenebilmesi

Geliştirilen bazı sistemler sadece homojen yapıları (Küme) desteklemektedir. Bazıları ise heterojen yapılara da destek vermektedirler. Heterojen yapılar sistemin büyüklüğünün değiştirilmesinde esnek çözümler sunmaktadırlar. Grid sistemlerin çoğu heterojen bilgisayarları desteklemektedir. Digipede işletim sistemi açısından heterojen yapıyı desteklemektedir.

- Düğümlerin birbirleriyle network bağlantıları: yüksek performanslı gevşek ve düşük performanslı gevşek

İşlemcilerin bir araya gelmesinden oluşan bu yapılar ağ bağlantı elemanları ile bağlanarak birbirleriyle haberleşirler. Küme bilgisayarı oluşturan işlemciler genelde aynı yerel ağ içinde birbirleriyle yüksek performanslı gevşek bağlıdır. Bir işlemciden diğerine bağlanmak direkt ya da ağ bağlantı elamanı (anahtar, hub) ile sağlanmaktadır. Dağıtık, Grid sistemler ise birden fazla farklı ağlarda olduğundan bilgisayarların birbirleriyle iletişimi için birden fazla ağ elemanını geçmeleri gerekecektir. Bu yapılarda bilgisayarlar düşük performanslı gevşek bağlıdır. İki bilgisayar arasındaki ağ bağlantı elamanı sayısı arttıkça, bilgisayarlar arası iletişim için daha fazla zaman gerekecektir.

- Birden fazla ağ da yer alabilme

Küme bilgisayarlar yerel ağda çalışmaktadırlar. Bu nedenle birden fazla ağı destekleyemezler. Dağıtık ve Grid yapıları ise birden fazla ağda çalışabilecek şekilde tasarlandıkları için birden fazla ağda yer alabilmektedirler.

- İşlerin çizelgeleme yöntemleri

Sisteme verilen işlerin hangi sırada ve nasıl çalıştırılacaklarını, işlemcilere nasıl verileceğini tanımlar. İşlerin çizelgelenmesi ilk gelen iş ilk verilecek (First Come First Serve), öncelik sırasına göre, en uzun işlere öncelik verme, en kısa işlere öncelik verme, tamamlanma süresi erken olana öncelik verme gibi farklı çizelgeleme yöntemleri kullanılabilmektedir.

- Güvenlik ve kullanıcı yönetimi

Kullanılan sistemlerde yerel ağda bir tek kullanıcı için kimlik doğrulama gerekmez. Sistemi kullanan birden fazla kullanıcı varsa, kullanıcı doğrulama yapılması gerekmektedir. Sistemi dış dünyaya açtığımızda kullanıcı doğrulama ve güvenli iletişim çözümlerinin sunulması gerekmektedir.

Bu bölümde dağıtık sistemin temel özellikleri kısaca özetlenmiştir. Bundan sonraki bölümlerde anlatılan modellerin dağıtık sistemin özelliklerini nasıl desteklediği de vurgulanacaktır.

2.3 Küme Bilgisayarlar

Küme bilgisayar bir yerel ağa bağlı masa üstü bilgisayarların bir arada kullanılabilmesine yardımcı olan yapıdır. Küme bilgisayarlar sunucu bilgisayar (*head node*) ve uç bilgisayardan oluşurlar. Küme bilgisayar için gereken programlar önceden kurulur. Bütün bilgisayarlar tahsisli olarak çalışmaktadır. Herhangi bir uç bilgisayarın istediği zaman eklenip çıkması kolay değildir. Her ekleme ve çıkarılma işleminde küme yapının güncellenmesi gerekmektedir. Kullanıcılar sunucu bilgisayara işleri

vermektedir. İşlerin dağıtılması, işlemcilerin ve işlerin takibi, sunucu bilgisayar tarafından yürütülmektedir.

Küme bilgisayarlar homojen bilgisayarların aynı yerel ağda yüksek performanslı gevşek ağ bağlantıları ile yapıldığından, *medium-grain* paralellik yapılabilmekte ve her uç bilgisayara MPI kütüphanelerinin kurulması gerekmektedir. Küme bilgisayarlar arasındaki iletişim MPI kütüphaneleri kullanılarak yapılmaktadır. MPI kütüphaneleri ile çalıştırılacak işlerin parçaları paralel çalışabileceği gibi, alt program ve program seviyesinde de paralellik yapılabilmektedir.

Küme bilgisayarlarda genelde işler tek bir bilgisayara (*head node*) verilir ve buradan merkezi olarak uç bilgisayarlara dağıtılmaktadır. Kullanıcılar tüm iş taleplerini sunucu (*head node*) bilgisayara verirler. Yazılıma göre değişmekle beraber genelde *head node* verilen işlerin takibini yapmaktadır.

Küme bilgisayarlar hataya duyarlıdır. Herhangi bir bilgisayarda hata oluşunca bunu anlayıp tekrar iş dağıtılabilecek şekilde programların yazılması gerekmektedir. Küme yapıda işlerin ve işlemcilerin durumlarını sürekli olarak takip eden bir yapı yoktur. Hatalı işlemciden dolayı yapılamayan işler verilen işin özelliğine göre diğer bilgisayarlardaki işleri etkileyebilir. Kullanıcıların oluşabilecek hataları kendileri gidermeleri gerekmektedir.

Diğer küme sistemler ile MPI kütüphaneleri seviyesinde bütünleşme yoktur. Verilen işler sadece bağlı bulunan küme bilgisayarda çalıştırılabilmektedir.

Küme bilgisayarda uç bilgisayarlar sadece küme bilgisayar için tahsisli olarak kullanılmaktadırlar. Küme bilgisayar için kullanılırken, uç bilgisayarlar küme bilgisayar haricinde başka bir amaçla kullanılamazlar. Uç bilgisayarlardan bir tanesinin ayrılması veya yeni bir uç bilgisayar eklenmesi küme bilgisayar yapısının yeniden yapılandırılmasını gerektirmektedir. Uç bilgisayarlar kolayca küme bilgisayara eklenip istedikleri zaman kolayca ayrılamazlar.

Küme yapılar, hem Unix işletim sistemleri hem de Microsoft işletim sistemlerinde çözüm sunmaktadırlar. Ayrıca açık kaynak kodlu çözümler Linux işletim sistemlerinde kullanılmaktadır.

Küme bilgisayarı oluşturmak için homojen (aynı tip) bilgisayarlar kullanılmaktadır. Uç bilgisayarlar benzer donanım özelliklerine sahiptir. Bütün uç bilgisayarlarda aynı yazılımlar yüklenmelidir.

Küme bilgisayarı oluşturan uç bilgisayarlar yerel ağda yüksek performanslı gevşek bir şekilde bağlıdır. Bu yüzden bir bilgisayardan diğerine erişmek aynı yerel ağda ağ anahtarı üzerinden yapılmaktadır. İletişim için Grid sistemlere göre daha az süre harcanmaktadır. Küme bilgisayarı oluşturan bilgisayarlar tek bir ağda bulunmaktadır.

Küme bilgisayarda *head-node*'a verilen işler farklı çizelgeleme yöntemleri kullanılarak dağıtılabilmektedir. Bunun için LSF (*Load Sharing Facility*), *Torque* gibi programlar kullanılmaktadır.

2.4 Merkezi Yönetimli Mimari

Alchemi heterojen masa üstü bilgisayarların Internet temelli kümelenmesi ile yapılmaktadır. Bu yapıda uç bilgisayarlar tahsisli veya gönüllü (tahsisli olmadan) çalışabilmektedir. Nesne yönelimli Grid uygulama modeli ile ince tane soyutlama (*fine-grain abstraction*) desteklenmektedir. Dosya tabanlı Grid iş modeli ile büyük taneli soyutlama (*coarse-grain abstraction*) desteklenmektedir. Alchemi Web servis ara yüzlü iş modelini de desteklemektedir.

Alchemi usta-işçi (*master-worker*) paralel programlama değerler dizisini takip etmektedir. Merkez bileşen bağımsız paralel iş bileşenlerini işçi düğümlere yönlendirir ve onları yönetir. Bu paralel çalıştırma birimi *Grid thread* olarak adlandırılacaktır. *Grid thread* çalıştırılacak komutlardan oluşur. Merkezi bileşen yönetici olarak adlandırılır.

Grid uygulama birçok *Grid thread*'lerden oluşur. Grid uygulama ve *thread*'ler Alchemi .NET API ile .NET sınıf / nesnelerden oluşur. Bu API kullanılarak yazılan bir uygulama çalıştırıldığında *Grid thread* nesnesi Grid yöneticisine çalıştırması için verilir. XML kullanılarak dosya tabanlı işler oluşturulur. İşler Alchemi *console* ara yüzü veya dış erişim (*Cross-platform*) yöneticisi web servis ara yüzü ile verilir. Bütün işler yöneticiye verilmeden önce *Grid thread*'lere dönüştürülür.

Grid oluşturan dört tip düğüm vardır. Alchemi Grid'i bir veya daha fazla çalıştırıcı düğümün yönetici düğüme bağlanmasından oluşur. İsteğe bağlı bileşen dış erişim yöneticisi diğer Grid sistemlerine bağlantıyı gerçekleştirir. Bu düğümler sırasıyla açıklanacaktır.

Yönetici: Yönetici (*manager*) düğümü Grid uygulama ve *thread*'lerin icrasını yönetme ile ilgili servisleri sağlar. Çalıştırıcılar yöneticiye kayıt olurlar. Kullanıcıdan alınan *thread* iş havuzuna yerleştirilir ve uygun çalıştırıcılarda icra edilmek üzere çizelgelenir. Her *thread*'in önceliği oluşturulduğunda veya verildiğinde açık olarak belirtilmiştir. *Thread*'ler ilk gelen ilk servis alacak (*First Come First Serve*, FCFS) şekilde çizelgelenir. Çalıştırıcılar tamamlanan *thread*'leri bir sonraki isteklerde kullanılmak üzere yöneticiye geri döndürürler. Çizelgeleme API'si özel çizelgeleme yazılmasını da destekler.

Çalıştırıcı: Çalıştırıcı düğümü yöneticiden *thread*'leri alır ve icra eder. Bir çalıştırıcı tahsisli (yönetici tarafından kaynakları merkezden yönetilecek) veya tahsisli olmayacak (kaynaklar isteğe bağlı, ekran koruyucusu veya açıkça kullanıcı tarafından belirtilerek) şekilde ayarlanabilir. Tahsisli olmayan çalıştırmada yönetici ve çalıştırıcı arasında tek yönlü iletişim vardır. Bu durumda çalıştırıcı yöneticiden *thread*'leri çalıştırmak için istekte bulunduğu için, çalıştırıcıda bulunan kaynak gönüllülük esasına göre yönetilir. İki yönlü iletişim mümkün ve tahsisli çalıştırma istendiği zaman, çalıştırıcı yönetici ile direkt haberleşebilecek ara yüzü açar. Bu durumda çalıştırıcıdaki kaynakların merkezi yönetilmesi sonucunda *thread*'leri çalıştırması için yönetici açıkça çalıştırıcıyı görevlendirir.

Kullanıcı: Grid uygulamalar kullanıcı düğümünde çalıştırılır. API kullanıcıdan Grid uygulamasını soyutlar ve kullanıcı adına birçok servisi yapmakla sorumludur. Bu servisler uygulama ve *thread* bileşenlerini çalıştırmak için gönderme, tamamlanan *thread*'in kullanıcısını bilgilendirme, sonuçları sağlama ve hata detayı ile hatalı *thread*'in kullanıcısını bilgilendirmedir.

Dış Erişim Yönetici: Dış erişim (*cross-platform*) yöneticisi web servis ara yüzüdür. Grid işlerin çalıştırılmasını yönetmek için Alchemi'yi aktif yapacak yönetici fonksiyonunun bir kısmına sahiptir. Cross-platform yöneticisine verilen işler yönetici tarafından kabul edilebilecek şekle (*thread*) dönüştürülür. Daha sonra çizelgeleme ve çalıştırma işlemi yukarıda açıklandığı gibi yapılır.

Alchemi de bütün işler yöneticiye (*manage*r) verilir. Yönetici birimi verilen işleri çalıştıracak bilgisayarlara dağıtır. İşler tek bir noktadan merkezi olarak çalıştırıcı bilgisayarlara verilir. Alchemi yönetici (*manager*), çalıştırıcı (*executor*), kullanıcı (*user*) ve dış erişim (*cross-platform manager*) birimlerinden oluşmaktadır. Bu birimler arasındaki iletişim .NET remoting kullanılarak gerçeklenmektedir. Dış erişim birimi kullanılarak diğer platformlarla (Grid'lerle) yapılan iletişimde web servisleri kullanılmıştır. Alchemi dış erişim birimi (*Cross platform manager*) ile diğer Grid sistemlerle bütünleşmeyi sağlar. Bu sayede başka Grid sistemleri ile iş alışverişi yapılır.

Alchemi de çalıştırıcı (uç) bilgisayarlar yönetici tarafından merkezi olarak yönetiliyorsa tahsisli olarak işleri çalıştırmaktadırlar. Buna ek olarak çalıştırıcı bilgisayarlar isteğe bağlı olarak da çalışabilmektedirler. Uç bilgisayarlar ekran koruyucuya bağlı olarak veya belirli zamanlarda tahsisli olmayacak şekilde bu sistemin bir parçası olabilmektedirler. Tahsisli olmayan durumda uç bilgisayarlar yöneticiden iş talebinde bulunurlar ve aldıkları işleri çalıştırırlar. Yönetici kullanıcılar tarafından kendisine verilen işleri uç bilgisayarlara push yöntemi ile vermektedir. Kullanıcılar yöneticiden iş beklerler. İşin nerede çalıştığını, tamamlanıp tamamlanmadığını yönetici birimi takip etmektedir.

Alchemi Microsoft .NET platformları için geliştirilmiştir. Uç bilgisayarlar Microsoft 2000 ve sonrası işletim sistemlerini desteklemektedir. Alchemi sistemi bir çeşit Windows küme yapıdır. Uç bilgisayarlar birbirleri ile yüksek performanslı gevşek bağlı olup .NET platformu içinde haberleşmektedir.

Alchemi sistemi birden fazla ağda çalışmayı desteklememektedir. Diğer Grid sistemleri ile bütünleşme dış erişim arabirimi (*cross-platform*) ile sağlanmaktadır.

İşler yönetici birimden öncelik ve ilk gelen ilk çalıştırılır (FCFS) çizelgeleme yöntemleri kullanılarak çalıştırılmaktadır.

2.5 Microsoft Küme Bilgisayar Yaklaşımı

Grid'e ait tüm uç bilgisayarlarda (işlemcilerde) Digipede ajanları çalışmaktadır. Ajanlar hangi işi çalıştırabileceklerine kendileri karar verirler. Uç bilgisayarlar p*ull* modeli ile işleri sunucudan almaktadır. Uç bilgisayarlar kendi özelliklerini sunucuya göndererek uygun olan işleri talep ederler. Digipede sisteminde bütün kullanıcılar

yapılacak işleri sunucu bilgisayara vermektedirler. İşler tek merkezden işleri çalıştıran işlemcilere gönderilir.

Digipede sistemlerine verilen problem büyüklüğü *coarse grain*'dir. İş seviyesinde paralellik desteği verilmektedir. Verilen işlerde kullanılan program farklı giriş parametreleri ile farklı işlemcilerde çalıştırılmaktadır. Digipede altında küme yapı varsa, gelen iş küme yapıda çalışabilecekse iş küme yapıya atanarak çalıştırılır. Bu sayede *fine grain* paralellik desteği de sağlanmaktadır.

Uç bilgisayarda hata oluşunca, sunucu hatalı bilgisayara atanan işleri yeniden başka bir bilgisayara gönderir. Sunucu bütün işleri takip eder ve tamamlanmasını garanti eder. Tüm ajanların durumları takip edilerek sistemdeki işlerin çalışması garantilenir.

Digipede sadece Microsoft işletim sistemi ve kullanıcılarını desteklemektedir. Microsoft Küme hesaplama ve bireysel Microsoft bilgisayarlar uç bilgisayar olarak kullanılmaktadır. Microsoft Visual Studio .NET 2003, Visual Studio 2005 platformları, Visual Basic, C#, C++, Java, Fortran uygulamaları desteklenmektedir. Digipede heterojen (farklı donanım ve yazılımlardan oluşan) bilgisayarları desteklemektedir. Digipede sunucu MS Windows sunucu 2003'te çalışmaktadır. Digipede ajanları MS Windows 2000'den sonraki 32 ve 64 bitlik Windows işletim sistemlerinde çalışmaktadır.

Digipede bulunan tüm uç bilgisayarlar birbirlerine düşük performanslı gevşek bir şekilde bağlıdır ve birden fazla alt ağda olabilirler.

2.6 JISGA

JISGA (*Jini-based Service-oriented Grid Architecture*) Jini (Marinescu 2002) sisteminde OGSA uyumlu Grid uygulamalar için servis tabanlı uygulamadır. Jini tüm cihazların ve servislerin birbirleriyle etkileşimini sağlayan Java sınıflarıdır. Servis tabanlı uygulamaları tanımlamak için çek bırak çalışan görsel bir araçtır olan VSCE (*Viusal Service Composition Environment*) kullanılmaktadır. Sisteme verilecek işler için SWFL (*Service Workflow Description Language*) ile uygulama dosyası oluşturulmaktadır.

WorkFlow Engine servisi SWFL ile tanımlı işler için çalıştırma ortamı

sağlamaktadır. *WorkFlow Engine* servisi *blocking* ve *non-blocking*, seri, paralel iş verilmesini desteklemektedir.

JavaSpaces Jini tarafından sağlanan temel servistir. *JavaSpaces* JISGA'da iş kuyruklarını tutmaktadır ve dağıtık süreçler (*process*'ler) arasındaki iletişimde *shared memory* metodunu kullanmaktadır.

JSHousekeeper servisi *JavaSpace* kullanan bazı işlerde kullanılmaktadır. Bu işlerin son zamanlarda kullanılmayan nesnelerini çıkarmada ve bazı nesnelerin durumlarını güncellemede *JSHousekeeper* kullanılmaktadır.

ServiceManager servisi servisleri ilan etme ve çıkarmada kullanılmaktadır. *ServiceBrowser* JISGA'daki servisleri görsel olarak takip etmede kullanılmaktadır.

