T.C
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GRİD HESAPLAMA SİSTEMLERİ VE UYGULAMALARI

Niyazi ETİK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR ANABİLİM DALI

Bu tez 17.07.2008 tarihinde aşağıdaki jüri tarafından oybirliği/oyçokluğu ile kabul edilmiştir.

Prof.Dr. Ahmet ARSLAN (Danışman)	Prof.Dr. Şirzat KAHRAMANLI (Üye)	Yrd.Doç.Dr. Ahmet BABALIK (Üye)

ÖZET

Yüksek Lisans Tezi

GRİD HESAPLAMA SİSTEMLERİ VE UYGULAMALARI

Niyazi ETİK

Selçuk Üniversitesi Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Ahmet ARSLAN

2008, 70 Sayfa

Jüri:

Prof. Dr. Ahmet ARSLAN

Prof. Dr. Şirzat KAHRAMANLI

Yrd. Doç. Dr. Ahmet BABALIK

Çok sayıda matematiksel işlem yapılması gereken durumlarda tercihimiz genellikle yüksek işlem kapasitesine sahip bilgisayarlar olmuştur. Bilgisayarların işlem kapasitesindeki artış ise fiyat/performans oranını parabolik biçimde yükseltmektedir. Yüksek işlem gücüne sahip fakat pahalı bilgisayarlar yerine, nispeten daha yavaş birçok bilgisayara paralel olarak hesaplama yaptırmak, zaman ve maliyet açısından çok daha verimli olmaktadır. Elbette bu işlem yalnızca paralelleştirilebilen algoritmalar için geçerlidir. Grid Hesaplama Sistemleri "Süper Bilgisayar" olarak da adlandırılan, çok sayıda bilgisayarın bir araya gelerek şekillendirdiği paralel ve dağıtık hesaplama sistemleridir.

Bu çalışmamızda grid hesaplama sistemleri incelenerek, web servisleri yardımıyla bir grid hesaplama sistemi uygulaması geliştirilmiştir. Uygulamada belirli noktalar arasındaki en kısa yolu bulma problemi web servisleri kullanılarak paralel bir şekilde çözülmüş ve sonuçlar değerlendirilmiştir.

Anahtar Kelimeler: Grid, Grid Hesaplama, Süper Bilgisayar, Yüksek Hesaplama Sistemleri, Paralel Programlama

ABSTRACT

Master Thesis

GRID COMPUTING SYSTEMS AND APPLICATIONS

Niyazi ETİK

Selçuk Universty

Graduate School of Natural and Applied Sciences

Department of Computer Engineering

Supervisor: Prof. Dr. Ahmet ARSLAN

2008, 70 Pages

Jury:

Prof. Dr. Ahmet ARSLAN

Prof. Dr. Şirzat KAHRAMANLI

Asist. Prof. Dr. Ahmet BABALIK

In general, when we need to process a lot of complex procedures, we prefer computers which are well designed with high technology. However, improving the technology of computers increases the cost and the performance as a parabolic graphic. Whereas, using parallel processing with low technology computers is preferable, due to the processing time and cost. Of course, this is only acceptable for the algorithm that is suitable for parallel processing. Grid Computing Systems which are called "Super Computer" are parallel and distributed computing systems, that are formed a lot of computers.

In this study, grid computing systems have been examined and an application of grid computing systems is developed with web services. In this application, the problem of finding the shortest path between certain dots is solved by parallel processing with web services and the results are then assessed.

Keywords: Grid, Grid Computing , Super Computer, High Performance Computing, Parallel Programming

ÖNSÖZ

Çalışmalarım sırasındaki yönlendirme ve desteklerinden dolayı sayın hocam Prof. Dr. Ahmet ARSLAN ’a, maddî ve manevî yardımlarını esirgemeyen çok sevdiğim eşime ve aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

ÖZET i
ABSTRACT ii
ÖNSÖZ iii
İÇİNDEKİLER iv
ŞEKİL LİSTESİ vi
TABLO LİSTESİ vii
KISALTMALAR VE SİMGELER viii
1. GİRİŞ 1
2. KAYNAK ARAŞTIRMASI 2
3. MATERYAL VE METOD 7
3.1. GRİD HESAPLAMA 7
3.1.1. Grid Hesaplama Nedir? 7
3.1.2. Grid Hesaplama Sistemleri Çeşitleri 8
3.1.3. Grid Hesaplama Sistemlerinin Yapısı 9
3.1.4. Grid Hesaplama Sistemlerinin Bileşenleri 13
3.1.5. Grid Hesaplama Sistemi Çizelgeleri 16
3.1.6. Grid Hesaplama Sistemlerinin Yazılımlarında Dikkat Edilmesi Gereken Hususlar 17
3.1.7. Grid Hesaplama Sisteminin Kurulumu 19
3.1.8. Grid Hesaplama Sistemlerinde Güvenlik 20
3.1.9. Grid Hesaplama Uygulamaları 21
3.1.10. Grid Hesaplamada Kullanılan Araçlar 22
SmartFrog (Smart Framework for Object Groups) 23
Global Grid Form (GGF) 23
3.1.11. Grid Hesaplamada Performans Hesabı 23
3.2. SERİ HESAPLAMA 24
3.3. PARALEL HESAPLAMA 25
3.3.1. Neden Paralel Hesaplama? 26
3.3.2. FLYNN Klasik Sınıflandırması 27
3.3.3. Paralel Bilgisayar Hafıza Modelleri 29
3.4 PARALEL PROGRAMLAMA MODELLERİ 33
3.4.1. Paylaşılan Hafıza Paralel Programlama Modeli 34