JobProcessor servisi işleri çalıştıran servistir. Paralel işleri çalıştırabilmek için sistem yöneticisinin JISGA'da birden fazla *JobProcessor* tanımlaması gerekmektedir.

JISGA sistemi işler ve alt işler için kullanılan iki kuyruğu tutmaktadır. Bu kuyruklardaki işler ilk gelen ilk servis alacak (FCFS) şekilde çalıştırılır.

Özetle JISGA sistemi ağ ile birbirine bağlı dağıtık birçok işlemcisi olan sanal bir bilgisayardır. *JavaSpaces* hafıza olarak çalışır ve *JSHousekeeper* servisi hafıza yönetim birimidir. JISGA'da *JobProcessor* işlemci olarak çalışmaktadır. *WorkFlow Engine* servisi kullanıcıların Grid sisteme iş vermesi ve sonuçları almada bir ara yüz olarak kullanılmaktadır.

2.7 Yüksek Kullanılabilirlik Servis Yaklaşımı

Yüksek kullanılabilirlik (*High Available*) sistemler yazılım ve donanım bileşenlerinin birleşimi ile servis vermektedir. Bu servisin (yazılım ve donanım bileşenlerinin birlikte ortak kullanımı) kullanılabilirliğini her bileşenin ayrı kullanılabilirliğinden çok daha büyüktür. istemdeki en önemli bileşen işlemci (CPU) düğümleridir. Her işlemci düğümü Solaris işletim sistemi çalıştırmaktadır. Bireysel düğümler diğerlerini etkilemeden yeniden başlatılabilmektedir.

Donanım bileşenleri çoklu bağımsız bölmeler ve veri yolları boyunca dağıtılmıştır. Bu yüzden sadece fan, güç kaynağı, CPU, *Ethernet* kartı hataları sistemin tamamını etkilememektedir. Bütün bireysel düğümler "*Highly Available Service Cluster*" formunda birbirlerine bağlanmıştır. Düğümler yüksek kullanılabilirlik

servisine destek verecek şekilde işbirliği yapmaktadır. Düğümlerden bir tanesi çalışmıyorsa veya başka bir serviste ise, diğer düğüm bu işi kabul eder ve küme bilgisayar servislerini sağlamaya devam etmektedir.

İşletim sistemi (OS) çekirdeği ve orta katman (*middle-ware*) aracı sürekli sistem servislerinin hatalı olup olmadığını kontrol etmektedir. Sağlık gözetleyici çerçevesi hem sistemin hem de uygulama servislerinin hatalarını bulmaya yardım etmektedir. *Watchdog Timers* ve *Cluster Membership* küme bilgisayarda her düğüm işleminin tam gözetimini yapmaktadır. Her bireysel bileşende dahili hata bulma ve analizi hatanın sebebini izole temek için en hızlı ve güvenilir yoldur. Kullanılabilirlik yönetim çerçevesi uygun bileşenleri her an aktif bileşenler için kullanılacak şekilde atamaktadır

Yüksek kullanılabilirlik sistemler hataya daha duyarlı olabilmesi için önerilmiş olup diğer sistemlere alternatif değildir. Sistemlerin herhangi bir hatada nasıl davranacaklarını, nasıl çözüm sunacaklarını tanımlamaktadır.

2.8 İdeal Dağıtık Sistemin Özellikleri

Dağıtık sistemlerde işlerin uç işlemcilere atanması dağıtık olmalıdır. Dağıtık sisteme kullanıcılar tarafından verilen işler tek bir noktadan (merkezden, sunucudan) değil de birden çok noktadan verilebilmelidir. Bütün kullanıcılar sisteme iş verebilmelidir. Kullanıcıların iş verebilmek için kayıtlı olmasına gerek olmamalıdır. Yapılması gereken işler dağıtık yapıya uygun *pull* yöntemi ile alınmalıdır. Sistem içindeki uç bilgisayarlar isteğe bağlı olarak çalışabilmelidir. Uç bilgisayarlar istedikleri zaman dağıtık sisteme dahil olup verilen işleri yapabilmeli, istedikleri zaman sistemden ayrılabilmelidir. Farklı iş çizelgeleme yöntemleri desteklenmelidir. Uç bilgisayarların dahil olması veya ayrılması için kayıt mekanizmasına gerek olmamalıdır.

Dağıtık sistemlerin donanım, işletim sistemi, platform gibi gerçekleme kısıtlarının olmaması istenmektedir. Bunları destekleyebilmek için sistemin heterojen yapıları desteklemesi beklenmektedir. Web servisleri bütün platformlarda çalışabildiği için uygun çözüm sunabilmektedir. Dağıtık sistemlerin diğer sistemlerle bütünleşmesi olmalıdır. Dağıtık sistemler birden fazla ağda çalışabilmelidir. Dağıtık sistemlerde kullanıcı doğrulama ve güvenlik mekanizmaları desteklenmelidir.

Dağıtık sistemler hataya duyarlı (*fault tolerant*) olmalıdır. Sisteme verilen işler

çalışan diğer işlemciler tarafından başarı ile tamamlanmalıdır. Sisteme verilen işleri yapabilecek en az bir işlemci olduğu sürece tüm işler başarı ile tamamlanmalıdır.

Bir sonraki bölümde web Grid servis modeli detaylı anlatılacaktır.

BÖLÜM 3

HETEROJEN KÜME BİLGİSAYARLARIN WEB SERVİSLERLE MODELLENMESİ

Bu bölümde, tanımlı bir iş grubunun WEB servisleri (WSDL, Christensen 2001, Brittenham 2001) kullanarak yerel ağa bağlı heterojen bilgisayarlara (PC, sunucu, Workstation, vb.) dağıtılarak çalıştırılması için bir model geliştirilmiştir. Tanımlanan modelde işler, daha önceki geliştirilmiş merkezi yönetim ve dağıtım modellerinden farklı olarak, herhangi bir bilgisayar üzerinden sisteme verilebilmektedir. Böylece Grid kullanıcılarının kolaylıkla sisteme iş vermesi mümkün olabilmektedir. Yine önceki sistemlerden farklı olarak sisteme yeni bilgisayarların eklenmesi veya çıkarılması gibi süreçlerin kolaylaşması için işlerin dağıtımı, çalıştırılması ve takip edilmesi dağıtık karar mekanizması ile gerçekleştirilmiştir. Böylece sisteme ek bilgisayarların girmesi, çıkması ve bunların bireysel olarak hem de paralel bir sistemin parçası olarak kullanılması mümkün olmaktadır. Bu tip sistemlerde en önemli kısıtlardan birisi de sistemin bir parçası olan ve kullanılmakta olan bir yada daha fazla bilgisayarda hata oluşması durumudur (Limaye 2005). Bu durumda sisteme verilen işin hala tamamlanabilmesi önemlidir. Tanımlanan modelde yerel ağa bağlı bilgisayarlara verilen tüm işlerin, o işi yapabilecek en az bir bilgisayar olması halinde tamamlanacağı garanti edilmektedir. Tüm bunların ötesinde böyle bir sistemin yaygın kullanımı için tasarlanan model günümüzde neredeyse tüm bilgisayarlarda desteklenen bir ara yüz olan web servis mimarisi ile çalışmaktadır. Bu çözüm mimarisi ile işletim sistemi kısıtı kaldırıldığı gibi heterojen bilgisayarlar arasında da bilgi değişimi mümkün olabilmektedir. Yerel ağa bağlı heterojen bilgisayarlar üzerinde çalışan, onları paralel sistemin parçası olarak kullanıp iş dağıtımı yapan hataya dayanıklı web servisler kullanılarak modellenen çalışma web servis tabanlı Grid sistemi, WGS, olarak adlandırılacaktır.

Bundan sonraki bölümünde web servisler ile küme bilgisayarın modellenmesi, daha sonraki bölümünde ise WGS'nin web servisler ile gerçekleştirilmesi, son bölümünde de sonuç ve öneriler verilecektir.

3.1 Küme Bilgisayar Modeli (WGS)

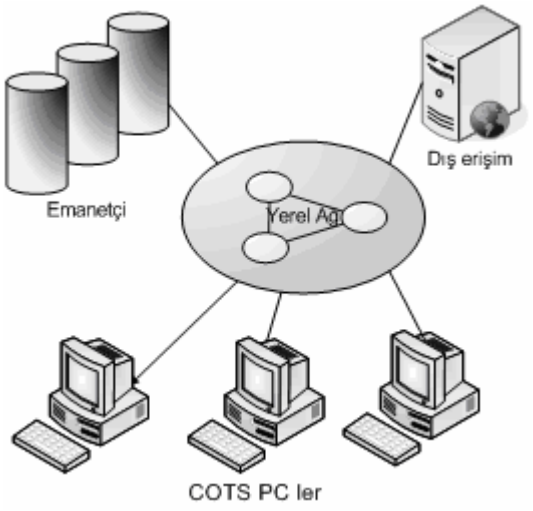
Tasarlanan model üç ana bileşenden oluşmaktadır. Bu bileşenler işlemciler, işler ve Emanetçi'dir.

• İşlemciler: Modeldeki tüm işlemcilerin yerel ağa bağlı olması ve üzerinde tanımlanan modeli gerçekleyen web servislerin çalışması gerekmektedir.

• İşler: Sisteme Emanetçi yada herhangi bir işlemci tarafından verilebilen işlerdir.

• Emanetçi: Çalışılan işlerin bir kopyasını tutan veri tabanını ve üst Grid yada dış sistemlerle iş alışverişi yapmakta kullanılabilecek iş tanımlama modülünü taşımaktadır.

Şekil 3.1'de görüldüğü gibi COTS PC'ler istemci bilgisayarları oluşturmaktadır. Bu bilgisayarlar normal kullanımda olup atıl zamanlarında sistemin bir parçası olarak kullanılmaktadır. İşler sisteme bağlı herhangi bir bilgisayar tarafından verilebileceği gibi dış erişim tanımı üzerinden doğrudan Emanetçi'ye kaydedilerek de verilebilmektedir.



Şekil 3.1 WGS'in mimari yapısı

İşlemciler üzerinde çalışan web servisler, işlemcilerin kullandığı işletim sistemine uyumludur. Web servislerin modelde verilen tanımlarla geliştirilmesi halinde, model işletim sisteminden bağımsız olmaktadır. Ayrıca modelde kullanılan işlemci

bilgisayarların aynı tip olması gerekmemektedir.

Bu modelde işler, alt bilgisayarlarda çalıştırılabilir kodları içermektedir. Çalıştırılabilir kodların bilgisayarlarda önceden var olması halinde işler, gerekli giriş ve çıkış veri dosyalarını ve çalıştırılabilir kodun tanımını içermektedir. İşler sisteme herhangi bir işlemci tarafından verilebilmektedir. Sistemin başka yerel ağlarda çalışan alt sistemlerle ya da dış Grid sistemleri ile iş alışverişini sağlamak üzere dış erişim modülü geliştirilmiştir. Dış erişim modülü alt işlemcilerde kullanılmakta olan web servisleri kullanmaktadır. Bu anlamda sistemdeki işlemcilerin herhangi birisi dış erişim ünitesinin rolünü üstlenebilmektedir.

Dağıtık sistemde herhangi bir işlemci ya da ağ bağlantısında hata oluşması durumunda çalıştırılan tüm işlerin bir kopyası Emanetçi veri depolama biriminde tutulmaktadır. Bu hali ile Emanetçi yönetici modülü gibi görülebilir. Ancak bu birimde merkezi bir karar verme, sistemde çalışan tüm işlemcilerin her an durumunu izleme ve iş dağıtma mekanizması yoktur. Emanetçi'nin tüm fonksiyonu bir işin kopyası verildiğinde bunu kaydetmek ve iş talep edildiğinde boş işlerden birini talep edene vermektir. Emanetçi sadece bir veritabanı taşıyan, veritabanı sunucuları ile kolaylıkla güvenilir olarak gerçeklenebilecek bir birimdir. Böylece donanım / yazılım temelli bir sorunla karşılaşılması olasılığı yönetici fonksiyonlarını barındıran sistemlere (yoğun çalışan ve pek çok fonksiyonu eş zamanlı çalıştıran sistemler) oranla çok daha düşük olacaktır.

Bu modelde tüm işlerin tamamlama durumlarının takibi için tutulan zamanlar relatif olarak kullanıldığı için global bir zaman senkronizasyon mekanizmasına da gerek yoktur.

Modeli oluşturan alt birimlerin açıklamaları izleyen bölümlerde detaylandırılacaktır.

3.1.1 İşlemciler

İşlemciler farklı hız ve özelliklere sahip olabilirler. İşlemcilerin sürekli çalışma ve arıza yapmama garantileri yoktur. Herhangi bir anda bir işlemci arızalı ya da kapalı olabilir. İşlemcilerin bir kısmı bazı işlemleri yapmak üzere özelleşmiş olabilirler. İşlemciler tek başlarına çalışabildiği gibi işlemci grubu olarak da çalışabilirler.

Dışarıdan bir işlemciye iş verildiğinde bu işin bir kopyası ve işin durum bilgileri Emanetçi'ye gönderilir.

İşlemci herhangi bir anda kendisine verilen bir iş üzerinde çalışmaya devam eder veya işlemci hatası oluşur. İşlemci hatası bilgisayardaki herhangi bir birimin veya bilgisayar ile küme sistem arasındaki iletişimin gerektiği gibi çalışmamasından kaynaklanmaktadır. Nedeni ne olursa olsun hatalı işlemci kendisinin hatalı olduğunu küme bilgisayar sistemine bildiremez. Tasarlanan model ile herhangi bir işlemcide hata oluşunca, Emanetçi tarafından hatalı işlemciye atanan işler talep edilmesi halinde başka bir işlemciye verilmektedir. Böylece hatalı işlemciye atanmış bütün işler diğer işlemciler tarafından tamamlanacaktır.

Her işlemcinin anlık iş yükünü *AnlıkYük* değeri göstermektedir. Bu değer Çalıştırıcı uygulaması ile sürekli güncellenmektedir. İşlemci, iş yükü *EnAzYük* eşik değerinden az ise Emanetçi'den yeni iş talep etmektedir.

3.1.2 İşler

Küme bilgisayar sistemine işler herhangi bir anda, herhangi bir bilgisayar üzerinden verilebilir. Her iş bir işlemci veya paralel çalışan işlemci grubunda yapılacak biçimde tanımlanabildiği gibi, o işe özel bir işlemci veya işlemci grubu tarafından yapılabilecek biçimde de tanımlanabilir. Verilen iş için özel bir işlemci ya da işlemci grubu tanımlanmamışsa herhangi bir işlemcide yapılabileceği varsayılmaktadır. İşin ne kadar sürede yapılacağı önceden bilinmektedir. Yeni bir iş verildiğinde işle ilgili bilgiler ve işin bir kopyası Emanetçi'ye gönderilmektedir.

3.1.3 Emanetçi

Sürekli çalışma ve hata yapmama garantisi olan bir veri depolama birimidir. Kendisine dışarıdan bir iş verilebilir. Emanetçi kendisine verilen işleri yapmaz. Bu işleri talep edilmesi halinde sistemdeki diğer işlemcilere gönderir. Küme bilgisayar sistemine gelen tüm işlerin kayıtları (işin tanımı, çalıştırma süresi, tamamlanıp tamamlanmadığı yada herhangi bir işlemcide çalıştırıldığı, vb. bilgileri) Emanetçi'de tutulmaktadır. Böylece herhangi bir nedenle işlemci hatası oluşursa, hatalı işlemcideki

tamamlanmayan işlerin ortadan kaybolması önlenmiş olacaktır. Emanetçi'deki yeni işler ve hatalı işlemciye verilen işler talep eden diğer işlemcilere verilir.

İş durum tablosuna göre dört çeşit iş durumu vardır:

Yeni İş: Emanetçi'ye verilen işler. Bunlar yeni iş olarak sistemdeki diğer işlemcilere istendiğinde verilir. Yeni işler geliş sırasına göre verilir. İlk gelen iş ilk son gelen iş en son verilir. Bu yolla işlerin ortalama bekleme süreleri azaltılmış olur. Bu aşamada bir öncelik tanımlaması da kullanılabilir.

Atanmış İş: Emanetçi dışındaki bir işlemciye direk atanan işler.

Bitmiş İş: İşlemciden "iş tamamlandı" bilgisi gelmiş işler.

Bitmemiş İş: Bir işlemciye atanan ve işin bitme süresi dolduğu halde "iş tamamlandı" bilgisi alınmayan işler. Bu durumda işlemci hatası olduğu kabul edilir. Bitmemiş işler listesi bekleme süresine göre azalan yönde sıralanır. Bu gruptan bir iş istendiğinde listenin üstünden itibaren işler verilir.

3.1.4 Modelin Çalışması

Küme bilgisayarı oluşturan işlemciler P1, P2, …, Pn yerel ağ ile birbirleriyle ve Emanetçi ile haberleşebilmektedir (Şekil 3.1). Dışarıdan gelen işler herhangi bir işlemciye verilebilir. İşi alan işlemci, Pi işin bir kopyasını Emanetçi'ye gönderir. İşi alan işlemci aldığı işi ne zaman tamamlayacağını da gönderir. Emanetçi işi sisteme Pi'ye atanmış olarak bitiş süresiyle birlikte kaydeder. Verilen iş Pi tarafından yapılır. Bu şekilde sisteme gelen işler verilen işlemcilerde çalıştırılmaktadır.

Herhangi bir anda Pi işlemcisinin iş yükü (*AnlıkYük*) en az yük seviyesinden *(EnAzYük*) az ise Emanetçi'den yeni bir iş talep edilmektedir. Emanetçi varsa sıradaki yeni işi verir. Eğer yeni iş yoksa iş durum tablosundaki bitmemiş işlerden sıradakini verir. İşi alan Pi işlemcisi işi bitireceği göreli zaman bilgisini Emanetçi'ye bildirir. Emanetçi işi sisteme Pi işlemcisine atanmış olarak bitiş süresiyle birlikte kaydeder. Pi işlemcisi kendisine verilen işleri çalıştırır.

Herhangi bir işlemcide hata oluştuğunda, bu işlemciye verilen işler zamanında tamamlanamaz. Emanetçi'de bulunan atanmış işlerden zamanında tamamlanmayan işler bitmemiş işler listesine eklenir. Üzerindeki iş yükü azalan işlemciler Emanetçi'den bu işleri alacaklar ve çalıştıracaklardır. Tasarlanan bu model sayesinde hatalı işlemcilere

verilen işler de diğer işlemciler tarafından tamamlanabilecektir. Sistemde çalışan en az bir işlemci olduğu ve yeteri kadar zaman verildiği sürece verilen tüm işler tamamlanacaktır.

3.2 WGS'in Gerçeklenmesi

Modelin gerçeklenmesi işlemciler üzerinde çalışacak servis ve uygulamalar yardımıyla sağlanmaktadır. Bu servis ve uygulamalara ek olarak, her işlemci çalıştıracağı ya da çalıştırmakta olduğu işlerin bir listesini de tutmaktadır. Her işlemcide kullanılan servisler İş_Ver, İş_İstek, Liste_Güncelleme'dir. Bu servislere ek olarak bu servisleri kullanan iki uygulama da İş_Ara ve Çalıştırıcı bulunmaktadır. Kullanılan servis ve uygulamalar sırasıyla aşağıda açıklanacaktır.

İş_Ver Servisi: Herhangi bir işlemciye dışarıdan iş vermek için kullanılan servistir. İşlemci işi aldığında, *AnlıkYük* değeri *EnÇokYük* eşik değerini geçmiyorsa, yapılacaklar listesine ekler ve çalıştırmak üzere sıraya koyar. Ayrıca işin bir kopyasını Emanetçi'deki İş_Teslim servisi ile Emanetçi'ye teslim eder. Bu durumda işin sahibi işi teslim eden işlemcidir. *AnlıkYük* değeri *EnÇokYük* değerini geçiyorsa ya da işlemcinin yapamayacağı türden özel bir iş geldiyse Emanetçi'ye sahipsiz iş olarak gönderilmektedir. İş doğrudan Emanetçi'ye verilmiş ise bu işin sahibi yoktur ve ilk talep edene verilmektedir.

İş_İstek Servisi: Bir işlemcinin *AnlıkYük* değeri *EnAzYük* değerinin altına düştüğünde bu servisi kullanarak Emanetçi'den iş talebi yapar. Emanetçi öncelikle yeni işlerden sıradakini, yoksa bitmemiş işlerden sıradakini verir. İşi alan işlemci aldığı işin tamamlanma süresini Emanetçi'ye relatif süre olarak bildirmektedir. Bu servis Emanetçi'deki işler listesine erişimi ve bir işlemciye atanmasını atomik bir işlem olarak yapar. Yani birden fazla işlemcinin İş_İstek talebi sıraya konularak cevaplanmaktadır.

Liste_Güncelleme Servisi: Emanetçi'deki işlerin durumlarını güncelleyen bir servistir. İş_İstek servisi ile verilen işin bitiş süresi, işin verildiği işlemci tarafından güncellenir. Bir işlemci bir işi başarı ile tamamladığında Emanetçi'deki kaydı bu servis aracılığı ile

günceller.

İş_Ara Uygulaması: Bu uygulama herhangi bir işlemcideki iş yükünü (*AnlıkYük*) kontrol eder ve en az yük seviyesinden (*EnAzYük*) küçük olduğunda İş_İstek servisini çalıştırır.

Çalıştırıcı Uygulaması: İşlemci içinde kayıtlı işlerin çalıştırılmasını sağlayan uygulamadır. Bu uygulama kendi kayıt listesindeki işleri sırasıyla alır ve işlemci üzerinde çalıştırır. İş tamamlandığında İşlemci kendi iş durum listesini ve Emanetçi'deki Kayıt listesini günceller. İstenirse dış dünyaya "iş tamamlandı" bilgisini de gönderir. Her işin bitiminde çalıştırıcı işlemci iş yükünü, *AnlıkYük*, ve Liste_Güncelleme servisini kullanarak Emanetçi'deki kayıtları günceller.

Küme bilgisayar sistemine işler herhangi bir işlemci ya da Emanetçi üzerinden verilebilir. Bir işlemciye verilen işin bir kopyası Emanetçi'ye İş_Teslim servisi ile gönderilir. İşi az olan işlemci İş_İstek servisi ile Emanetçi'den iş talebinde bulunur. Emanetçi elindeki işleri (hiç bir işlemciye atanmamış ya da bir işlemciye atandığı halde bitiş süresi geçmiş işler) gönderir.

3.3 WGS'in Çalışması

Sisteme işler herhangi bir işlemci ya da Emanetçi üzerinden verilebilir. İşin verildiği işlemci, genellikle işin sahibidir. Ancak verilen iş işlemcinin en çok yük kapasitesini aşması halinde Emanetçi'ye sahibi olmayan iş olarak da teslim edilmektedir. Böylece iş yükünün sürekli verilen işlemci yerine boştaki diğer işlemcilere dağıtılması da mümkün olmaktadır. Her işlemci kendisine verilen işin bir kaydını Emanetçi'ye teslim eder. Böylece herhangi bir nedenle işlemcide hata oluşması durumunda verilen işin tamamlanma süresi dolduğunda başka bir işlemci tarafından Emanetçi'deki kopya alınarak işin tamamlanması mümkün olacaktır. Üç web servisi ve iki uygulamanın önceden yüklendiği (ya da kullanıcı izniyle durum başına yüklendiği) anda işlemci sistemin bir parçası olarak çalışmaya başlar. Merkezi bir yönetim ve kaynak durum kaydı olmadığı için işlemcilerin devreye girip çıkması sistemin

çalışmasını etkilemez. Bu ölçeklenebilir yapı ve tanımlı servislerin işletim sisteminin bir parçası olması halinde, kabul eden tüm kullanıcıların sistemin bir parçası olması kolaylıkla mümkün olabilecektir. Burada dikkat edilecek önemli bir konu merkezi yönetim biriminin kaldırılmış olması ve bu halde bile hata güvenliğinin sağlanmış olmasıdır.

Ayrıca sistem kullanıcılarının güvenliği için her servis belli kimlik sorgulaması ve güvenli iletişim aşaması ile kullanılabilmektedir. Sistemin başka sistemler ile birlikte kullanılabilmesi yada başka bir Grid yapısının alt öğesi olarak çalışabilmesi için Dış Erişim birimi kullanılmaktadır. Dışarıdan bir iş vermek ya da işin durumunu takip etmek; işin Emanetçi'ye teslim edilmesi ve durum listesinin sorgulanmasından ibaret olacaktır. Dışarıdan verilmiş olan bir iş yine İş_Ver web servisi ile sisteme aktarılabilir. Sonuçlanan iş Emanetçi'deki durumu sorgulayan / raporlayan bir servis tarafından dış dünyaya iletilebilir.

Sistem yerel ağa bağlı 7 PC bilgisayar (7 Pentium IV, 1 HP 24 *port Switch* 10/100) ile test edildi (Adar 2006, 2007, Akçay 2007) . Uç bilgisayarlarda MS Windows işletim Sistemi .NET framework ile kuruldu. Her bir bilgisayara servis ve uygulamalar yüklendi. Bilgisayarlardan bir tanesi Emanetçi olarak kullanıldı ve bunun üzerine web servisler ile erişilebilen MS SQL veri tabanı kuruldu. İstemci bilgisayarlarda hata durumu ağ bağlantısını çıkararak veya istemci bilgisayar(lar)ın gücünü tamamen keserek yapıldı. Yapılan testlerde aşağıdaki durumlar test edildi. Bunlar;

- Tüm birimlerden aynı anda işler sisteme yüklenildi ve sonuç olarak tüm işlerin tamamlandığı gözlendi.
- Tüm birimlerden işler yüklendikten sonra sadece 1 istemci bilgisayarda hata yaratıldı. Hata oluşturulan istemci bilgisayarın işleriyle birlikte tüm işlemlerin tamamlandığı gözlendi.
- Tüm birimlerden işler yüklendikten sonra 1 istemci bilgisayar kalacak şekilde diğer tüm istemcilerde hata yaratıldı. Hata oluşturulan istemci bilgisayarların işleriyle birlikte tüm işlemlerin tamamlandığı gözlendi.
- Tüm işler sadece dış erişim ünitesinden verildi ve çalışmakta olan istemcilerin sayısı rasgele değiştirildi (devreden çıkarma ve devreye alma). Tüm işlerin tamamlandığı gözlendi.

Uygulama sırasında bu eşik zamanlarında yapılan değişikliklerin toplam iş bitirme zamanlarını etkilediği gözlendi. İşlerin ortalama büyüklüklerinin artması halinde tüm işlerin tamamlanma süresinde artışlar gözlendi. Bunun nedeni hata yapan işlemci(ler)deki işlerin tamamlanma süre(ler)i geçtikten sonra tekrar dağıtılabilir olmasıdır.

3.4 Sonuçlar

Bu çalışmada tanımlı bir iş grubunun web servisleri kullanarak bir yerel ağa bağlı heterojen bilgisayarlara dağıtılarak çalıştırılması için bir model geliştirildi. Model sayesinde standart bilgisayarların hem bireysel hem de paralel bir sistemin parçası olarak kullanılması mümkün olmaktadır. Ayrıca yerel ağda çalışan sisteme (gerekirse bir Grid çalışmasının alt öğesi olarak) dışarıdan iş verilmesi mümkündür. Tanımlanan modelde yerel ağa bağlı bilgisayarlara verilen tüm işlerin o işi yapabilecek en az bir bilgisayar olması halinde tamamlanacağı garanti edilmektedir. Yapılan testlerde *EnAzYük* ve *EnÇokYük* eşik değerlerinin ortalama iş büyüklüğü ile orantılı verilmesi gerektiği ve bu oranların da iş büyüklüğü ile ters orantılı olarak değişmesi gerektiği gözlendi. Bir istemci bilgisayara verilen işlerin toplam çalıştırma zamanının o istemcinin *EnÇokYük* eşik değerini aşması halinde işin Emanetçi'ye sahipsiz olarak teslim edilmesi yerine dağıtık bir mekanizma ile dağıtılması yönünde geliştirme çalışmaları sürdürülmektedir.

Bir sonraki bölümde WGS modeli kullanılarak elde edilen sonuçlar detaylı verilmektedir.

BÖLÜM 4

MODELİN UYGULANMASI

Bu bölümde WGS modelinin gerçekleme, simülatör ve teorik elde edilen sonuçları detaylı olarak anlatılacaktır.

4.1 Gerçekleme Sonuçları

Bir önceki bölümde açıklanan WGS modelinin 3 uç bilgisayar ve Emanetçi ile gerçeklemesi yapıldı. Her uç bilgisayarda 2.6 GHz P4 işlemcisi, 512 MB RAM yerel hafıza, 80 GB sabit hafıza ve Ethernet kartı bulunmaktadır. Uç bilgisayarlar gigabit ağ anahtarı ile yerel ağda haberleşecek şekilde kuruldu. Sisteme verilen işlerin isimleri t1'den t30'a kadar sıralanmaktadır. İşlerin uzunlukları 5 ile 90 birim arasında değişmektedir. Başlangıçta bilgisayarlara atanan işler "işin adı, işin tamamlanması için gereken süre" şeklinde verildi. İşler yukarıdan aşağıya doğru verildi ve bilgisayarlarda geliş sırasına göre sıralandı. İlk gelen iş en üstte, ikinci iş üstten ikinci, son gelen iş en son satırda çalıştırılacak işlemcide yer almaktadır. Uç bilgisayarlardaki işler ilk gelen ilk çalıştırılır şeklinde yürütülmektedir. Verilen işler sistemdeki herhangi bir bilgisayarda çalışabilen türden işlerdir. Bilgisayar isimleri P5, P9, P13 ve Emanetçi olarak isimlendirildi.

Bir önceki bölümde açıklanan WGS modelinin gerçeklemesi için üç tane temel test yapıldı. Bunlar WGS sistemine verilen tüm işlerin hatasız çalışma ortamında tamamlanıp tamamlanmadığı, herhangi bir işlemcide hata olduğunda hatalı işlemcideki işlerin tamamlanıp tamamlanmadığı ve sistemde işleri çalıştırabilecek bir işlemci kaldığında verilen tüm işlerin tamamlanıp tamamlanmadığıdır. Bunlar sırasıyla takip eden bölümlerde anlatılacaktır.

4.1.1 Hatasız Çalıştırma ile Verilen Tüm İşler Tamamlandı

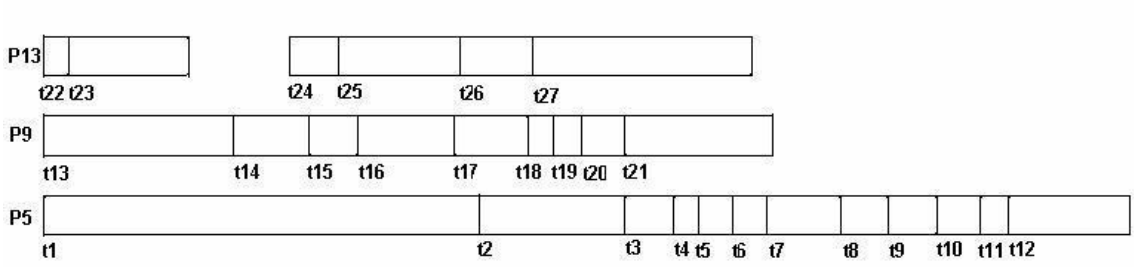
Bu testin amacı WGS modelindeki tüm işlemcilere atanan işlerin normal çalışma şartlarında (sistemde bulunan tüm birimlerin beklendiği gibi çalışması) tamamlanıp

tamamlanmadığını kontrol etmektir. Başlangıçta P5 bilgisayarına t1'den t12'ye kadar olan işler, P9 bilgisayarına t13'den t21'e kadar olan işler, P13 bilgisayarına t22, t23 işleri ve Emanetçiye t24'den t27'ye kadar olan işler Çizelge 4.1 de görüldüğü gibi atandı. Tüm bilgisayarlardaki işlerin yürütülmesine aynı anda başlandı. Verilen bütün işlerin atandıkları işlemcilerde başarı ile tamamlandıkları gözlendi. Şekil 4.1'de yatay eksen zamanı, dikey eksen işlemcilerin adlarını göstermektedir. P5 ve P9 bilgisayarlarına verilen işlerin başarı ile tamamladığı şekil 4.1'de görülmektedir. P13 bilgisayarı kendi işlerini tamamladıktan sonra iş talebinde bulunarak t24'den t27'ye kadar olan işleri de Emanetçi'den alarak başarı ile tamamladığı şekil 4.1'de görülmektedir. Emanetçi'deki işlerin P13 bilgisayarında başlaması için yerel ağdaki gecikmeden dolayı bir süre beklenilmesi gerekmektedir. Sisteme verilen bütün işlerin tamamlandığı gözlenmektedir.

Sonuç olarak WGS modelindeki tüm işlemcilere atanan işlerin normal çalışma şartlarında tamamlandığı, Emanetçi'deki işlerin boşta kalan işlemci de tamamlandığı test edildi. WGS' de uç işlemcilerden herhangi bir tanesinde hata olduğunda yapılan test sonuçları bir sonraki alt bölümde anlatılmaktadır.

Çizelge 4.1 Hatasız çalıştırma için başlangıçta bilgisayarlara verilen işler

P5	P9	P13	Emanetçi
t1, 90	t13, 40	t22, 5	t24, 10
t2, 30	t14, 15	t23, 25	t25, 25
t3, 10	t15, 10		t26, 15
t4, 5	t16, 20		t27, 45
t5, 7	t17, 15		
t6, 7	t18, 5		
t7, 15	t19, 6		
t8, 10	t20, 9		
t9, 10	t21, 30		
t10, 9			
t11, 6			
t12, 25			



Şekil 4.1 Hatasız çalıştırma Gantt şeması

4.1.2 Hatalı Bilgisayar Varsa Verilen Tüm İşler Tamamlandı

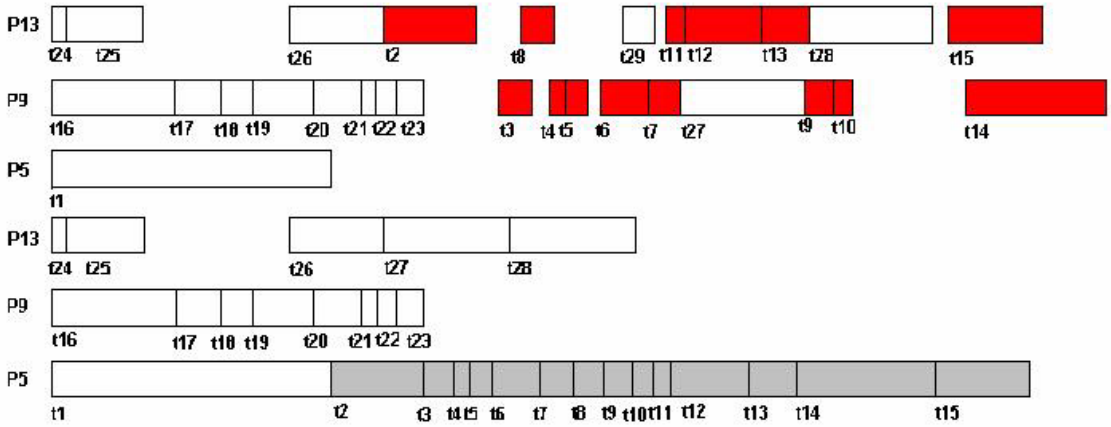
Bir önceki testte açıklanan WGS gerçeklemesi bu test için de kullanıldı. Başlangıçta bilgisayarlara atanan işler Çizelge 4.2'de gösterilmektedir. P5 bilgisayarına t1'den t15'e kadar olan işler, P9 bilgisayarına t16'dan t23'e kadar olan işler, P13 bilgisayarına t24, t25 işleri ve Emanetçi'ye de t26'dan t28'e kadar olan işler atandı. Bilgisayarlar aynı anda kendilerine atanan işleri çalıştırmaya başladı. P5 bilgisayarı t>90 için hatadan dolayı çalışmamaktadır. Şekil 4.2'de alttaki 3 satır hata olmasaydı sisteme verilen işlerin hangi bilgisayarlarda çalıştırılacaklarını göstermektedir. Şekil 4.2 de üstteki 3 satır P5 bilgisayarında t>90 için hata varsa verilen işlerin nasıl tamamlandığını göstermektedir. P5 bilgisayarında sadece t1 işi tamamlanacaktır. P5'e atanan diğer işler (t2-t15) P9 ve P13 bilgisayarları tarafından tamamlanacaktır. P9 ve P13 bilgisayarları önce kendilerine başlangıçta verilen işleri tamamlayacaktır. P9 ve P13 daha sonra Emanetçi ve P5'e verilen işlerden alarak kendilerine verilen işleri tamamlayacaktır. İşler çalıştırılırken bazı işler arasında oluşan boşluklar, işler P9 ve P13 bilgisayarlarına aktarılırken yerel ağdaki beklemelerden kaynaklanmaktadır.