3.4.2. Thread Paralel Programlama Modeli 34
3.4.3. Mesaj Aktarmalı Paralel Programlama Modeli 37
3.4.4. Data Paralel Paralel Programlama Modeli 37
3.4.5. Simple Program Multiple Data (SPMD) Paralel Programlama Modeli .. 38
3.4.6. Multiple Program Multiple Data (MPMD) Paralel Programlama Modeli 39
3.4.7. Hibrit Paralel Programlama Modeli 39
3.5 PARALEL PROGRAM TASARIMI 40
3.5.1. Programın Anlaşılması 41
3.5.2. Parçalara Ayırma 42
3.5.3. Haberleşme 45
3.5.4. Senkronizasyon 48
3.5.5. Veri Bağlılığı 49
3.5.6. Yük Dengeleme (Load Balancing) 50
3.5.7. Parçacıklılık (Granularity) 52
3.5.8. Giriş-Çıkış (I/O) 53
3.6. PARALEL PROGRAMLAMANIN LİMİTLERİ VE GİDERLERİ 54
3.6.1. Amdahl Kuralı: 54
3.6.2. Karmaşıklık: 56
3.6.3. Taşınabilirlik: 56
3.6.4. Kaynak İhtiyaçları: 56
3.6.5. Ölçeklenebilirlik: 57
3.6.6. Performans Analizi ve Ayarları: 58
3.7 WEB ÇİFTLİĞİ 58
3.8. WEB SERVİSİ 58
3.8.1 Web Servisi Nedir? 58
3.8.2. Web Servislerinde Kullanılan Standartlar 59
4. UYGULAMA 60
4.1. Örnek Problem 60
4.2. Mevcut Seri Çözüm 60
4.3. Mevcut Seri Çözümün Paralelleştirilmesi 60
4.4. Uygulama Programının İşleyişi 61
4.5. Uygulama Programının Sonuçları ve Değerlendirme 63
5. SONUÇ VE ÖNERİLER 66
6. KAYNAKLAR 68

ŞEKİL LİSTESİ

Şekli 3.1 Grid Çeşitleri…… 11
Şekil 3.2 Grid Yapısı…….. 12
Şekil 3.3 Grid Hesaplama Katmanları……. 13
Şekil 3.4 Grid Hesaplama Sistem Mimarisi……. 13
Şekil 3.5 Grid Hesaplamada İş Paylaşımı…… 14
Şekil 3.6 Grid Hesaplama Yönetim Katmanları……. 15
Şekil 3.7 Globus Toolkit…….. 24
Şekil 3.8 Seri Hesaplama…… 26
Şekil 3.9 Paralel Hesaplama…… 27
Şekil 3.10 Paylaşılan Hafıza Modeli…… 31
Şekil 3.11 Dağıtılmış Hafıza Modeli…… 33
Şekil 3.12 Çok İşlemcili Dağıtılmış Hafıza Modeli……... 34
Şekil 3.13 Thread Paralel Programlama Modeli…….. 37
Şekil 3.14 Data Paralel Programlama Modeli……. 40
Şekil 3.15 SPMD Paralel Programlama Modeli…….. 41
Şekil 3.16 MPMD Paralel Programlama Modeli……. 41
Şekil 3.17 Domain Decomposition Parçalama……. 45
Şekil 3.18 Dizi/Matris/Küp Parçalama……. 45
Şekil 3.19 Fonksiyonel Parçalama…….. 46
Şekil 3.20 Ses Filtreleme Örneği……. 46
Şekil 3.21 Haberleşme Bölgeleri……. 49
Şekil 3.22 Karmaşıklık…… 50
Şekil 3.23 Görev-Zaman Dağılımı…… 52
Şekil 4.1 Tek Bilgisayarda 11 Nokta İçin En Kısa Yol Hesaplaması….. 64
Şekil 4.2 7 Bilgisayarda 11 Nokta İçin En Kısa Yol Hesaplaması…….. 64
Şekil 4.3 Nokta Sayısına Göre Alternatif Yol Sayısındaki Artış……. 66
Şekil 4.4 Nokta Sayısına Göre İşlem Süresindeki Artış…….. 67

TABLO LİSTESİ

Tablo 3.1 Akım Performans İnceleme Tablosu…….. 26
Tablo 4.1 Nokta Sayısına Göre Alternatif Yol Sayıları…….. 65
Tablo 4.2 Nokta Sayısına Göre İşlem Süreleri…… 66

KISALTMALAR VE SİMGELER

AFS	Andrew Dosya Sistemi
API	Yazılım Programlama Arayüzü
BT	Bilişim Teknolojisi
CC	Hafıza Tutarlılığı Kontrol Sistemi
CC-NUMA	Hafıza Tutarlılığı Kontrollü Düzensiz Bellek Erişimi
CC-UMA	Hafıza Tutarlılığı Kontrollü Düzenli Bellek Erişimi
CPU	Mikroişlemci
DFS	Dağıtılmış Dosya Sistemi
DM	"Dağıtılmış Hafıza" Paralel Programlama Modeli
DNS	Alan Adı Sunucusu
GGF	Global Grid Formu
http	Hipermetin Aktarma İletişim Kuralı
IP	Internet Protokolü Ağ Adresi
MIMD	"Çok Program Çok Veri" Paralel Programlama Modeli
MISD	"Çok Program Tek Veri" Paralel Programlama Modeli
MPI	Mesaj Aktarma Arayüz Programı
N	Hesaplamaya Giren Nokta Adedi
NFS	Ağ Dosya Sistemi
NUMA	Düzensiz Bellek Erişim Modeli
RMI	Java Uzaktan Metot Çağırma Teknolojisi
RPC	Uzaktan Prosedür Çağırma İşlemi
SIMD	"Tek Program Çok Veri" Paralel Programlama Modeli
SISD	"Tek Program Tek Veri" Paralel Programlama Modeli
SM	"Paylaşılan Hafıza" Paralel Programlama Modeli
SMP	Simetrik Çokişlemcili Bilgisayar Sistemi
SOAP	Basit Nesne Erişim Protokolü
UDDI	Web Servislerinin Tanıtılması ve Bulunması İçin İletişim Kuralı
UMA	Düzenli Bellek Erişim Modeli
WSDL	Web Servisi Tanımlama Dili
WSMF	Web Servisi Yönetim İletişim Kuralı
WSRF	Web Servisi Kaynaklarına Ulaşmak için İletişim Kuralı
XML	Genişletilebilir İşaretleme Dili

1. GİRİŞ

Yirminci Yüzyılın en büyük buluşu sayılan bilgisayar ve internet ile insanoğlu tarihinde görülmedik bir teknolojik gelişme ivmesine ulaşmıştır. Yakın bir geçmişte, çözümüne neredeyse imkânsız güzüyle bakılan matematik işlemleri ve algoritmaları günümüz ev bilgisayarları bile kolaylıkla ve kısa sürede çözülebilmektedir. Yalnız matematik ve mühendislik alanında değil, başta tıp, biyoloji, kimya, ekonomi, güzel sanatlar ve endüstri olmak üzere neredeyse her alanda bilgisayar bilimin, üretimin ve hayatın bir parçası olmuş durumdadır. Yapay zekânın kişisel bilgisayarlara kadar yayıldığı düşünülürse artık bilgisayarla paylaşılan dünya bilim-kurgu filmlerinden dışarı taştığını kabul etmemiz gerekmektedir.

Bu akıl almaz hızdaki teknolojik gelişmeler insanoğlunun bilim iştahını kamçılamakta, daha yeni ve daha karmaşık problemleri bilgisayar yardımıyla çözme isteği doğurmaktadır. İnsanoğlunun hayal gücü ve teknolojiden beklentisi ise teknolojiden çok daha hızlı gelişmektedir. Bu da ne kadar hızlı gelişirse gelişsin günümüzün en hızlı ve en yüksek kapasiteli bilgisayarının bile insanoğlunun isteklerini karşılamada aciz kalmasına sebep olmaktadır. İşte Izgara Hesaplama (Grid Calculating) adı verilen sistemler bu açığı kapatmak amacıyla ortaya atılmıştır ve henüz ideal anlamda bekleneni veremeseler de önemli bir ihtiyaca cevap vermişlerdir. Bu ihtiyaç günümüzde üretilmiş olan en hızlı bilgisayardan 8-10 kat daha hızlı işlem yapabilen bilgisayar sistemleridir.

Bu çalışmada XML web servislerini kullanarak bir grid hesaplama sistemi şekillendirilmiştir. Bu sistem üzerinde C# programlama dili ile “thread”, “data paralel” ve “dağıtılmış hafıza” paralel programlama modelleri kullanan hibrit bir yazılım hazırlanmıştır. Bu yazılım “n” adet nokta arasında “En Kısa Yolu Bulma” problemini çözmek için kullanılmıştır. Paralel programdan elde edilen sonuçlar ve işlem süreleri aynı algoritmanın seri hali ile karşılaştırılmıştır.

2. KAYNAK ARAŞTIRMASI

Grid hesaplama sistemleri ile ilgili çalışmalar özellikle CERN LHC projesinin ile hız kazanmıştır. Ağ güvenliği, ağ trafik akışı, yetkilendirme gibi konular yanında grid hesaplama konusunda en önemli unsur hiç şüphesiz paralel programlamadır. Bu alanda yapılan pek çok çalışmanın yanı sıra ülkemizde de önemli projeler yürütülmektedir.

Grid hesaplama sistemleri alanında ülkemizde en kapsamlı çalışma TÜBİTAK-ULAKBİM ve bazı üniversitelerin sahip oldukları yüksek başarımlı bilgisayar merkezlerinin birleştirilmesi ve Avrupa Birliği Çerçeve programları kapsamında yer alan grid projelerine katılım sağlanması amacıyla yürütülmektedir. Proje ULAKBİM koordinasyonunda TR-Grid Ulusal Grid Oluşumu (TR-Grid UGO) adı altında 2003 yılında başlatılmıştır.

Fidan (2004) "Eğitim Ontolojisi" isimli akademik dijital kütüphane uygulamasını geliştirmiştir. Sayısal ortamda yer alan akademik ve bilimsel kaynakları arama motoru teknolojisi ile sunan bir uygulama olan çalışma internetteki Türkçe bilgilere ulaşarak indekslemeyi, indeksleri sürekli yenileyerek güncel bilgi sunmak ve nitelikli Türkçe arama sonuçlarını kullanıcılara sunmayı amaçlamaktadır.

Aykanat ve Ark. (2004) Grid tabanlı bir diğer arama motoru olan SE4SEE'yi geliştirmişlerdir. Geleneksel arama motorlarından en önemli farkı grid altyapısında çalışabilen, ülkeye özel, kategorik bir arama motoru olmasıdır.

Şener, Şahin ve Ark (2007) tarafından GridEA uygulaması SEE-GRID2 projesi kapsamında geliştirilmiş yapay evrim projesidir. Çalışmada doğadaki işleyişiyle bilinen evrim sürecinin, değişik problemlerin bilgisayar ortamında

çözümünde metot olarak kullanılması olarak nitelendirilebilir. Bu metotta, bir problemin çözümü şu şekilde aranır:

- Problemin çözüm uzayındaki değişik çözümleri temsil edebilecek bir "genotip" tasarlanır.
- Buna göre, her bir çözüm, genotipteki belli bir dizeye karşılık gelir.
- Başlangıçta bir havuz dolusu genotip rastgele yaratılır.
- Sonra her genotipin temsil ettiği çözümün 'iyi'liği (fitness), problem üzerinde denenir.
- Bu denemelerin sonunda iyilik sıralamasında yukarılarda yer alan çözümler, mutasyon, çaprazlama (cross-over) yöntemleriyle değiştirilerek bir sonraki çözüm nesli yaratılır.
- Bu işlemler, yeterince iyi bir çözüm bulunana kadar tekrarlanır.

Gürsoy, Aykanat ve Ark. (2006) tarafından SEE-GRID2 Projesi kapsamında GPIP uygulaması geliştirilmektedir. Uygulama protein veri bankalarındaki proteinler arasındaki etkileşimleri tahmin eden bir algoritmanın geliştirilmesini kapsamaktadır. Algoritma protein arayüzlerini kullanarak çalışmaktadır. Proteinler bağlanma alanları yoluyla etkileşirler. Bu alanlar proteinleri bağlanması için gerekli olan kimyasal ve fiziksel özellikleri sağlarlar. Protein arayüzü burada iki tane bağlama alanıyla tespit edilen bir çifttir. Tahmin algoritması yapısal olarak bilinen bir küme protein arayüzü ile başlar ve verile bir protein bankasındaki her çift proteine bu arayüzü eşleyerek veritabanındaki bütün protein etkileşim çiftlerini keşfeder.