Sonuç olarak WGS sistemine verilen tüm işlerin tamamlandığı, hatadan dolayı tamamlanamayan işlerin sistemdeki diğer bilgisayarlar tarafından tamamlandığı gözlenmektedir(Akçay, 2005).

Bir sonraki alt bölümde işleri çalıştırabilecek tek bir bilgisayar kaldığında sisteme verilen işlerin durumları incelenecektir.

Çizelge 4.2 Hatalı bilgisayar varsa başlangıçta bilgisayarlara verilen işler

P5	P9	P13	Emanetçi
t1, 90	t16, 40	t24, 5	t26, 30
t2, 30	t17, 15	t25, 25	t27, 40
t3, 10	t18, 10		t28, 40
t4, 5	t19, 20		
t45, 7	t20, 15		
t6, 15	t21, 5		
t7, 10	t22, 6		
t8, 10	t23, 9		
t9, 9			
t10, 6			
t11, 6			
t12, 25			
t13, 15			
t14, 45			
t15, 30			



Şekil 4.2 Hatalı bilgisayar varsa Gantt şeması

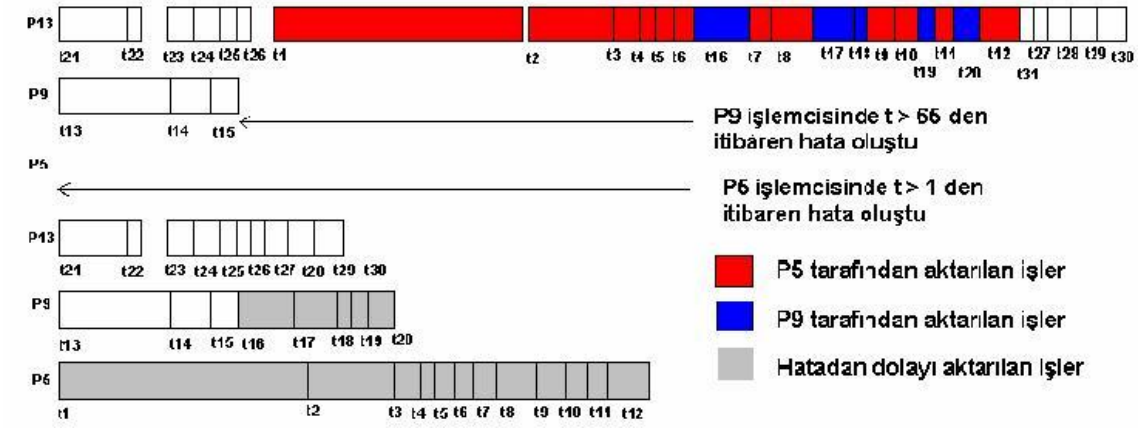
4.1.3 Çalışan En Az Bir Bilgisayar Varsa Verilen Tüm İşler Tamamlandı

Bu testte başlangıçta tüm bilgisayarlara Çizelge 4.3'te görüldüğü gibi işler atandı. P5 bilgisayarına t1'den t12'ye kadar olan işler, P9 bilgisayarına t13'den t20'ye kadar olan işler, P13 bilgisayarına t21, t22 işleri ve Emanetçi'ye t23'den t30'a kadar olan işler başlangıçta verildi. P5 bilgisayarında t>1 için hata olduğundan kendisine verilen hiçbir iş tamamlanamayacaktır. P5'e verilen işler P13 tarafından tamamlanmaktadır. P9 bilgisayarında t>65 için hata olduğundan t13, t14, t15 işleri

tamamlandı. Ancak P9 bilgisayarına atanan diğer işler (t16-t20) sistemde çalışan tek bilgisayar P13 tarafından tamamlanmaktadır. Şekil 4.3'de en altta hata olmasaydı bilgisayarlarda tamamlanan işler gösterilmektedir. Emanetçi'ye verilen işlerin P13 bilgisayarı tarafından başarı ile tamamlandığı görülmektedir. P5 ve P9 bilgisayarlarında hatadan dolayı tamamlanamayan işler şekil 4.3'de gösterilmektedir. Şekil 4.3'de üstte ise hatadan dolayı işlerin P13'de tamamlandığı görülmektedir. Bazı işler arasında oluşan boşluklar yerel ağdaki işlerin iletilmesinden kaynaklanmaktadır. Sisteme verilen bütün işleri çalıştırabilecek en az bir bilgisayar olduğu sürece, sisteme verilen tüm işlerin tamamlandığı gözlenmektedir.

Çizelge 4.3 Çalışan en az bir bilgisayar varsa tüm işler tamamlanır için başlangıçta verilen işler

P5	P9	P13	Emanetçi
t1, 90	t13, 40	t21, 25	t23, 10
t2, 30	t14, 15	t22, 5	t24, 9
t3, 10	t15, 10		t25, 6
t4, 5	t16, 20		t26, 5
t5, 7	t17, 15		t27, 5
t6, 7	t18, 5		t28, 8
t7, 8	t19, 6		t29, 10
t8, 15	t20, 9		t30, 10
t9, 10			
t10, 8			
t11, 7			
t12, 15			



Şekil 4.3 Çalışan en az bir bilgisayar varsa tüm işler tamamlanır için Gantt şeması

Sonuç olarak tasarlanan WGS modelinde 3 bilgisayar ve Emanetçi'den oluşan küçük bir model örnek işler verilerek test edildi. Herhangi bir hata yoksa verilen tüm işlerin atanan bilgisayarlarda çalıştığı gözlendi. Herhangi bir bilgisayarda hata varsa, hatadan dolayı tamamlanamayan işler diğer bilgisayarlarda tamamlanmaktadır. WGS' de verilen işleri yapabilecek en az bir bilgisayar olduğu sürece verilen tüm işlerin tamamlandığı görülmektedir.

Bu bölümdeki gerçeklemelerdeki ilk amaç küçük bir iş grubu ile WGS modelinin çalışıp çalışmadığının testidir. WGS modelinin küçük bir iş grubu için başarı ile çalıştığı görüldü. WGS modelinde hatalı bilgisayara verilen işlerin diğer çalışan bilgisayarlar tarafından tamamlandığı test edildi. Sonuç olarak WGS modelinin çalıştığı gerçekleme ile desteklendi.

4.1.4 Gerçekleme ile EnAzYük ve EnÇokYük Değerlerinin Bulunması

Bir önceki bölümde tanımlanan WGS modelinin küçük iş grubu ile gerçeklemesi yapıldı. Bu bölümde daha büyük iş grubu ve işlemci kullanılarak gerçekleme yapıldı. Emanetçi ve 6 adet bilgisayardan oluşan WGS sistemi, her bilgisayar hızlı ağ anahtarı ile haberleşecek şekilde kuruldu. Her bir uç bilgisayarda gerekli web servisleri ve uygulamalar yüklendi.

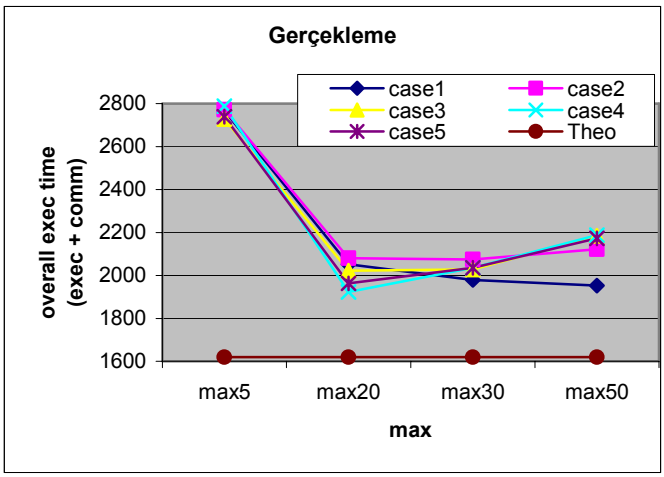
Bu alt bölümde en uygun *EnAzYük* ve *EnÇokYük* değerlerini bulabilmek için farklı başlangıç iş dağılımları ile WGS'in performansı incelenecektir. Başlangıçta uç bilgisayarlara verilen iş miktarı çizelge 4.4 de verilmektedir. Çizelge 4.4'de AVG değeri *EnAzYük* ve *EnÇokYük* değerlerinin aritmetik ortalamasıdır ((*EnAzYük*+*EnÇokYük*)/2). Çizelge 4.4'de MAX+ değeri *EnÇokYük* (max) değeri ve MIN+ değeri *EnAzYük* değeri ile aynı alınmaktadır. Çizelge 4.4'de BOS değeri için hiçbir iş yükü (0 birimlik iş) atanmamaktadır. Çizelge 4.4'de kullanılan OTHERS WGS sisteminde başlangıçta uç bilgisayarlara atanan işlerden geri kalan işleri göstermektedir. Bu işler başlangıçta Emanetçi'ye atanmaktadır. Farklı başlangıç durumları case1, case2, case3, case4, case5 ile isimlendirilmektedir. Her durumda P1, P2, P3, P4, P5, P6 ve Emanetçi'ye atanan iş miktarları hesaplanarak ilgili bilgisayarlara atanmaktadır.

Çizelge 4.4 Başlangıçta bilgisayarlara verilen iş miktarları

	Emanetçi	P1	P2	P3	P4	P5	P6
case1	OTHERS	AVG	AVG	AVG	AVG	AVG	AVG
case2	OTHERS	MAX+	MAX+	MIN+	MIN+	MIN+	MIN+
case3	OTHERS	MAX+	MAX+	MIN+	MIN+	BOS	NOS
case4	OTHERS	MAX+	MAX+	BOS	BOS	BOS	BOS
case5	OTHERS	MAX+	MAX+	MAX+	MAX+	MAX+	MAX+

Çizelge 4.5 Gerçekleme ile farklı durumlar ve EnÇokYük (max) için iş tamamlama süreleri

Test A	max5	max20	max30	max50
case1_3_max	2768	2052	1979	1953
case2_3_max	2772	2080	2075	2122
case3_3_max	2729	2022	2028	2184
case4_3_max	2788	1923	2035	2187
case5_3_max	2739	1963	2035	2173
Teo	1619	1619	1619	1619



Şekil 4.4 Gerçekleme ile farklı durumlar ve EnÇokYük (max) için iş tamamlama sürelerinin değişimi

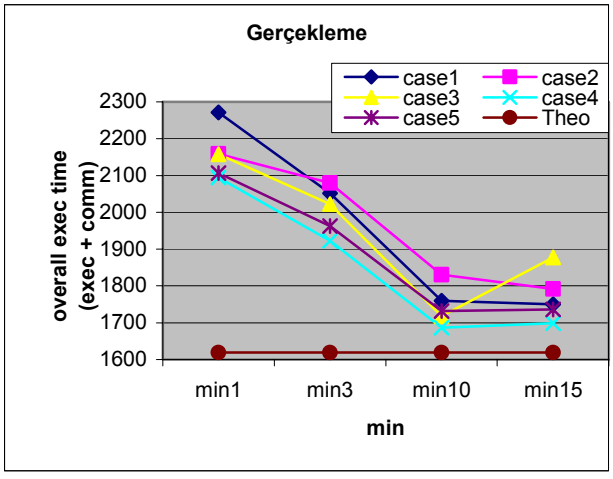
Çizelge 4.5'de ilk sütunda farklı başlangıç durumları bulunmaktadır. "Theo" ise ideal çalışma şartlarında her bilgisayarın eşit iş yükü aldığında elde edilebilecek sonuçları göstermektedir. İkinci sütunda *EnÇokYük* (max) ortalama iş uzunluğunun 5 katı için verilen işlerin tamamlanma sürelerini vermektedir. Çizelge 4.5'de her değer

karşılık gelen satır (başlangıç dağılımı) ve sütün (*EnÇokYük*) parametrelerinde verilen işlerin tamamlanması için gereken süreyi göstermektedir. Örneğin, ilk satırdaki değerler başlangıç dağılımı case1, *EnAzYük* ortalama iş uzunluğunun 3 katı için farklı *EnÇokYük* değerlerinde (3, 20, 30 ve 50 ortalama iş uzunluğu) verilen tüm işlerin tamamlanması için gereken süreleri göstermektedir.

İlk önce bütün testler için *EnAzYük* (min) değeri için sabit ortalama iş uzunluğunun 3 katı (30 iş çalıştırma süresi) alındı. *EnÇokYük* (max) değerleri de 5, 20, 30, 50 (50, 200, 300, 500 iş çalıştırma süreleri) alınarak bütün işlerin tamamlanması için gereken süreler bulundu. Şekil 4.4'de *EnÇokYük* değeri max20 (ortalama iş uzunluğunun 20 katı, 200 iş tamamlanma süresi) en iyi sonuç elde edildi (Akçay, 2007).

Çizelge 4.6 Gerçekleme ile farklı durumlar ve EnAzYük (min) için iş tamamlama süreleri

Test B	min1	min3	min10	min15
Case1_min_20	2271	2052	1760	1750
Case2_min_20	2159	2080	1831	1792
Case3_min_20	2158	2022	1718	1878
case4_min_20	2094	1923	1687	1699
case5_min_20	2107	1963	1732	1736
Theo	1619	1619	1619	1619



Şekil 4.5 Gerçekleme ile farklı durumlar ve EnAzYük (min) için iş tamamlama sürelerinin değişimi

Bir önceki test (Test A) ile bulunan *EnÇokYük* değeri (max20) kullanılarak farklı *EnAzYük* (min) değerleri için bütün işlerin tamamlanması için gereken süreler bulundu. En uygun *EnAzYük* değerini bulabilmek için başlangıçta bilgisayarlara atanacak iş miktarları çizelge 4.4'de verilmektedir. Bütün işlerin farklı başlangıç ve *EnAzYük* değerleri için tamamlanması gereken süreler şekil 4.5'te verilmektedir. Şekil 4.5'den görüleceği gibi toplam süreyi en küçük yapan *EnAzYük* değeri min10 (ortalama iş uzunluğunun 10 katı, 100 iş tamamlama süresi)'dur.

Geliştirilen WGS modeli için en uygun *EnAzYük* ve *EnÇokYük* değerleri bulundu. WGS modeli farklı iş çizelgeleme yöntemlerini desteklemektedir.Bundan sonraki iki bölümde ilk gelen ilk servis alır ve açık artırma iş çizelgeleme yöntemleri gerçeklenmiştir.

4.1.5 Gerçekleme ile İlk Gelen ilk Servis Edilir Testi

Bu bölümde WGS modelinin gerçeklemesi 6 uç bilgisayar ve Emanetçi ile yapıldı. Uç bilgisayarlara atanan işlerin atanması ve çalıştırılması sıralarının sistem performansını nasıl değiştirdiği araştırılacaktır. Bilgisayar iş yükü *AnlıkYük* değeri *EnAzYük* değerinden küçük ise Emanetçi'den iş talebinde bulunmaktadır. İlk yapılan test, gelen işlerin geldiği sırada çalıştırılmasıdır. Uç bilgisayara ilk gelen iş ilk çalıştırılacak, ikinci gelen iş ikinci, son gelen iş en son çalıştırılacaktır. Bu yaklaşım literatürde ilk gelen ilk servis edilir (*First Come First Serve*, FCFS) olarak isimlendirilmektedir.

Çizelge 4.7'de AVG değeri *EnAzYük* (max) ve *EnÇokYük* (min) değerlerinin aritmetik ortalamasıdır ((*EnAzYük* + *EnÇokYük*)/2). Çizelge 4.7'de MAX+ değeri *EnÇokYük* (max) değeri ve MIN+ değeri *EnAzYük* (min) değeri ile aynı alınmaktadır. Çizelge 4.7'de BOS değeri için hiçbir iş yükü (0 birimlik iş) atanmamaktadır. Çizelge 4.4'de kullanılan OTHERS WGS sisteminde başlangıçta uç bilgisayarlara atanan işlerden geri kalan işleri göstermektedir. Bu işler başlangıçta Emanetçi'ye atanmaktadır. Farklı başlangıç durumları case1, case2, case3, case4, case5 ile isimlendirilmektedir. Her durumda P1, P2, P3, P4, P5, P6 ve Emanetçi'ye atanan iş miktarları hesaplanarak ilgili bilgisayarlara atanmaktadır.

Farklı *EnAzYük* ve *EnÇokYük* değerleri için bütün işlerin tamamlanma süreleri

çizelge 4.8'de verilmektedir. *EnAzYük* ortalama iş uzunluğunun 10 katı *EnÇokYük* ortalama iş uzunluğunun 20 katı için en küçük toplam çalıştırılma süreleri elde edildi.

Çizelge 4.7 İlk gelen ilk çalışır (FCFS) yöntemi için başlangıçta bilgisayara verilen işler

	Emanetçi	P1	P2	P3	P4	P5	P6
case1	OTHERS	AVG	AVG	AVG	AVG	AVG	AVG
case2	OTHERS	MAX+	MAX+	MIN+	MIN+	MIN+	MIN+
case3	OTHERS	MAX+	MAX+	MIN+	MIN+	BOS	NOS
case4	OTHERS	MAX+	MAX+	BOS	BOS	BOS	BOS
case5	OTHERS	MAX+	MAX+	MAX+	MAX+	MAX+	MAX+

Çizelge 4.8 İlk gelen ilk çalıştırılır (FCFS) yöntemi ile farklı EnAzYük, EnÇokYük (min,max) değerleri için işlerin gerçekleme ile çalıştırılması için gereken süre

min,max	1,20	3,20	10,20	15,20	3,5	3,30	3,50
case1	2271	2052	1760	1750	2768	1979	1953
case2	2159	2080	1831	1792	2772	2075	2122
case3	2158	2022	1718	1878	2729	2028	2184
case4	2094	1923	1687	1699	2788	2035	2187
case5	2107	1963	1732	1736	2739	2035	2173

4.1.6 Gerçekleme ile Açık Artırma Testi

Herhangi bir uç bilgisayar iş yükü *AnlıkYük* değeri *EnAzYük* eşik değerinden küçük ise Emanetçi'den iş talebinde bulunmaktadır. Yavaş bilgisayar işleri alıp hızlı olan bilgisayarın boşta beklemesini önlemek için Emanetçi'den iş alırken açık arttırma yöntemi kullanılarak işler bilgisayarlara atanacaktır. Açık arttırma belirli zaman aralıklarında yapılacaktır. Ortalama iş uzunluğunun 1, 2, 3 katı birim sürelerde bilgisayarın iş isteyip istemediği (*AnlıkYük* değeri *EnAzYük* değerinden küçük ise iş istenecektir) beklenecektir. Bu açık arttırma süresi (ortalama iş uzunluğunun 1, 2, 3 katı) sonunda iş isteyen bilgisayarlar arasında işi en kısa sürede tamamlayabilecek bilgisayara iş atanır. Açık arttırma süresi sonunda iş isteyen bilgisayar varsa Emanetçi'de bekleyen işleri en kısa sürede tamamlayacak bilgisayara işler atanır.