Algoritmanın içyapısında girdi proteinlerin potansiyel etkileşim yüzeylerini çıkarmak için protein kısımlarının göreceli yüzey erişilebilirliğini hesaplayan NACCESS programı kullanılmaktadır. Bu safhada çıkarılan yüzeyler, daha sonra protein arayüzlerinin iki adet olan bağlama alanlarıyla karşılaştırılır. Karşılaştırılma işlemi MULTIPROT adındaki yapısal hizalama programıyla gerçekleştirilir. MULTIPROT verilen iki protein arasındaki benzer geometrik özleri sıradan bağımsız biçimde bulur. MULTIPROT'un çıktısından bir benzerlik puanı hesaplanır ve belli bir alt sınırın üstündeki puanlar bir eşleşme olarak kabul edilir. Bu karşılaştırma

sonunda bir protein arayüzünün iki ayağının da veritabanına eşlenmesi durumu bir etkileşimi bulunduğu anlamını taşır.

Tahmin algoritmasının çalışması sonucu bulunan protein etkileşimleri daha sonra halka açık etkileşim veritabanlarında (BIND, DIP, PDB) doğrulanmak için aranmıştır ve bazıları doğrulanmıştır. 67 protein arayüzü ve 6170 hedef proteinle başlayan tahmin işlemi sonunda 62616 protein-protein etkileşimi keşfedilmiştir.

Şener (2003) 'in de çalışmalarında ve testlerinde yer aldığı SEE-GRID2 projesi kapsamındaki P-GRADE Portalı ve P-GRADE Geliştirme Ortamı ilk olarak Macaristan Bilimler Akademisi, Bilgisayar ve Otomasyon Araştırma Enstitüsü (MTA-SZTAKI) Paralel ve Dağıtık Sistemler Laboratuarı tarafından geliştirilmiştir. Daha sonra SEE-GRID, Coregrid ve EGEE projeleri kapsamında geliştirilmeye devam edilmiştir.

P-GRADE iş akışına dayalı bir grid portalıdır. P-GRADE portalı ile seri ya da paralel işlerde oluşan iş akışları tanımlanıp çalıştırılabilir, bu işlerin durumu portal üzerinde takip edilebilir. Üst düzey kullanıcı arayüzleri kullanıldığı için, alt seviyedeki ayrıntıları kullanıcının bilmesine gerek kalmaz. P-GRADE portalıyla tanımlanan iş akışları, farklı grid platformlarına kolaylıkla taşınabilmektedir. Birden fazla grid sistemine aynı anda ulaşılması sağlandığı için, karmaşık uygulamalar kolaylıkla çeşitli platformlara dağıtılabilir.

Özturan, Yılmazer ve Ark. (2003) tarafından, Kandilli Rasathanesinde toplanan sismik verilerin grid üzerinde erişim ve kullanıma açılmasını sağlamayı amaçlayan SDA uygulaması geliştirmektedir.

Kandilli Rasathanesi ve Deprem Araştırma Enstitüsü Türkiye'nin deprem konusunda araştırma yapmak, eğitim ve hizmet vermekle yükümlü tek organizasyonudur. Kandilli Rasathanesi ile yapılan ortak çalışmalar 3 TB büyüklüğündeki sismik veriye ulaşım imkanı yaratmaktadır. Sismik veri Türkiye genelinde yaklaşık 100 farklı istasyondan toplanmaktadır. Mevcut durumda bu

veriler SAC formatında dosyalar olarak saat başı Kandilli Web sunucusuna yüklenmektedir. Kandilli tarafından sağlanan sismik veriler pek çok ulusal araştırma merkezi tarafından kullanıldığı gibi uluslararası organizasyonlar tarafında da deprem araştırmaları için kullanılmaktadır.

Kandilli Web sunucusunda yer alan verilerin yansısı TR-Grid Depolama biriminde tutulacaktır. Grid büyük miktarda sismik verini depolanması ve hesaplama yapmak üzere erişimin sağlanması imkânı yaratacaktır.

Sismik veri setlerine zaman ve istasyon bilgileri dâhilinde erişim için gerekli geliştirmeler C++ iterasyonları ile sağlanacaktır. Sismik verinin grid üzerinde kullanılmasını göstermek amacı ile Kandilli tarafından kullanılan bir uygulama grid üzerine taşınacaktır.

Bingöl ve Ark. (2007) tarafından Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümü'nde EUMEDGrid projesi kapsamında HuM2S (İnsan Belleğinin Simülasyonla Modellenmesi) çalışması geliştirilmiştir. Çok sayıda birimin birbirleri ile etkileşmeleri karmaşık sistemlerin ortak özelliklerindendir. Karmaşık sistemleri incelemede etkin bir yaklaşımla bu birimler modellenmiş ve dinamikleri simüle edilmiştir. HuM2S uygulamasında birimler insanlardır. İnsanlar birbirleri ile etkileşirler. Etkileşimler arasında tavsiye etme, sosyal baskı gibi değişik yapılar düşünülebilinir. Bu etkileşmeler bireylerin belleklerinde değişime yol açar. Bu şekilde tanımlanan bir modelde çok fazla sayıda parametre olması beklenen bir durumdur. Her parametre değeri için sistemin davranışının simülasyon yardımı ile incelenmesi gerekecektir. Modelde olasılığa dayalı değerler vardır. Bu nedenle sistem aynı parametre değerleri ile birçok defa simüle edilip, sonuçların istatistiksel olarak incelenmesi gerekmektedir.

Blaise (2006) paralel programlama konusunda kapsamlı bir çalışma yapmış; paralel program modeller, paralel program hafıza mimarileri ve paralel program tasarımında dikkat edilecek hususlar, güvenlik ve yük dağılımı gibi konuları

ayrıntılarıyla ele almıştır. Ayrıca bazı paralelleştirilebilen problemlerin çözümüne de çalışmasında yer vermiştir.

Foster (2006), Designing and Building Parallel Programs çalışmasında paralel program yazımı tekniklerini ele almış; güvenlik, yük dağılımı ve performans değerlendirmesi gibi konuların üzerinde durmuştur.

3. MATERYAL VE METOD

3.1. GRİD HESAPLAMA

3.1.1. Grid Hesaplama Nedir?

Geçtiğimiz yüzyılın sonlarında meydana gelen olağanüstü hızdaki teknolojik gelişmeler sayesinde insanoğlu bugüne kadar hayal bile etmediği hesaplama işlerinde bilgisayardan faydalanabileceğini fark etmiştir. Buna hayal gücünü de ekleyince günümüz bilgisayarlarının işlem ve hesaplama gücü yetersiz kalmıştır. İşte tam bu çıkmazın içindeyken bazı üniversite ve araştırma grupları, bilgisayar ağlarındaki hız artışının da avantajını kullanarak pek çok bilgisayarın işlem güçlerini birleştirip tek bir süper bilgisayar gibi çalışmaları fikrini ortaya atmışlardır. Fiziksel hatta coğrafi anlamda farklı noktalarda bulunan işlemci ve verinin paylaşılmasını ve birlikte kullanılmasını sağlayan paralel ve dağıtık sistemler ortaya çıkmıştır.

Grid Hesaplama Sistemleri, coğrafik olarak dağıtılmış bağımsız kaynakların kullanılabilirlik, yetenek, performans, maliyet ve kullanıcıların servis kalitesine bağlı olarak çalışma anında dinamik olarak paylaşım ve seçimine imkan tanıyan bir tür paralel ve dağıtılmış sistemlerdir.

Bu teknoloji sayesinde bilgiişlem kaynakları daha verimli kullanılabilmekte, atıl kaynakların ihtiyaç duyan kurumlarla paylaşılmasına imkan sağlanmaktadır. Böylece yatırımların geri dönüşümü daha hızlı gerçekleşmektedir. Yine bu teknoloji sayesinde kişisel bilgisayar kullanıcılarının kendilerine ait

bilgisayarların atıl durumlarından başkalarının faydalanmasına izin vermelerine ve bunları belli kurallar dahilinde kiralamalarına imkan sağlanmaktadır. Bu kullanım ve kiralama işlemi seçilen kriterlere ve özelliklere göre belirlenebilmektedir.

Kişisel bilgisayarlar günlük mesai dışında, atıl bekletildikleri zamanlarda çeşitli yazılım ve algoritmalar yardımıyla bir araya getirilerek bir işlem kümesi oluşturulmaktadır. Böylece pek çok bilgisayarın işlem gücü birleşerek çok yüksek işlem gücüne ve hacmine sahip bir sanal bilgisayar sistemi gerçekleştirilir.

Önemli bir husus da yüksek maliyet getirmeksizin ortaya çıkartılan bu süper sanal bilgisayarın bakım ve onarım maliyetlerinin, geleneksel sistemlerin bakım ve onarım maliyetlerine göre ihmal edilebilecek kadar küçük olmasıdır. Ayrıca çalışma esnasında ortaya çıkabilecek bir problem sistemin aksamasına sebep olmayacak; yönetici programlar durumu fark edip problemsiz bir düğüme aynı işi yapması için görev vereceklerdir. Bu problemsiz düğüm coğrafi olarak da bağımsız bir noktada olabilmektedir. Diğer düğümler paralel ve bağımsız olarak çalıştığı için hatanın tesbit edilmesine kadar geçen sürede de diğer işlemler çalışmalarına devam edeceklerdir.

Kısaca bir Grid Hesaplama Sistemi, Bilişim Teknolojisi (BT) alanındaki neredeyse her şeyi birleştiren bir sistemlerdir diyebiliriz. Bilgisayarlar, bilgisayar kümeleri, depolama alanları, uygulamalar, veri dosyaları, veritabanları v.b. bilişim ürünleri grid elemanlarına örnek olarak verilebilir.

3.1.2. Grid Hesaplama Sistemleri Çeşitleri

Grid hesaplama sistemleri basit cluster (küme) yapılarıyla başlanan sürçte git gide tüm dünyaya yayılmış, global bir hal almıştır.

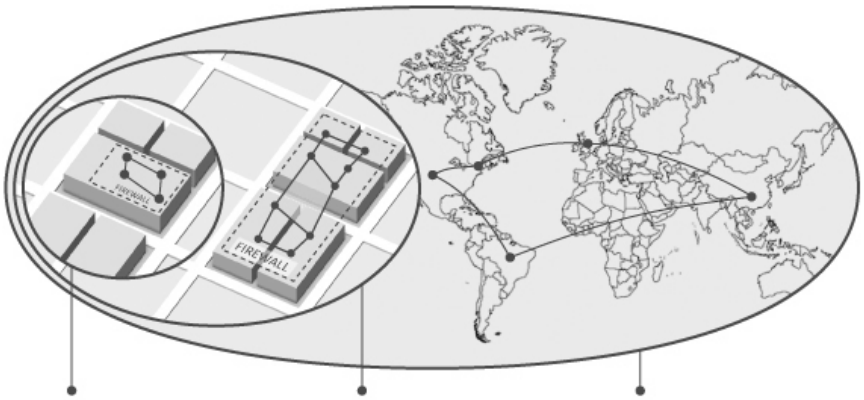
Yayıldıkları alana göre grid hesaplama sistemleri “Cluster Gridler”, “Enterprise Gridler” ve “Global Gridler” olmak üzere üçe ayrılırlar.

Cluster Grid: Planlanması ve yerleştirilmesi kolay olan grid türüdür. Doğrudan erişime kapalı ortamlarda gerçekleştirildiği için güvenlik protokollerine fazla ihtiyaç duyulmamaktadır. Bu sebeple ortam kaynaklarından maksimum faydalanılabilmektedir. Kaynaklar öncelik sırasına göre paylaştırılırlar.

Enterprise Grid: Bölüm veya kampüs genişliğinde gruplara imkan tanımaktadır. Alt ağ seviyesinde güvenlik genellikle yeterli olmaktadır.

Global Grid: Tüm dünya coğrafyasına dağılabilmektedir. Kaynaklar internet üzerinden paylaştırılır. İleri güvenlik politikaları ve protokollere ihtiyaç duyar.

Yayıldıkları alana göre grid çeşitleri şekil 3.1'de görülmektedir.