Çalışmada Emanetçi ve 6 bilgisayardan oluşan WGS sistemine gerçek işler

verildi. Toplam iş sayısı 500 ve toplam çalıştırılması için gereken süre 5000 alındı. Bilgisayarların her biri birim zamanda 2 birim iş çalıştırmaktadır. Teorik olarak bütün işler eşit dağıtılabilseydi 5000 / (2*6) = 416 birim sürede tüm işler tamamlanabilecekti.

Başlangıçta bilgisayarlara atanan iş miktarları çizelge 4.9 da verilmektedir. Açık arttırma için beklenen süre (ortalama iş uzunluğunun 1, 2, 3 katı) değiştirilerek işlerin toplam çalışma süreleri çizelge 4.10'da verilmektedir. Farklı başlangıç durumları case1, case2, case3 için en küçük iş tamamlama süresi Auc=20 (ortalama iş uzunluğunun 2 katı) için bulunmuştur. Case4 için en küçük süre Auc=10 (ortalama iş uzunluğu) ve case5 için Auc=30 (ortalama iş uzunluğunun 3 katı) için en küçük süre bulunmuştur. Açık arttırma süresi (Auc) küçük seçilirse daha sık açık arttırma yapılmaktadır. Fakat az sayıda bilgisayar iş talebinde bulunabilmektedir. Açık arttırma değeri büyük seçilecek olursa çok sayıda bilgisayar açık arttırmaya katılacaktır.

Çizelge 4.9 Açık arttırma yöntemi için başlangıçta bilgisayara verilen işler

Açık Arttırma	Emanetçi	P1	P2	P3	P4	P5	P6
case1	2960	340	340	340	340	340	340
case2	3420	610	610	90	90	90	90
case3	3600	610	610	90	90	0	0
case4	3780	610	610	0	0	0	0
case5	1340	610	610	610	610	610	610

Çizelge 4.10 Farklı açık arttırma süreleri için gerçekleme ile işlerin çalıştırılması için gereken süre

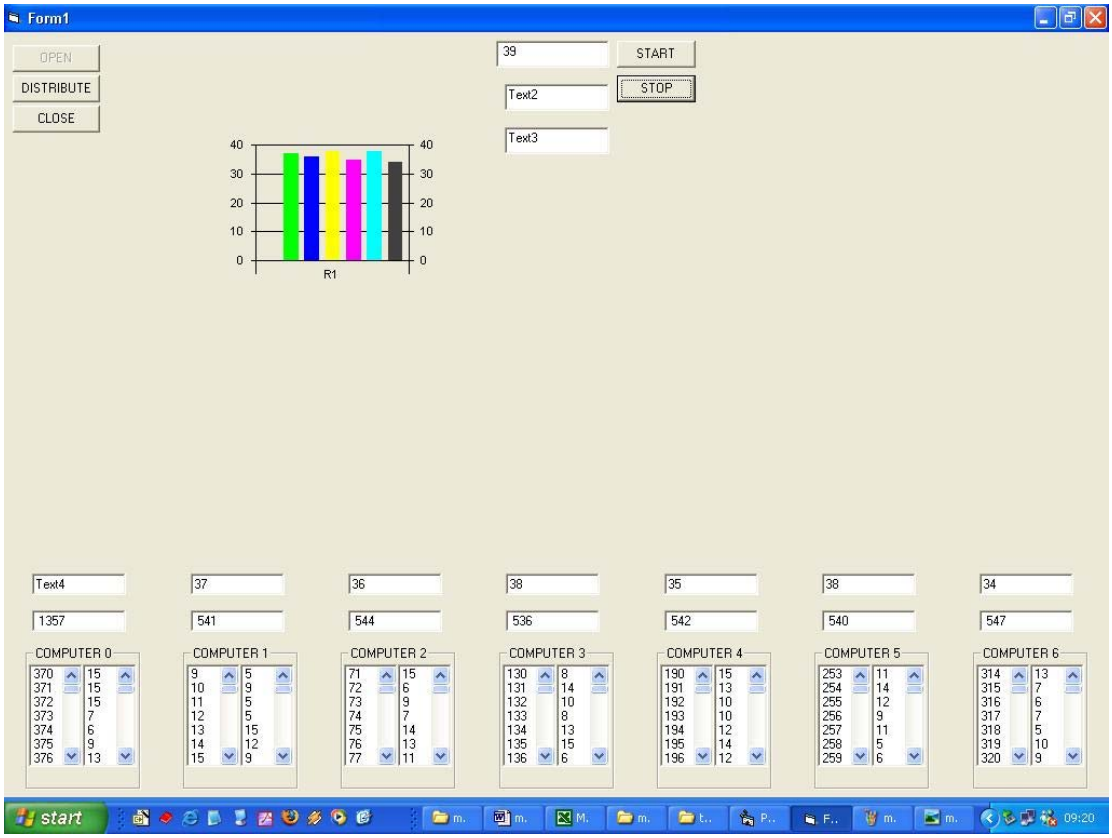
Auc	10	20	30
case1	468	452	477
case2	497	492	577
case3	517	474	520
case4	474	484	570
case5	461	469	446

4.2 Simülatör

Bir önceki bölümde geliştirilen model için simülatör programı Visual Basic kullanılarak yapıldı. Simülatör formunda OPEN, DISTRIBUTE, CLOSE, START,

STOP komut kutuları yerleştirildi. Emanetçi ve uç bilgisayardaki iş adları ve uzunlukları da liste olarak gösterilmektedir. Daha önce yapılan gerçekleme ile uyumlu olması için simülatörde Emanetçi ve 6 adet uç bilgisayar kullanıldı. Simülatör ekran görüntüsü şekil 4.6'da gösterilmektedir.

Emanetçi ve uç bilgisayarlara gelen işler (iş adları ve uzunlukları) listelerde tutulmaktadır. Son gelen iş listenin sonuna eklenmektedir. Çalıştırılan iş listenin başındaki iştir. Tamamlanan iş bilgisayar iş listesinden çıkarılır.



Şekil 4.6 Simülatör ekran görüntüsü

OPEN: Simülatör programının çalışması için kullanılır. Buna basmadan DISTRIBUTE ve CLOSE komutları çalışmaz.

DISTRIBUTE: Simülatör giriş dosyasını okur. Bu dosyada başlangıçta hangi işlerin hangi bilgisayarlara atanacakları vardır. Giriş dosyasının her satırı bir işi tanımlar. İşin adı, uzunluğu, hangi bilgisayara atanacağı verilmektedir. İşin adı 1, 500 arasında tam sayılarla, işin uzunluğu 5, 15 arasında tam sayılarla ve işin atanacağı bilgisayar 0, 6

arasında tamsayılarla verilir. Emanetçi 0 nolu bilgisayar olarak atanmıştır. Giriş dosyasındaki herhangi bir satır "11, 8, 4" ise, 11 nolu 8 birim uzunluğundaki iş 4 nolu bilgisayara (P4) atanmıştır. Her satırdan yeni bir iş okunurken, işin atandığı bilgisayarın iş yükü *EnÇokYük* değerinden küçük ise, yeni gelen iş bilgisayarın iş listesine eklenir. Eğer yeni okunan işin atanacağı bilgisayarın iş yükü *EnÇokYük* değerinden büyük ise, yeni gelen iş Emanetçi'ye atanacaktır.

CLOSE: Simülatör programının çalışması tamamlandıktan sonra log bilgilerini çıkış dosyasına yazar.

START: Simülatör için zamanlayıcıyı çalıştırır.

STOP: Simülatör için zamanlayıcıyı durdurur.

Simülatör programının normal çalışması için sırasıyla OPEN, DISTRIBUTE, START, STOP, CLOSE komut tuşlarına basılmalıdır. OPEN ile simülatör çalışmaya hazırlanır. DISTRIBUTE ile işleri tanımlayan giriş dosyası okunarak işler istenen bilgisayarlara atanır. START ile bilgisayarlardaki işler yürütülür. Her birim zamanda bilgisayarlardaki işlerin tamamlanacak miktarları azaltılır. İşin çalıştırılacak kısmı 0'a ulaşınca iş bilgisayarın listesinden çıkarılır ve sıradaki iş alınır. Eğer bilgisayarın *AnlıkYük* değeri *EnAzYük* değerinden küçük ise Emanetçi'den iş talebinde bulunulur. Bütün bilgisayarlardaki işler tamamlandığında STOP ile simülatör durdurulur. CLOSE ile sonuçlar çıkış dosyalarına yazılır.

Bir önceki bölümde açıklanan WGS modeli simülatör ile gerçeklendi. Simülasyonda 6 tane işlemci ve Emanetçi kullanıldı. Ortalama iş uzunluğu 10 (her bir iş uzunluğu 5 ile 15 arasında değişen) olan 1000 adet iş kümesi kullanıldı. Başlangıçta bilgisayarlardaki iş miktarı çizelge 4.7 de verildiği gibi alındı. Simülatör de her bilgisayar kendilerine atanan işleri çalıştırmakla yükümlüdür.

Herhangi bir anda bir bilgisayarın *AnlıkYük* değeri *EnAzYük* eşik değerinin altına indiğinde bilgisayar WGS sisteminden iş talebinde bulunur. Emanetçi daha önce hiçbir bilgisayar tarafından alınmayan işlerden sıradakini verir. Herhangi bir anda sadece bir bilgisayar iş isteğinde bulunuyorsa sadece ona iş verilebilir. Eğer aynı anda birden fazla bilgisayar iş talebinde bulunursa kime, ne kadar, nasıl iş verileceğine iş dağıtma, çizelgeleme yöntemleri karar verecektir. İlk gelen ilk iş alır (*First Come First Serve*, FCFS) ve açık artırma (*Auction*) ile iş dağıtma yöntemleri uygulandı.

Daha önce bölüm 4.1.4'de gerçekleme ile *EnAzYük* ve *EnÇokYük* değerlerinin bulunması için yapılan bütün testler aynı şartlarda (*EnAzYük*, *EnÇokYük*, iş sayısı, ortalama iş büyüklüğü) simülatör ile yapıldı. Başlangıçta bilgisayarlara verilen iş miktarları çizelge 4.4'de verildiği gibi alındı. Bütün işlerin tamamlanması için gereken süreler çizelge 4.5 ve çizelge 4.6'da verilen sonuçlar ile aynı olarak elde edilmiştir.

Daha önce bölüm 4.1.5'de gerçekleme ile ilk gelen ilk iş alır (FCFS) ve bölüm 4.16'da açık artırma (*Auctio*n) iş çizelgeleme yöntemlerinin testi için yapılan bütün testler aynı şartlar altında simülatör ortamında yapıldı. Başlangıçta bilgisayarlara verilen iş miktarları çizelge 4.7'de ilk gelen ilk iş alır ve çizelge 4.9'da açık artırma için gerçeklemedeki gibi alındı. Bütün işlerin tamamlanması için gereken süreler çizelge 4.8'de ilk gelen ilk iş alır yöntemi ve 4.10'da açık artırma yöntemi için alınmıştır.

Sonuç olarak daha önce gerçeklemesi yapılan bütün durumlar için aynı testler simülatör ortamında yapılmış ve aynı sonuçlar elde edilmiştir. Bundan sonra daha fazla test yapabilmek için simülatör kullanılacaktır.

4.3 Teorik Olarak Hesaplama

Başlangıçta tüm bilgisayarlarda hiçbir iş yoktur. Bütün işler Emanetçi'ye verilmektedir. WGS modelinde kullanılan *EnÇokYük* (max) değeri bilgisayarlarda olabilecek en fazla iş yükü değeridir. *EnAzYük* (min) değeri ise bilgisayarlarda olabilecek en küçük iş miktarıdır. Bir işin tamamlanması için gereken süre işin çalıştırılması için gereken süre (t_{exec}) ile işin Emanetçi'den çalıştırılan bilgisayara taşınması (iletişim) için gereken sürenin ($t_{iletişim}$) toplamı olacaktır. Kullanılan test işlerinin tamamının çalıştırılması için gereken süre 9714 birim süredir. WGS modelinin gerçeklemesi için kullanılan uç bilgisayar sayısı (P) 6 adettir.

İletişim katkısı: (*max-min*) kadar iş atanırken boşta beklenilen iletişim zamanıdır.

```
if ((max-min) * m + setup > (max-min)) then
      iletişim_katkısı=(max-min) * m + setup
else
      iletişim_katkısı=0
```

Setup değeri iki bilgisayar arasında iletişim kurmak için gereken süreyi göstermektedir. İki bilgisayar arasında gönderilen verinin işin çalıştırılması süresi ile arasındaki oran iletişim oranı olarak adlandırılacaktır. İletişim oranı (m) (0, 0.2, 0.3, 0.5, 0.7, 1) değerlerini almaktadır. İletişim oranı küçük boyutlu veriler (*small data*, SD) için m=0, büyük boyutlu veriler (*large data*, LD) için m=0.2 ve çok büyük boyutlu veriler (huge data, HD) için m=1 alınmıştır.

İlk turda Emanetçi'den uç bilgisayarlara alınan iş miktarı bilgisayarlar boş olduğundan *EnÇokYük (max)* kadar olacaktır. İlk tur sonunda bilgisayarda kalan iş *EnAzYük* kadar olacak, çalıştırılan iş miktarı (*EnÇokYük –EnAzYük) = max -min* olacaktır. İlk tur için gereken iletişim

```
ilk_turda_alinan_iş_miktarı = max
ilk_turdan_kalan_iş_miktarı = min
ilk_tur_exec = max - min
ilk_tur_iletişim = max * m + setup
```
dir

İlk turdan sonra sistemde kalan iş miktarı

```
Toplam_iş – ilk_turda_alınan_iş miktarı * P
```
kadardır.

İlk turdan sonra yapılan iş atama ve çalıştırmalar "ara TUR" olarak isimlendirilecektir. Ara TUR sayısı:

```
ara_TUR_sayısı = INT (İlk_turdan_kalan_iş_miktarı / (max - min) * P).
```

Burada INT iki sayının bölümünden elde edilen tam sayıyı göstermektedir.

```
son_tur_toplam_iş = MOD (ilk_turdan_kalan_iş miktarı, (max-min) * P)
```

MOD: iki sayının bölümünden elde edilen kalan miktarı göstermektedir.

```
ara_tur_exec_toplam = (max-min) * ara_TUR_sayısı
ara_tur_iletişim_toplam =  iletişim_katkısı * ara_TUR_sayısı

son_tur_dolu_olan_P_sayısı =  INT(son_tur_toplam_iş / (max-min))

son_tur_tam_dolu_exec_toplamı = max -min
son_tur_tam_dolu_iletişim_toplamı = iletişim_katkısı

son_tur_yarım_dolu_exec = MOD (son_tur_toplam_iş, (max-min) )

if (son_tur_yarım_dolu_exec * m + setup ) >  min then
      son_tur_yarım_dolu_iletişim_katkısı  =  son_tur_yarım_dolu_exec  *
                                              m + setup
else
      son_tur_yarım_dolu_iletişim_katkısı = 0

toplam_avg_exec = ilk_tur_exec + ara_tur _exec_toplam +
                  (son_tur_tam_dolu_exec_toplam *
                   son_tur_tam_dolu_P_sayısı +
                   son_tur_yarım_dolu_exec) / P

toplam_avg_iletişim = ilk_tur_iletişim + ara_tur_iletişim_toplam +
            (son_tur_tam_dolu_iletişim * son_tur_tam_dolu_P_sayısı +
             son_tur_yarım_dolu iletişim_katkısı) / P

toplam_avg = toplam_avg_exec + toplam_avg_iletişim
```

N : Uç bilgisayarın sayısı 6 alınmıştır.

T_i : Uç bilgisayar i 'de bütün işlerin tamamlanması için gereken süresi

T_{avg} : Uç bilgisayarlardaki işlerin ortalama tamamlanma süresi,

T_{teo} : İşlerin ideal olarak eşit miktarda uç bilgisayarlarda çalıştırılması ile bütün işlerin tamamlanması için gereken süre

T_{toplam} : Bütün işlerin tamamlanması için gereken süre

$$T_{teo} = \frac{T_{toplam}}{N}$$

$$T_{avg} = \frac{1}{N}\sum_{i=1}^{N} T_i$$

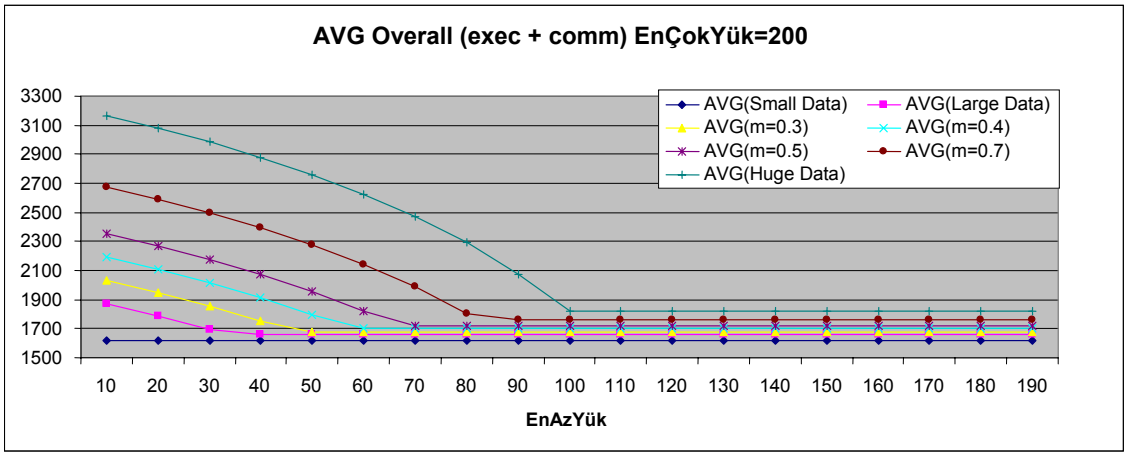
$$Error1 = \sum_{i=1}^{N} \left| T_i - T_{avg} \right|$$

Bir önceki bölümde açıklanan WGS modeli teorik hesaplama formülleri baz alınarak testler tekrarlandı ve bölümün izleyen bölümlerindeki testler tekrarlanarak farklı iş dağıtma modelleri ve iletişimin performansa etkileri incelendi. Bu testlerde 6 tane işlemci ve Emanetçi kullanıldı. Ortalama iş uzunluğu 10 (her bir iş uzunluğu 5 ile 15 arasında değişen) olan 1000 adet iş kümesi kullanılmıştır. Başlangıçta hiçbir işlemcide herhangi bir iş bulunmadığı durum ele alındı. Bu yüzden bütün işler Emanetçi'de toplanmıştır.