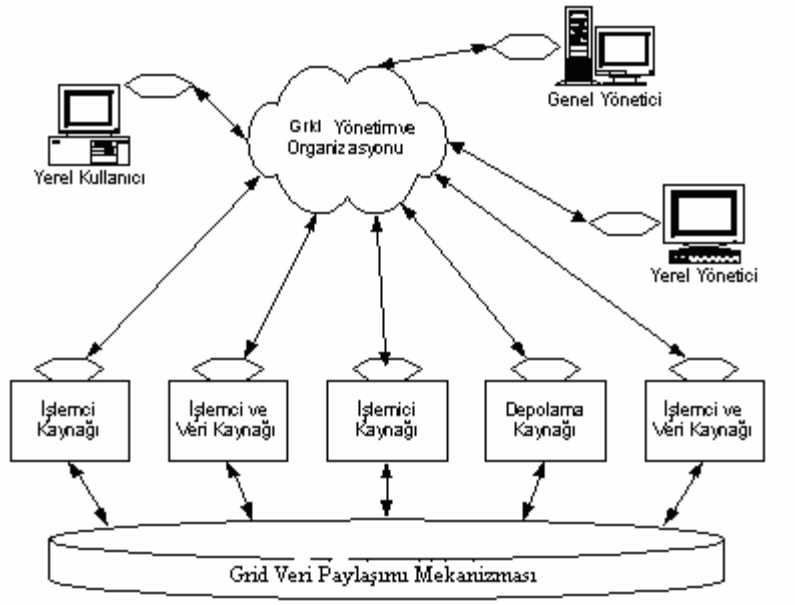


Şekli 3.1 Grid Çeşitleri, Hassaine, O.

3.1.3. Grid Hesaplama Sistemlerinin Yapısı

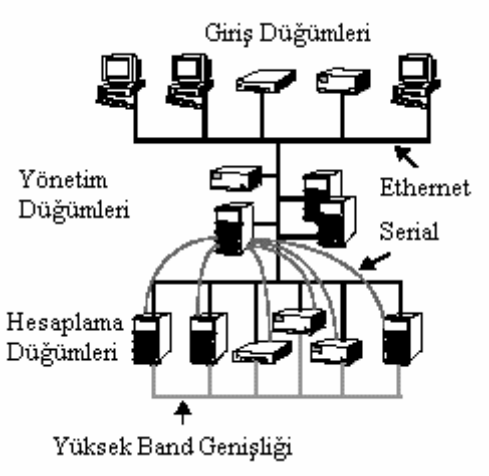
Grid hesaplama sistemlerinin mümkün olduğunca geniş ama güvenli bir alana yayılması amaç edinildiğinden bu sistemlerde pek çok güvenlik, zaman ve iş

paylaşım protokolü söz konusu olmaktadır. Bu da beraberinde genel ve yerel yönetici bilgisayar ve programları getirmektedir. Buna göre bir gridin şeması kabaca şekil 3.2'deki gibi olmaktadır.



Şekil 3.2 Grid Yapısı, Macit, İ.

Bir Grid Hesaplama Sistemi üç ana katmandan meydana gelmektedir. Bu katmanlar sırasıyla düğümlere (node) ve yazılımlara ulaşmayı sağlayan Giriş Katmanı (Access Tier), düğümlerin ve yazılımların yönetimini sağlayan Yönetim Katmanı (Management Tier) ve işlemlerin yürütülmesini, hesaplamaların yapılmasını sağlayan Hesaplama Katmanı (Compute Tier) 'dır.



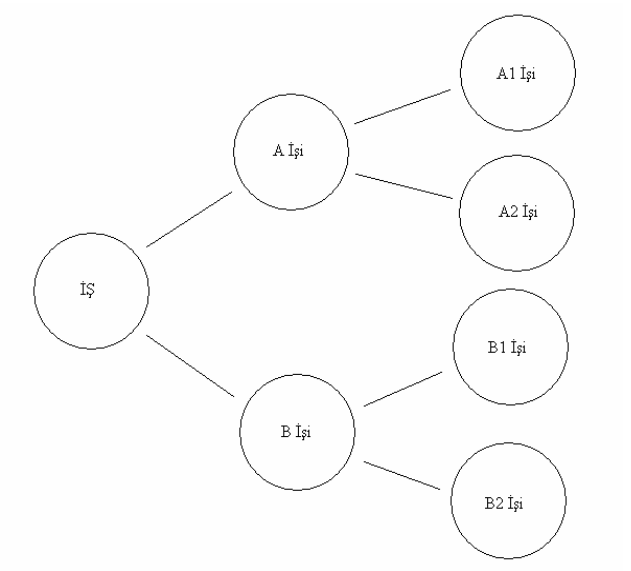
Şekil 3.3 Grid Hesaplama Katmanları, Sun Microsystems

Grid Hesaplamayı Sistem mimarisi açısından değerlendirmek gerekirse sistem seviyesinden kullanıcı seviyesine doğru altı ana katmandan bahsetmemiz gerekmektedir. Bu katmanlar şekil 3.4'te görülmektedir.



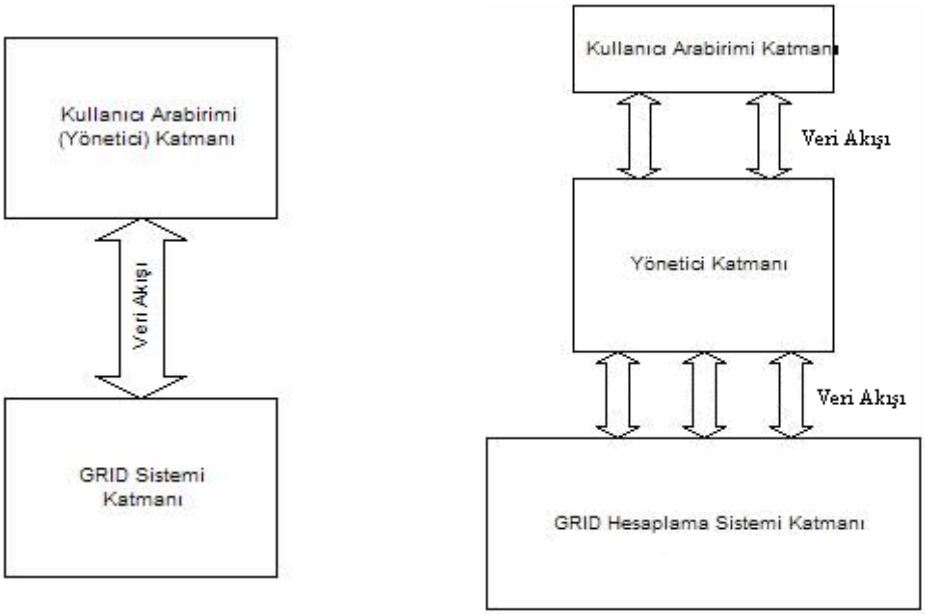
Şekil 3.4 Grid Hesaplama Sistem Mimarisi