Herhangi bir anda bir uç işlemcinin *AnlıkYük* değeri *EnAzYük* eşik değerinin altına indiğinde işlemci sistemden iş talebinde bulunur. Emanetçi daha önce hiçbir işlemci tarafından alınmayan işlerden sıradakini verir. Herhangi bir anda sadece bir işlemci iş isteğinde bulunuyorsa sadece ona iş verilebilir. Eğer aynı anda birden fazla işlemci iş talebinde bulunursa kime, ne kadar, nasıl iş verileceğine iş dağıtma, çizelgeleme yöntemleri karar verecektir. FCFS ve açık arttırma (*Auction*) ile iş dağıtma yöntemleri uygulandı.

4.3.1 İlk Gelen İlk İş Alır İş Dağıtma

Bölüm 4.1.5'de ilk gelen ilk iş alır (*First come first serve*, FCFS) iş dağıtma yöntemi detaylı açıklandı. İşlemcinin *AnlıkYük* değeri *EnAzYük* eşik değerinin altında ise işlemci Emanetçi'den iş talebinde bulunur. Emanetçi'de yeteri kadar iş varsa ilk talep eden işlemciye *EnÇokYük* – *AnlıkYük* miktarı kadar iş verilir. İş dağıtma işleminden sonra bütün işlemciler *EnÇokYük* kadar iş yükü ile taşırlar.



Şekil 4.7 Teorik *EnÇokYük*=200 için ortalama iş tamamlama süreleri

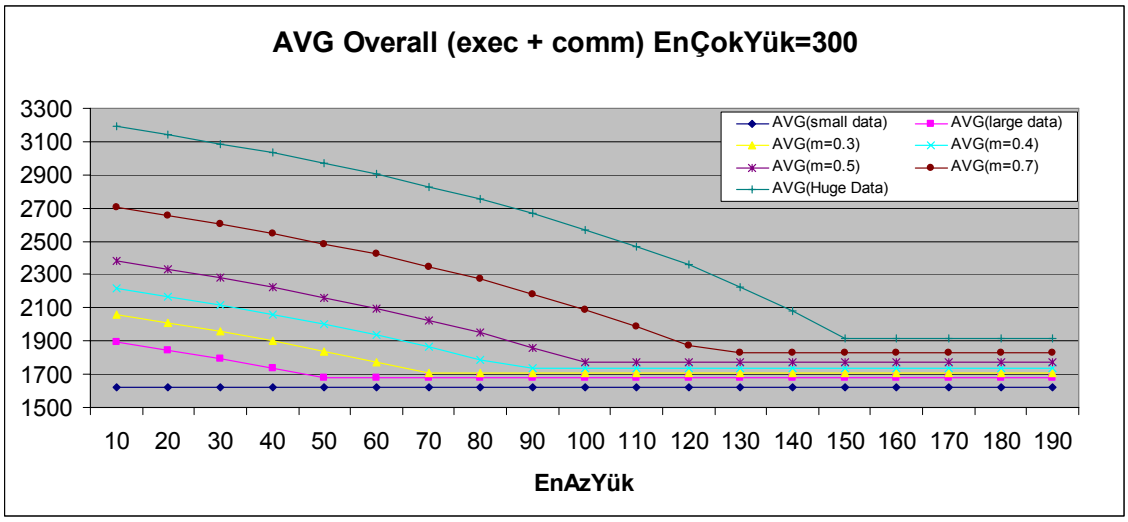
EnÇokYük 200 (ortalama iş uzunluğunun 20 katı) birim alındı. *EnAzYük* değeri 10, 20, 30, ...190 (ortalama iş uzunluğunun 1, 2, 3, .., 19 katı) kadar denenerek her bir işlemcide tamamlanan işlerin ortalaması alındı. Sonuçlar Şekil 4.7'de verilmiştir.

AVG *overall executio*n (*exec* + *comm*): Genel çalıştırma ve iletişim sürelerinin toplamları her bir işlemci için bulunur. Küçük veriler (*Small Data*, SD) için iletişim etkisi alınmaz. Farklı m değerleri iletişimin ortalama iş çalıştırma süresinin (10 birim) yüzde kaçına denk geldiğini temsil etmektedir. m = 0 küçük iş uzunlukları (*small data*) için kullanılır. m=0.2 değeri büyük iş miktarı (*large data*, LD) ve m=1 değeri çok büyük işler (*huge data*, HD) için kullanılan değerlerdir.

Sabit *EnAzYük* değeri 10 (ortalama iş uzunluğu) için m iletişim faktörü artarken toplam iş çalıştırma süresi ortalaması da artmaktadır. İletişim faktörünü çok büyük veriler m=1 (huge data) için sabit alındığında *EnAzYük* değeri artarken toplam icra süresinin ortalaması azalmaktadır. Bunun nedeni iş dağıtma algoritmasında işlemcide *EnAzYük* eşik değeri kadar iş varken iş dağıtmaya başlamamızdır. Ayrıca iletişim ile çalıştırma aynı zamanda yapıldığında ek iletişim olmamaktadır. Örneğin: *EnAzYük* değeri 100 (ortalama iş uzunluğunun 10 katı) iken, *EnAzYük* – *EnÇokYük* = 200–100 = 100 birimlik iş. Ortalama 10 uzunluktaki 10 iş almak gerekir. Her bir iş için gereken iletişim etkisi de 10 birim olunca 10 adet iş için 100 birimlik iş alma süresi gerekecektir. İşlemci her zaman *EnAzYük* =100 (ortalama iş uzunluğunun 10 katı) kadar iş olacağından, işlemci içindeki işler çalıştırılırken Emanetçi'den de işler alınabilir. İş almak için ek bir iletişim süresine gerek yoktur. *EnAzYük* değeri 100 (ortalama iş

uzunluğunun 10 katı) den küçük değerler için ek bir iletişim yükü getirecektir. *EnAzYük* değeri 100 (ortalama iş uzunluğunun 10 katı) den büyük değerler için alınacak iş için gereken sürede işlemci üzerindeki işleri çalıştıracaktır. Sadece başlangıçta bütün işlemciler boş olduğundan ilk turda bütün işlemciler için ek iletişim süresi gerekecektir.

Sonuç olarak öyle bir *EnAzYük* değeri seçmeliyiz ki, işlemci onları çalıştırırken aynı zamanda da işleri getirmek için iletişim gerekmesin.

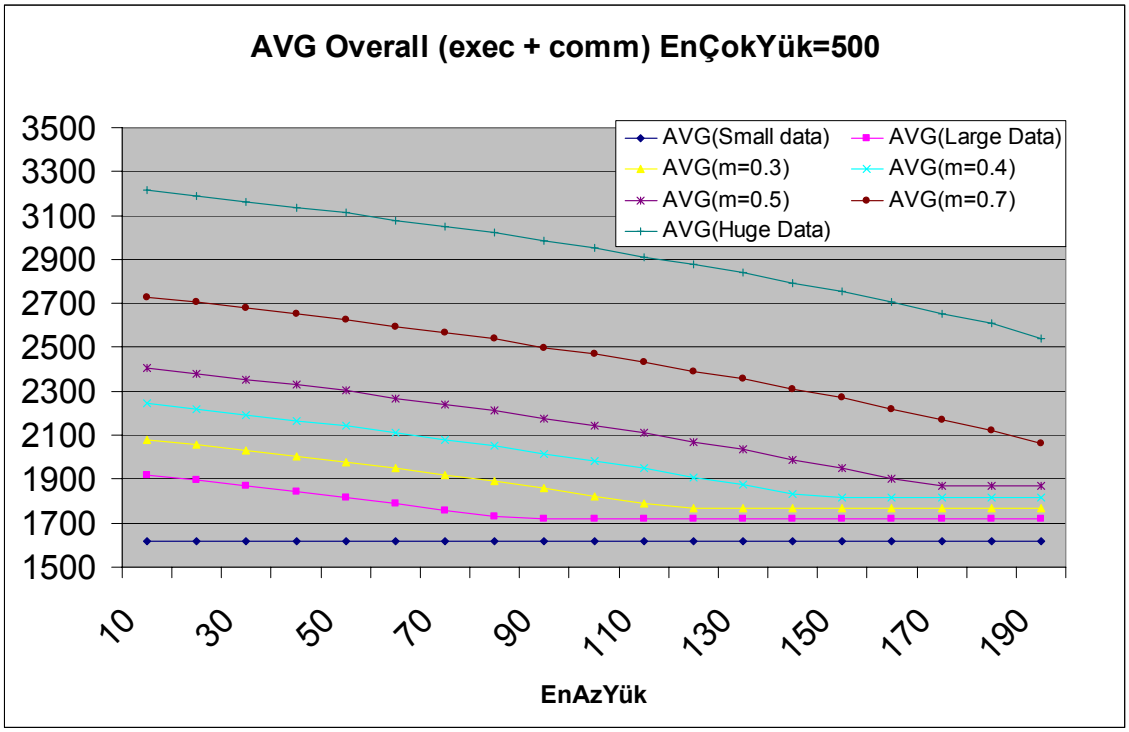


Şekil 4.8 Teorik *EnÇokYük*=300 için ortalama iş tamamlama süreleri

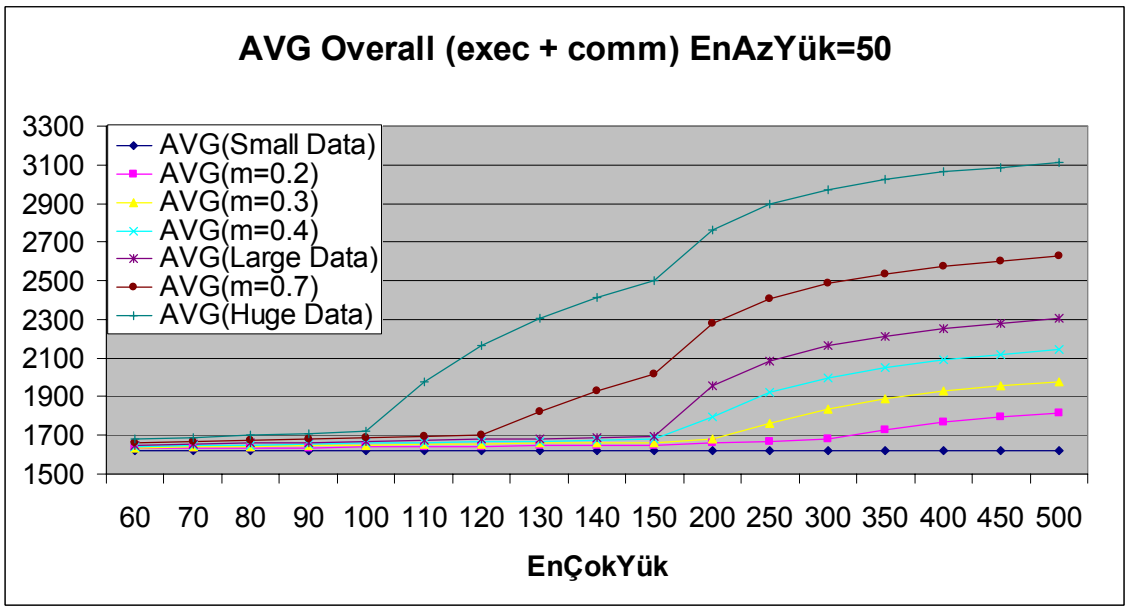
İletişim katkısı ile *EnÇokYük* arasındaki ilişkiyi görebilmek için *EnÇokYük* değeri arttırılmıştır. Farklı veri uzunlukları içinde iletişim katkısı değişmektedir. Bir önceki şekildeki (şekil 4.7) benzer açıklamalar Şekil 4.8 içinde geçerlidir. Şekil 4.8 için *EnÇokYük* değeri 300 (ortalama iş uzunluğunun 30 katı) alınmıştır. Burada da çok büyük işler için (*huge data,* HD) *EnAzYük* 150 (ortalama iş uzunluğunun 15 katı) değerine kadar iletişim etkisi görülmektedir.

İletişimin ağırlığını görebilmek için *EnÇokYük* değeri çok büyük alınırsa transfer edilecek veri için gereken süre çalıştırılacak işten çok fazla olacağında her zaman iletişim önemli olacaktır. Şekil 4.9'te de *EnÇokYük* değeri 500 (ortalama iş uzunluğunun 50 katı) alınarak ortalama iş tamamlama sürelerinin iletişime göre nasıl değiştiği gösterilmiştir.

Özet olarak, *EnÇokYük* değerini ne seçelim ki bilgisayarların iş yükleri dengeli



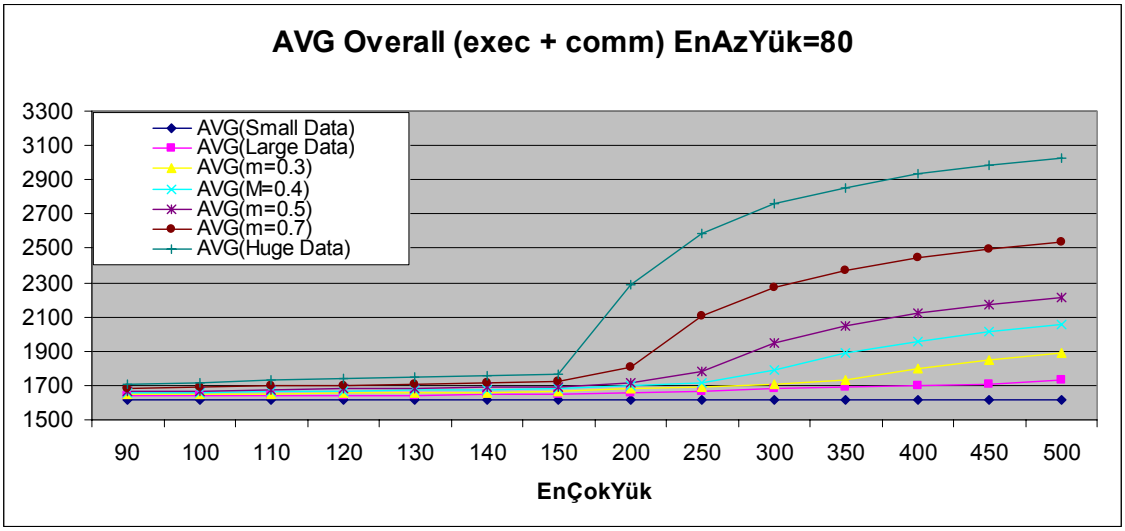
Şekil 4.9 Teorik *EnÇokYük*=500 için ortalama iş tamamlama süreleri



Şekil 4.10 Teorik *EnAzYük*=50 için ortalama iş tamamlama süreleri

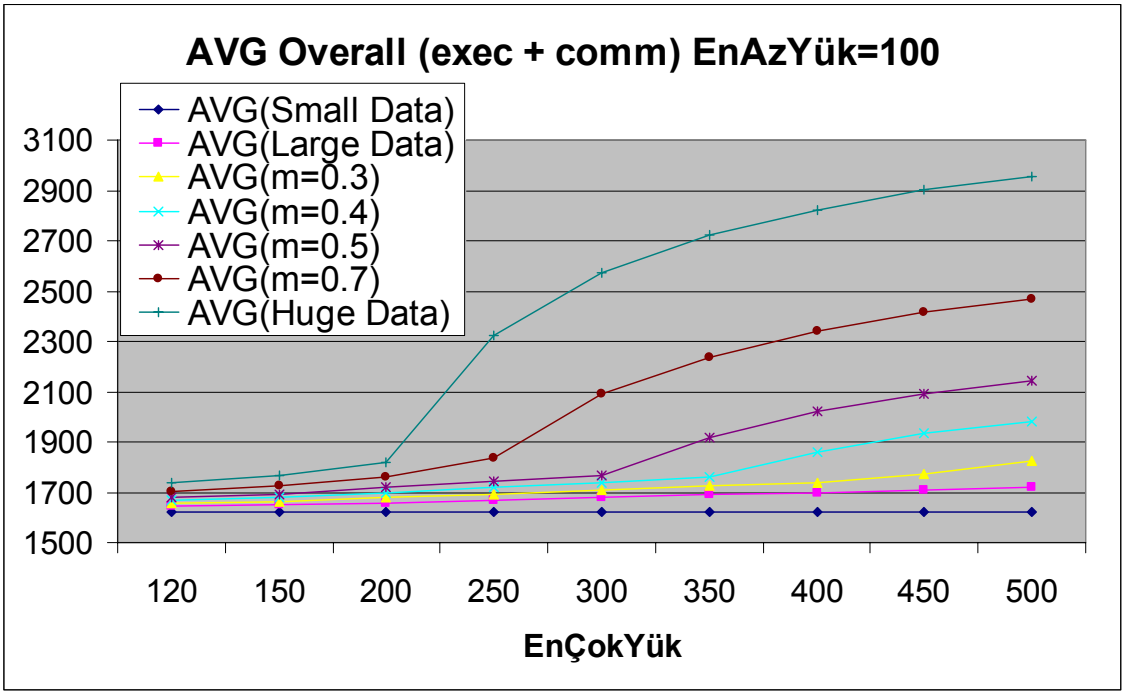
olsun ve ortalama iş tamamlama süreleri daha az olsun. Bu üç denemeden *EnÇokYük* değeri en küçük olan daha az iletişim süresi eklendiğinden, dolayısı ile daha az ortalama iş süresi getirdiği için daha iyi sonuç vermiştir.

Şekil 4.10'da *EnAzYük* değeri 50 (ortalama iş uzunluğunun 5 katı) alınarak farklı *EnÇokYük* değerleri için ortalama iş tamamlama süresi nasıl değiştiği görülmektedir. *EnÇokYük*'ün çok büyük değerleri için daha fazla iş tamamlama süreleri gerekmektedir. *EnÇokYük EnAzYük*'ün 2 katından büyük olduğu sürece ek iletişim süresi getirmektedir.



Şekil 4.11 Teorik *EnAzYük*=80 için ortalama iş tamamlama süreleri

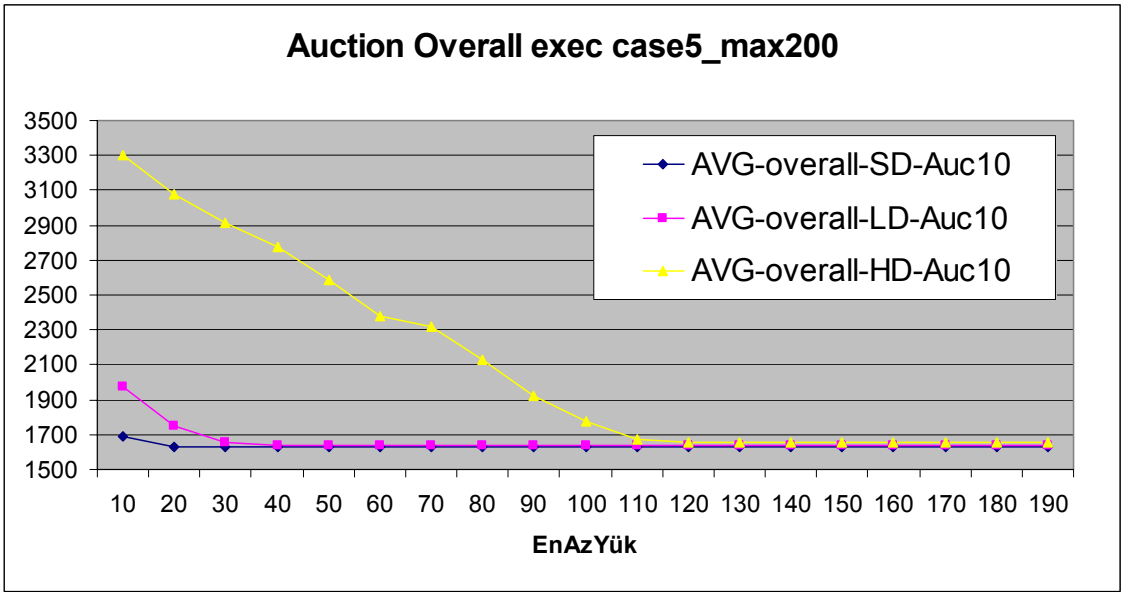
Şekil 4.11 ve Şekil 4.12 da *EnAzYük* değeri 80 (ortalama iş uzunluğunun 8 katı) ve 100 (ortalama iş uzunluğunun 10 katı) için değişen *EnÇokYük* değerlerinde ortalama iş tamamlanma sürelerinin nasıl değiştiği görülmektedir.



Şekil 4.12 Teorik *EnAzYük*=100 için ortalama iş tamamlama süreleri

4.3.2 Açık Artırma Yöntemi

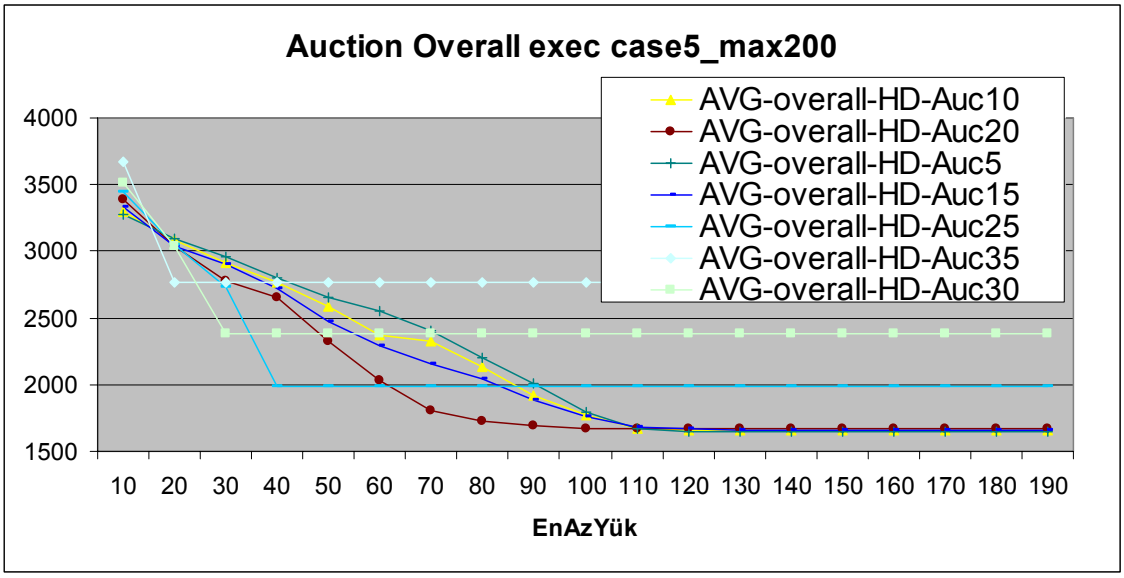
Teorik olarak FCFS ile ilk iş talebinde bulunan işlemciye *EnÇokYük – AnlıkYük* değeri kadar iş verilmektedir. *EnÇokYük* ve *EnAzYük* arasındaki fark büyüdüğü zaman işlemcilere dengesiz iş atama yapılacaktır. Bu da özellikle son iş dağıtma anında kendisini gösterecektir. *EnÇokYük – AnlıkYük* alan işlemciler çalışırken, bazı işlemcilere hiç iş kalmayacaktır. Sonuç olarak işlemcilerin ortalama iş tamamlama sürelerinde farklılıklar olacaktır. Buna çözüm olarak her alınacak iş için açık artırma yapılmasıdır. Belirli aralıklarla iş dağıtmak için açık artırma yapılacaktır. Bu esnada iş talebinde bulunacak işlemciler varsa Emanetçi'den alınacak her işi en kısa sürede tamamlayabilecek işlemciye verilecektir. Açık artırma her 10 (ortalama iş uzunluğu) birim sürede yapılmıştır. İşlemcilerin *AnlıkYük* değerleri *EnAzYük* değerinin altında ise iş talebinde bulunabilecektir.



Şekil 4.13 Açık artırma ile ortalama iş tamamlama süreleri

Şekil 4.13'de açık artırma ile yapılan testlerde ortalama iş tamamlama süreleri gösterilmektedir. *EnÇokYük* değeri 200 (ortalama iş uzunluğunun 20 katı) alınmıştır. Her 10 (ortalama iş uzunluğu) birim sürede açık artırma yapılmıştır. SD küçük işler için, LD büyük işler için, HD ise çok büyük işler için alınan değerleri göstermektedir. Küçük işleri Emanetçi'den taşımak için ek iletişim gerekmemektedir. Büyük işler için iletişim etkisi *EnAzYük* küçük değerleri için gözükmektedir. *EnAzYük* değeri büyüdüğünde iletişim yapılırken işlerin çalıştırılması da yapılacağından ek olmayacaktır. *EnAzYük* küçük iken daha fazla iş alınabileceğinden ek iletişim gerekecektir. *EnAzYük* yeteri kadar büyük olduğunda gereken iletişim çalıştırılan işlerle aynı anda yapılabildiğinden ek bir süre getirmeyecektir.

Açık artırmayı 5, 10, 15, 20, 25, 30, 35 gibi farklı sürelerde yaptığımızda çok büyük işler için ortalama iş tamamlama sürelerinin sonuçları Şekil 4.14 de verilmiştir. 5, 10, 15 20 değerleri için *EnAzYük* seviye 110'dan sonra ek iletişim süresi getirmemektedir. Oysa 25, 30, 35 değerlerinde ise *EnAzYük* değeri değişirse bile ek ortalama iş tamamlama süreleri artmaktadır.

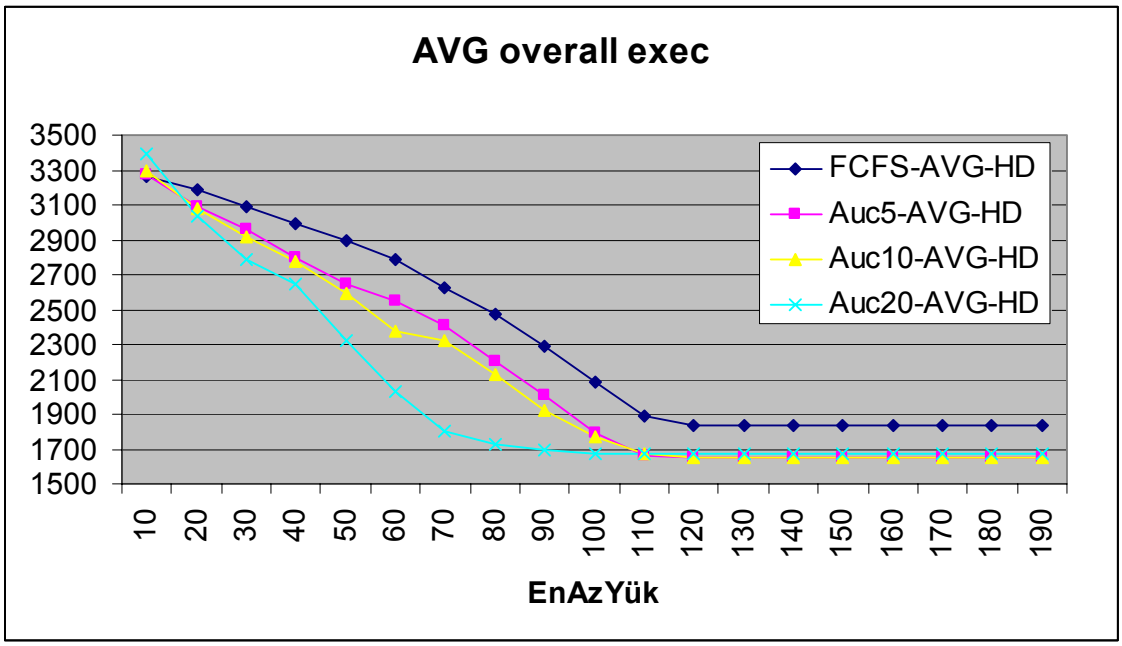


Şekil 4.14 Farklı açık artırma ile ortalama iş tamamlama süreleri

4.3.3 FCFS ile Açık Artırma Karşılaştırılması

Aynı test kümeleri, *EnAzYük*, *EnÇokYük*, çok büyük işler (*huge data*, HD) için ortalama iş tamamlama süreleri FCFS ve açık artırma için şekil 4.15'de verilmiştir. Açık artırma sonuçları (Auc5, Auc10, Auc20) arasından Auc20 en iyi sonucu vermektedir.

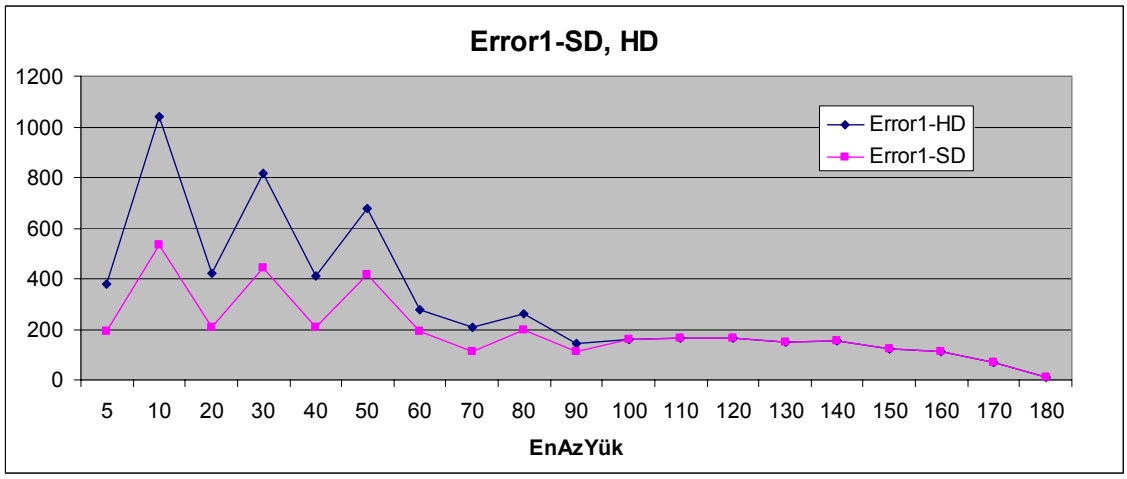
FCFS veya açık artırma arasında tercih yapmamız gerekiyorsa açık artırma daha iyi sonuç vermektedir.



Şekil 4.15 FCFS ve açık artırma ile karşılaştırma

4.3.4 Yük Dağılımı

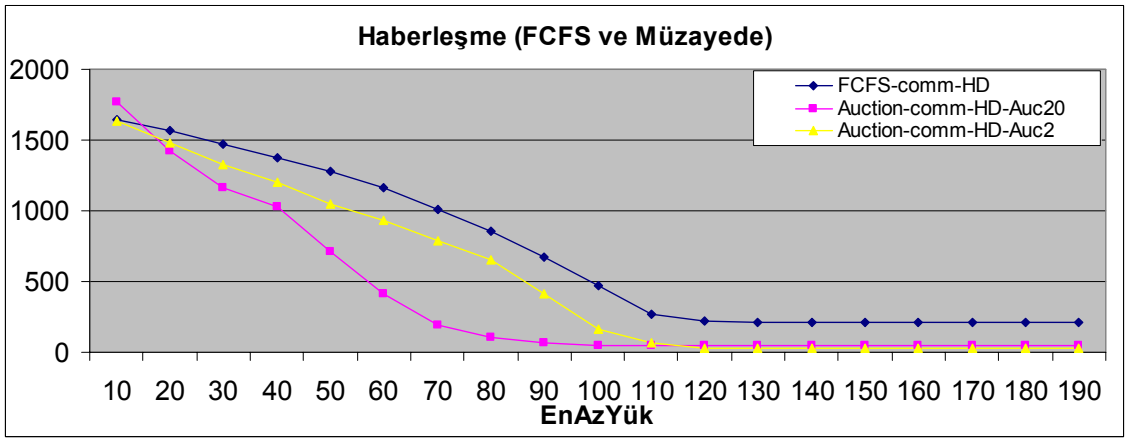
Teorik çalışma ile işlemcilerde yüklerin nasıl dengeli dağıldığını gösteren değerler Şekil 4.16'da gösterilmiştir. Her bir P_i işlemcisinin toplam iş çalıştırma sürelerinin ortalaması A olsun. Her P_i işlemcisinin toplam iş çalıştırma süresinin ortalama değer A'dan farklarının mutlak değerlerinin toplamı Error1 hatasını verecektir. Bu değer işlemcilerin farklı *EnAzYük* değerleri için ne kadar dengeli dağıldığını gösterecektir. Eğer işlemci yükleri ortalamaya çok yakın ise bu Error1 hata değerleri çok küçük olacaktır. Eğer işlemciler ortalama iş çalıştırma sürelerinden uzakta iseler, Error1 hata da büyük olacaktır. Şekil 4.16'da küçük işler için Error1-SD, çok büyük işler için Error1-HD görülmektedir. Çok büyük işler (HD) için *EnAzYük* değeri 100 (ortalama iş uzunluğunun 10 katı)'den küçük olduğu yerlerde ek iletişim süreleri içerdiği için Error1 değerleri de yüksek olacaktır. İletişim etkisinin olmadığı yerlerde (SD) Error1 değerleri daha küçüktür.



Şekil 4.16 FCFS küçük ve çok büyük işler için Error1 grafiği

4.3.5 FCFS ve Açık Artırma İçin İletişim

Çok büyük (HD) işleri Emanetçi'den almak için ek iletişim süresine ihtiyaç vardır. Eğer işlemci üzerinde yeteri kadar iş yükü varsa işleri çalıştırırken aynı zamanda Emanetçi ile haberleşerek işleri alabilecektir. Eğer işlemcide çok az iş yükü olursa alınan işler için ek iletişim sürelerinin eklenmesi gerekecektir. İşlemcinin iş dağıtma ile iş alırken *EnAzYük* eşik seviyesi kendisinin iş yükünü belirleyecektir. *EnAzYük* değeri alınacak işin iletişim süresini karşılayabilecek kadar büyük ise işler çalıştırılırken iletişim de olacaktır. Yoksa ek iletişim gecikmesi olacaktır. FCFS ve açık artırma için farklı *EnAzYük* değerleri için iletişim Şekil 4.17'de gösterilmiştir.



Şekil 4.17 FCFS ve açık artırma için iletişim

4.3.6 Değerlendirme

Bir önceki bölümde açıklanan WGS modelinin gerçeklemesi yapıldı. Modeli farklı parametre ve yöntemlerle test edebilmek için simülatör yapıldı. Gerçekleme ve simülatör sonuçları aynı elde edildi. Daha sonra geliştirilen model için teorik hesaplama yapılarak simülatör ve gerçekleme sonuçlarını desteklediği görülmüştür.

Geliştirilen modelde uç bilgisayarların iş yüklerinin *EnAzYük* ve *EnÇokYük* eşik değerleri arasında kalacak şekilde işleri dağıtarak bütün işlerin bilgisayarlarlara dengeli olarak dağıtılması hedeflenmiştir. Küçük *EnAzYük* değerleri için bazı bilgisayarlar iş alırken bazıları boşta bekleyebilecektir. Bu durumda bilgisayarlardaki işler dengeli dağılmayacaktır. *EnÇokYük* değeri *EnAzYük* değerine yakın olduğunda uç bilgisayarlar sürekli iş talebinde bulunacaklardır. *EnÇokYük* değeri *EnAzYük* değerinden çok büyük olduğunda bilgisayarlara daha fazla iş transfer etmek gerekeceğinden bazı bilgisayarlar iş çalıştırırken bazıları da boşta bekleyebilecektir. Bu da sistemin toplam iş çalıştırma süresinin artmasına neden olacaktır.