Grid üzerinde işlem yapan her noktadaki bilgisayar kendine ait işi alt işlere bölebilir ve bu alt işleri gride dahil olan başka bir bilgisayar ya da noktaya gönderebilir. Bu durumda alt işlemlerin sonucu üst işlemin sonucuyla birleşerek anlam ifade edebilir veya üst işleme girdi olarak kullanılabilir. Bu yapı şekil 3.5'te şematik olarak gösterilmiştir.



Şekil 3.5 Grid Hesaplamada İş Paylaşımı

Grid Hesaplama sistemlerinde hangi bilginin veya işlemin nerede olduğunun kontrolü tamamen yönetici programların kontrolündedir. Yönetici programlar tek katmanlı olabilecekleri gibi çok katmanlı da tasarlanması mümkündür. Grid hesaplama yönetim katmanları şekil 3.6'da gösterilmiştir.



a) İki Katmanlı Grid Yapısı **b)** Üç Katmanlı Grid Yapısı

Şekil 3.6 Grid Hesaplama Yönetim Katmanları, Macit, İ.

Grid Hesaplama Sistemlerinin tasarımı sırasında ölçeklenebilir ve genişlemeye uygun olmasına özellikle dikkat edilmelidir. Aksi takdirde genişleme gerektiren durumlarda sistem mimarisinin tamamen yenilenmesi gerekebilir.

3.1.4. Grid Hesaplama Sistemlerinin Bileşenleri

Hesaplama: İşlemciler gerektiğinde hem grid hesaplama yapılacak olan işlemin, hem yönetim veya yönetime bilgi gönderen ajan programların, hem de yerel bilgisayarın işlemlerini yürütmek zorunda oldukları için anlaşmalara uygun olarak işlem öncelikleri belirlenir.

Uygulamalar: Grid hesaplama sistemlerinde üzerinde durulması gereken asıl konu sistemin amacını da teşkil eden uygulamalar veya diğer adıyla işlerdir. Çünkü sistem uygulamaların daha hızlı ve ucuza gerçekleştirilmesi için

tasarlanmaktadır. Yönetici programlar yardımıyla işler birçok düğüm üzerinde eşzamanlı olarak gerçekleştirilirler. İşler gerekli görüldüğü takdirde alt işlere bölünüp paralel olarak gerçekleştirildikten sonra sonuçlar birleştirilebilir veya bir işin sonucu başka bir işe girdi olarak kullanılabilir. Bu sırada düğümde veya haberleşmede bir problem olur da yeterince uzun süre düğümden sonuç döndürülmez ise yönetici işin tamamlanamadığını kabul edip başka bir düğüme aynı işi baştan yaptırtabilir.

Yazılım ve Lisans: Grid hesaplama sisteminin çalışması sırasında bilgiler ve işlemler yönetici programlar tarafından parçalara ayrılmakta ve ayrı düğümlerde işlenen parçalar daha sonra yine bu yazılımlar yardımıyla birleştirilip üst düğümlere gönderilmektedir. Bir düğümüm ürettiği değer başka bir düğüm için girdi değeri olabileceği için bu değerlerin dağıtılıp toplanmasındaki senkronizasyonu sağlayan bu programlara çok önemli ve hassas iş düşmektedir. Bu yazılımların grid üzerindeki her düğüme ayrı ayrı lisans alınarak yüklenmesi önemli bir maliyet yükü getirmektedir. Bu yük sınırlı kullanımlı program lisansları veya aynı gridde lisans aktarımı ile eşzamanlı kullanım adedi sınırlaması gibi özel anlaşmalarla asgariye indirilebilmektedir. Bir grid hesaplama sisteminin yazılımları temel olarak şu maddelerden meydana gelmektedir:

- **Yönetim Yazılımı***:* Yetkilerinin genişliğine göre genel ve yerel yönetim yazılımları olmak üzere ikiye ayrılırlar. Bu yazılımlar hangi kaynakların hangi kullanıcılar tarafından hangi yoğunlukla kullanıldığını ve verimliliğini takip ederler. Sistemi kullanmak isteyen kullanıcı ve yerel yöneticilerin yetkili olup olmadığının kontrolünü yapmak ve yetkili kullanıcılara hangi kaynakları kullanabileceğine dair bilgi vermek genel yönetici programın sorumluluğundadır. İş atamalarının yapılması, gerektiğinde işin o kaynaktan alınarak başka bir kaynağa verilmesi ve önceki kaynağın iş bekleme durumuna alınması, çizelgeleme ve çizelgelerin kontrolü, haberleşme sırasındaki veri güvenliği, farklı kaynaklardan üretilen işlem sonuçlarının birleştirilmesi, yeni kaynakların ve kullanıcıların sisteme eklenmesi

veya eskilerinin çıkartılması, işlerin kaynaklara dağıtılması ve sonuçların toplanıp değerlendirilmesi gibi işlemlerin tümü yönetici yazılımların görevlerindendir.

- **Alt İşlem Yazılımları***:* Grid hesaplama sistemlerinde bir iş alt işlere bölünüp başka düğümlere gönderilebilir ve alt işlemler paralel olarak işletilebilir. Bu durumda işi alt işlere bölme ve işlem sonuçlarının toplanıp birleştirilme işlemini yöneten yazılımlardır.