Geliştirilen yapıda kullanılan *EnAzYük* ve *EnÇokYük* değerlerinin iletişim katkısına etkisi değişmektedir. İletişim etkisi *EnÇokYük* değeri çok büyük alınırsa transfer edilecek veri için gereken süre çalıştırılacak işten çok fazla olacağından her zaman önemli olacaktır. Bu nedenle *EnÇokYük* değeri çok büyük seçilmemelidir. Alınacak iş miktarını *EnÇokYük – EnAzYük* değeri ile bulunurken, bunları işlemciye getirmek için gereken iletişim süresi *EnAzYük* değerinden küçük olduğunda hem işlemcideki işler çalıştırılabilecek hem de iletişim ile işler alınacaktır.

WGS modelinde ilk gelen ilk servis alır ve açık artırma iş çizelgeleme yöntemleri çalışılmıştır. Açık arttırma iş çizelgeleme yöntemi ile işler en kısa sürede tamamlanacak bilgisayara verildiğinden işlerin en kısa sürede tamamlanması desteklenmiştir ve uç bilgisayarlara dengeli iş dağıtımı sağlanmıştır.

Geliştirilen model'de sisteme verilen işler işi çalıştıran uç bilgisayarların

arızalanması durumunda çalışan diğer bilgisayarlar tarafından tamamlanmaktadır. Çalışan en az bir bilgisayar kaldığı sürece verilen tüm işler tamamlanmaktadır. Uç bilgisayarların sistemden ayrılması sistem performansını etkilemediği gerçekleme ile desteklenmektedir.

BÖLÜM 5

GENEL SONUÇLAR

Bu çalışmada web servisler kullanılarak tanımlı iş grubunu yerel ağa bağlı heterojen bilgisayar grubuna dağıtacak model (Web servis tabanlı Grid sistemi, WGS) geliştirilmiştir. Geliştirilen modelin hataya duyarlılığı test edilip gerçeklenmiştir. WGS' ye bağlı bilgisayarlarda hata oluşunca, hatalı işlemciye verilen işler sistemdeki diğer işlemciler tarafından tamamlandığı gerçekleme ile gösterilmiştir. Verilen işleri yapabilecek bir işlemci kaldığında da hatalı işlemcilere verilen tüm işlerin tamamlandığı gerçeklenerek gözlenmiştir.

Geliştirilen yapıda kullanılan *EnAzYük* ve *EnÇokYük* eşik değerlerinin sistemin toplam iş çalıştırılması süresini nasıl etkilendiği araştırılmıştır. Uç bilgisayarların farklı başlangıç iş yükleri için sistemin performansını en iyi yapan gerçekleme değerleri araştırılmıştır. İlk olarak sabit *EnAzYük* eşik değeri için *EnÇokYük* eşik değeri değiştirilerek toplam iş çalıştırma süreleri bulunmuştur. Sistemin en iyi performansı *EnÇokYük* değeri ortalama iş uzunluğunun 20 katı için elde edilmiştir. Bulunan *EnÇokYük* değeri sabit alınarak farklı *EnAzYük* değerleri için toplam iş çalıştırma süreleri incelenmiştir. *EnAzYük* eşik değeri ortalama iş uzunluğunun 10 katı ve *EnÇokYük* eşik değeri ortalama iş uzunluğunun 20 katı alınarak en iyi toplam iş çalıştırma süresi elde edilmiştir. Geliştirilen yapıda ilk gelen ilk servis edilir (FCFS) iş çizelgeleme yöntemi ile farklı *EnAzYük* ve *EnÇokYük* değerleri alınarak gerçeklenmiştir.

Geliştirilen WGS modelinde testleri tüm alt seçenekleri ile gerçekleştirmek için ayrıca WGS simülatörü geliştirilmiş ve simülatör ile farklı iş dağıtım yöntemleri test edilmiştir. İlk iş talebinde bulunan ilk iş alır (FCFS) yöntemi ile ilk iş talebinde bulunan işlemciye Emanetçi'den iş verilmektedir. Daha önce gerçeklemesi yapılan bütün durumlar için aynı testler simülatör ortamında yapılmış ve aynı sonuçlar elde edildiği görülmüştür.

Ayrıca geliştirilmiş olan WGS sisteminde toplam iş çalıştırma sürelerini üreten teorik model çıkarılmış ve pratik, simülatör sonuçları ile uyumlu olduğu gösterilmiştir. Teorik sonuçlara göre parametre seçimlerinde kullanmak üzere grafikler

oluşturulmuştur. Farklı *EnAzYük* ve *EnÇokYük* eşik değerleri için sistemin performansı ve işleri getirebilmek için gereken ek süreler çalışılmıştır. Alınacak iş miktarını *EnÇokYük – EnAzYük* değeri ile bulunurken, bunları işlemciye getirmek için gereken iletişim süresi *EnAzYük* değerinden küçük olduğunda hem işlemcideki işler çalıştırılabilecek hem de iletişim ile işler alınacaktır. Açık arttırma ile iş dağıtma yönteminde farklı açık artırma aralıkları için simülatör testleri yapılmıştır.

FCFS ve açık artırma yöntemi aynı iş grubu için test edilmiştir. Ortalama iş uzunluğunun 2 katı büyüklükte birim aralıklarla yapılan açık artırma diğer aralıklara göre en iyi sonucu vermiştir. Bu sonuç açık artırma yönteminin FCFS iş dağıtma yönteminden çok daha kısa sürede işleri tamamlayacağını göstermiştir.

Geliştirilen Modelin Desteklediği Özellikler

Geliştirilen modelde her uç bilgisayara çalıştırılabilecek işler verilmektedir. WGS modelinde iş seviyesinde (*coarse grain*) paralellik yapılmaktadır. Her uç bilgisayarda farklı işler aynı anda yürütülmektedir. WGS sistemine işler herhangi bir işlemci üzerinden verilebilmektedir. İşleri dağıtan merkezi bir yapı yoktur. Kullanıcı istediği işi istediği bilgisayara verebilir. Ancak uç bilgisayarlar *EnÇokYük* değerinden fazla iş alamazlar. WGS modelinde işler dağıtık olarak işlemcilere dağıtılmaktadır. Ayrıca işlerin bir kopyası da Emanetçi'ye gönderilir. İşlerin çalıştırılma anında hangi aşamada olduğu takip edilmemektedir. Geliştirilen WGS modeli farklı iş çizelgeleme yöntemlerini desteklemektedir.

WGS modeli bölüm 3'de detaylı olarak web servislerle nasıl gerçeklendiği açıklandı. Gerekli web servisler ve uygulamalar uç bilgisayarlara yüklenmektedir. Bilgisayarlar, kullanıcılar, Emanetçi ve Dış Erişim Birimi arasında iletişim Web Servisleri ile sağlanmaktadır.

WGS modelinde verilen tüm işlerin en az bir işlemci olduğu sürece tamamlanacağı garanti edilmektedir. Kullanıcı tarafından verilen işlerin bir kopyası Emanetçi'de tutulmaktadır. Emanetçi'de uç işlemcilere verilen işlerin ne zaman tamamlanacağı tutmaktadır. Herhangi bir nedenle zamanında tamamlanamayan işler Emanetçi tarafından başka uç bilgisayarlara verilir. Bu şekilde verilen tüm işler tamamlanıncaya kadar süreç devam eder. Sonuç olarak WGS modeline verilen tüm işler

çalıştıracak en az bir işlemci olduğu sürece tamamlanır.

WGS modelinde diğer yöntemlerin aksine birçok noktadan dağıtık olarak iş verilebilmektedir. Kullanıcılar her uç bilgisayardan WGS modeline iş verebilirler. WGS modelinin diğer sistemlerle entegrasyonu Dış Erişim Birimi ile yapılmaktadır. Bu birim ile diğer küme, Grid ve dağıtık sistemlerle iş alış verişi yapılabilmektedir. WGS modelinde uç bilgisayarlar isteğe bağlı (*non-dedicated*) çalışmaktadırlar. Uç bilgisayarlar istedikleri zaman WGS'nin üyesi olabilir, istedikleri zaman WGS'den ayrılabilirler. Bunun için diğer sistemlerin (küme, Digipede, Alchemi) aksine ek bir şey yapmalarına gerek yoktur. Uç bilgisayarlar için kayıt ve takip mekanizması bulunmamaktadır. Uç bilgisayarlar sisteme dahil olduklarında yapılacak iş varsa alıp çalıştırırlar. Uç bilgisayarlar sistemden herhangi bir anda ayrılabilirler ve kendilerine daha önceden atanan işler diğer uç bilgisayarlar tarafından tamamlanır.

WGS sistemi web servislerini çalıştırabilen tüm platformlarda kullanılabilir. Bu nedenle hem donanım hem de yazılım olarak heterojen yapıları desteklemektedir. WGS de bilgisayarlar arası iletişim Web Servisleri ile yapıldığından düşük performanslı ağ bağlantısı ile bilgisayarlar birbirleriyle bağlıdır. WGS modelini oluşturan bilgisayarlar birden fazla ağda yer alabilirler. WGS' yi oluşturan bilgisayarlar web servisleri ile haberleşebildikleri sürece görevlerini yerine getirirler.

Geliştirilen WGS modelinde uç bilgisayarlardaki *AnlıkYük* iş yükleri *EnÇokYük* eşik değerinden büyük ise fazla işler Emanetçi'ye gönderilmektedir. WGS modelinde uç bilgisayarın *AnlıkYük* iş yükü *EnAzYük* eşik değerinden küçük olduğunda uç bilgisayarlar Emanetçi'den iş talebinde bulunmaktadır. WGS modelinde hataya duyarlı tek Emanetçi kullanılmıştır. Birden fazla Emanetçi kullanılarak WGS sistem modeli genişletilebilir. Geliştirilen WGS modelinden birden çok yapıyı aynı anda kullanarak daha büyük sistemler modellenebilir. Ayrıca uç bilgisayarlar kendilerinin bağlı oldukları Emanetçi'ler yerine istedikleri Emanetçi'den iş talebinde bulunabilir. İş talebini alan uç bilgisayarlar yüklerini Emanetçi'ye kaydettirirken isterlerse difüzyon yolu ile işleri yakın komşuluklarına da transfer edebilirler böylece Emanetçi'ler arasında yük dağılımı da dağıtık olarak çözülmüş olur. Geliştirilen modelin bir işletim sistemi eklentisi haline getirilmesi durumunda kullanımı kolaylaşacaktır. Bu yöndeki çalışmalar halen devam etmektedir.

BÖLÜM 6

KAYNAKLAR DİZİNİ

Adar, N., Canbek, S., Akçay, M., 2006, "Heterojen Kümeli Bilgisayarların Web Servislerle Modellenmesi", Bilgi Teknolojileri Kongresi IV – Akademik Bilişim 2006, Pamukkale Üniversitesi, Denizli.

Adar, N., Canbek, S., Seke, E., Akcay, M., 2007, "Modeling a Web Service Based Decentralized Parallel Programming Environment", International Conference on Parallel Computational Fluid Dynamics ParCFD 2007, Antalya, Turkey, May 21-24.

Akcay, M., Adar, N., 2005, "Beowulf Cluster Lab in an Academic Environment", 2nd International Conference on Electronics and Computer in Kyrgyzstan (IKECCO2005), Bishkek, Kyrgyzstan.

Akçay, M., Adar, N., Canbek, S., Seke, E, 2007, "Heterojen Kümeli Bilgisayarların Modellenmesi", Ulusal Teknik Eğitim, Mühendislik ve Eğitim Bilimleri Genç Araştırmacılar Sempozyumu UMES 2007, Kocaeli Üniversitesi Teknik Eğitim Fakültesi, İzmit.

Akçay, M., Adar, N., Seke, E., Canbek, S., 2007, "Kümeli Hesaplama Modelinin Değişik Parametrelerle İncelenmesi", 12. Elektrik, Elektronik, Bilgisayar, Biyomedikal Mühendisliği Ulusal Kongresi, Eskişehir.

Akçay, M., Adar, N., "Web Servisler ile Dağıtık / Grid Sistem Mimarisi ", 2007, "III. İLETİŞİM TEKNOLOJİLERİ ULUSAL SEMPOZYUMU", Adana.

Bray, T., Paoli, J. and Sperberg-McQueen, C.M., 1998, The Extensible Markup Language (XML) 1.0.

Brittenham, P. An Overview of the Web Services Inspection Language.2001, http://www.ibm.com/ developerworks/webservices/library/ws-wsilover

Buyya, R., 2002, Economic-based Distributed Resource Management and Scheduling for Grid Computing, Pd.D. Thesis, Monash University, Australia.

Buyya, R., Abramson, D., Venugopal, S., 2005, "The Grid Economy", Proceeding of the IEEE, Vol. 93, Issue 3, pp.698-714

Chamberlin, D., 2001, Xquery 1.0: An XML Query Language. W3C Working Draft 07

Chien, A., Calder, B., Elbert, S., and, Bhatia, K., 2003, Entropia: Architecture and Performance of an Enterprise Desktop Grid System, Journal of Parallel and Distributed Computing, vol 63, issue 5, Academic Pres, USA.

Christensen, E., Curbera, F., Meredith, G. and Weerawarana., S. 2001, Web Services Description Language (WSDL) 1.1. W3C, Note 15, 2001, http://www.w3.org/TR/wsdl.

Cottet, F., Delacroix, J., Kaiser, C., Mammeri, Z., 2002, Scheduling in Real-Time Systems, John Wiley and Sons Ltd., England.

Cunha, J. C., and Rana, O. F., 2006, Grid Computing: Software Environments and Tools, Springer-Verlag London Ltd.

Foster I., Kesselman C., Nick J., Tuecke S., 2002, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure WG, Global Grid Forum.
http://www.globus.org/research/papers/ogsa.pdf

Globus Toolkits. http://www.globus.org

Halvorson, M., 2001, Microsoft Visual Basic 6.0 Professional, Arkadaş Yayınevi, 2. Baskı, Ankara (Çeviri: Selim Göksu)

Heiss, H.U. , Schmitz, M.. 1995, Decentralized dynamic load balancing: the particles approach. Information Sciences, vol 84, Issue: 1-2, pp. 115-128

Huang, Y., 2003, JISGA: A Jini-Based Service-Oriented Grid Architecture, International Journal of High Performance Computing Applications, Vol. 17, No. 3, pp. 317-327

Humphrey, M., Wasson, G., Jackson, K., Boverhof J., Rodriguez M., Bester J., Gawor J., Lang S., Foster I., Meder S., Pickles S., and McKeown M., 2005a, State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations, 4th IEEE International Symposium on High Performance Distributed Computing (HPDC-14), Research Triangle Park, NC, 24-27 July 2005.

Humphrey, M., and Wasson, G., 2005b, Architectural Foundations of WSRF.NET, International Journal of Web Services Research, Vol. 2, No. 3, pp. 83-97

Ibm http://www-5.ibm.com/tr/solutions/edu/grid.html

Krauter K., Buyya R. and Maheswaran M., 2002, A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing, Software-Practice and Experience, 32, pp.135–164.

Li, M., and Baker, M., 2005, The Grid Core Technologies, John Wiley and Sons Ltd., England

Limaye, K., Leangsuksun, B., Greenwood, Z., Scott, S. L., Engelmann, C., Libby, R., and Chanchio, K., 2005, Job-Site Level Fault Tolerance for Cluster and Grid environments, IEEE International Conference on Cluster Computing (Cluster 2005), Boston, Massachusetts, USA

LSF http://www.platform.com

Luther, A., Buyya, R., Ranjan, R., and Venugopal, S., 2005, Alchemi: A .NET-based Enterprise Grid Computing System, International Conference on Internet Computing, ICOMP 2005, Las Vegas, Nevada, USA, pp. 269-278

Mpi, Message Passing Interface (MPI) Standard, http://www-unix.mcs.anl.gov/mpi/

Marinescu, D.C., 2002, Internet-Based Workflow Management: Toward a Semantic Web, John Wiley and Sons Inc. New York, USA

Mutka, M., and Linvy, M., 1991, The Available Capacity of a Privately Owned Workstation Environment, Journal of Perfomance Evaluation, Volume 12, issue 4, pp 269-284, Elsevier Science Publishers, The Netherlands.

Open Grid Service Architecture (OGSA) Working Group, http://forge.gridforum.org/projects/ogsa-wg

Pvm, Parallel Virtual Machine (PVM), http://www.csm.ornl.gov/pvm/

Microsoft Corp., Web Services Development Center, http://msdn.microsoft.com/webservices/

Travostino, F., Mambretti, J., and Karmous-Edwards, G., 2006, Grid Networks: Enabling Grids with Advanced Communication Technology, John Wiley and Sons Ltd., England.

Trivede, S. K., Vasireddy, R., Trindade, D., Nathan, S., and Castro, R., 2006, Modeling High Availability Systems, 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06)

Wasson, G. and Humphrey, M., 2005, Exploiting WSRF and WSRF.NET for Remote Job Execution in Grid Environments, IPDPS, p. 12a, 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05).

Web1, 2004, "Web Service Architecture", http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

Web Servis, http://java.sun.com/blueprints/ guidelines/designing_webservices/html/

WSDL, Web Services Flow Language. http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

Yang, L. T., and Guo, M., 2006, High Performance Computing: Paradigm and Infrastructure, John Wiley and Sons Inc., Hoboken, NJ, USA

BÖLÜM 7

ÖZGEÇMİŞ

Türkiye Cumhuriyeti vatandaşı olan yazar, Anadolu Üniversitesi, Mühendislik Mimarlık Fakültesi, Elektrik Elektronik Mühendisliği Bölümü'nde lisans ve yüksek lisans eğitimini tamamlamıştır. 2008 yılında ise, Eskişehir Osmangazi Üniversitesi, Mühendislik Mimarlık Fakültesi, Elektrik Elektronik Mühendisliği Anabilim Elektronik Bilim Dalında doktora eğitimini tamamlamıştır. Paralel ve dağıtık sistemler, Grid hesaplama, yük dengeleme konularında aktif olarak çalışmaları devam etmektedir