- **Tablolama Yazılımları***:* Kaynakların hangi işleri yapacaklarını, hangi kullanıcıların hangi kaynakları kullanabileceğini kontrol eden yazılımlardır. Bu yazılımlar sadece beklemedeki kaynağı kullanıcıya ve dolayısıyla işe tahsis edebileceği gibi işlem önceliği, kullanıcı veya işe göre uygun ya da yetkili kaynak ayarlanması yapabilirler. Ayrıca kaynağın işleme cevap süresi ve haberleşme trafiği de kaynak görevlendirmede etkili olabilir. İşlem sonucunun gecikmesi ve veri kaybı gibi durumlarda işlem aynı kaynakta yeniden başlatılabilir veya durdurulup başka bir kaynağa yönlendirilebilir. Sistem mimarisi ve anlaşmalara uygu olarak gerekirse işe göre kaynak rezervasyonu yapılabilir. Bu durumda kaynak yalnız belirlenen iş için kullanılacağından verimlilik artacaktır.

- **Haberleşme Yazılımları***:* İşlerin çok fazla alt işlere parçalanması durumunda hangi noktada hangi iş parçasının çalıştırıldığı, hangi işin hangi seviyede olduğu gibi bilgilere ulaşılmasını sağlayan yazılımdır.

- **Ölçme Yazılımları***:* Tablolama ve sistem yönetimi için gerekli performans bilgilerini ölçer ve yönetim ve tablolama yazılımlarına gönderir. Bu bilgiler kaynağın veya haberleşme sisteminin verimliliği gibi verimliliği etkileyen her türlü bilgi olabilir. Bu bilgiler tablolama yazılımı tarafından bir dahaki iş atamasında değerlendirilir.

Kullanıcılar: Sistemi kullanacak olan kullanıcıları belirlenmesi ve yetkilendirilmesi de önemli bir husustur. Kullanıcıların belirlenmesinden sonra yetkilendirme için genellikle sertifika vermeye yetkili kuruluşlardan alınan sertifikalar kullanılır. Kullanıcı sisteme girmeden önce bu sertifika yardımıyla kimliğini bildirir. Yönetim programı sertifika verme yetkisine sahip kuruluştan sertifikayı onaylatırsa kullanıcının sisteme girmesine izin verir.

Depolama*:* Grid hesaplama sistemlerinde veri depolamasını sağlayan araçlar da grid yapısındadırlar. Bu yapıya Data Grid adı verilir. AFS (Andrew File System), NFS (Network File System), DFS (Distributed File System) benzeri dosya sistemleri kullanılabilir. Ayrıca otomatik yedekleme ve RAID benzeri yapılar da kullanılmaktadır.

Haberleşme*:* Bilgisayar ağlarındaki teknolojilerin hızla gelişmesi sayesinde artık düşük sayılabilecek maliyetlerle yüksek hızda veri iletimi sağlanabilmektedir. Oluşabilecek darboğazları grid yönetim sistemi kontrol edip ortadan kaldırmaya çalışabileceği gibi, bu durumların önceden tespit edilip mimari yapının ona göre tasarlanması da çok önemli bir husustur.

3.1.5. Grid Hesaplama Sistemi Çizelgeleri

Grid Hesaplamada önemli görevlerden biri de çizelgeleme sistemine düşmektedir. Küçük çaplı gridlerde kullanıcı hangi işin hangi bilgisayarda yapılacağını kendisi seçebilir. Daha büyük gridlerde ise düğüm sayısının ve iş yoğunluğunun fazlalığı sebebiyle bu seçim işlemini çizelgeleme sistemi gerçekleştirir. Çizelgeleme hangi işin hangi düğümde gerçekleştirileceğini tablolayan sistemdir. Boşta bekleyen düğümler de bu tablo ile yönetici düğümler tarafından kontrol edilebilmekte ve gerektiğinde iş yönlendirmesi bu tabloya göre yapılmaktadır. İşi biten ve bekleme konumuna geçen kaynaklar da yine bu tabloda tutulur. Gelişmiş gridlerde belirli işler için kaynak rezervasyonu yapılabilir. Bu durumda belirli işlerin belirli kaynaklarda gerçekleştirileceği önceden belli olduğu

için işlem süresi azalacak ve servis kalitesi artacaktır. Öncelikli işlemler varsa iş önceliğine göre kaynak aktarımı sağlanmaktadır. Bekleyen işin durumuna göre uygun kaynak ataması yapılabilir. Ayrıca iş için atanan kaynak fazlasıyla uzun bir süre cevap vermezse sistem yöneticisi bu işin yapılmadığını kabul edip farklı bir kaynak ataması yapabilir ve eski kaynak bekleme durumuna geçirilebilir.

3.1.6. Grid Hesaplama Sistemlerinin Yazılımlarında Dikkat Edilmesi Gereken Hususlar

Grid hesaplama sistemleri yazılırken programcıların dikkat etmesi gereken bazı önemli hususlar şu şekilde sıralanabilir:

- Aynı tip olmak şartıyla farklı kaynakların birlikte kullanımı gerekmektedir. Birden fazla işlemci, disk v.b. kaynakların tek kaynakmış gibi veya paralel kullanımı sistemin temel fikrini meydana getirmektedir.

- Kaynakların lokal işlemlerini grid işlemleriyle paralel veya zaman paylaşımlı olarak yürütebilmeleri gerekmektedir.

- Lokal sistemlere (LAN, WAN v.b.) bağlantı ve güvenlik protokollerinin belirlenmesi gerekmektedir.

- Sistem ölçeklenebilir ve genişlemeye uygun olmalıdır. İleriki zamanlarda sisteme ciddi maddi yük getirmemesi için tasarım sırasında mimari yapıya büyük önem vermek gerekmektedir.

- Sisteme daha sonra eklenmesi muhtemel kaynaklar ve kullanıcılar için üyelik, bağlantı prosedürleri ve protokolleri belirlenmelidir.

- Yetkili kullanıcıların sisteme bağlandıkları anda yetki sertifika kontrolleri yapılmalı, kullanıcılar yalnız yetkilendirildikleri alanlarda işlem yapabilmelidirler.

- Güvenlik problemi klasik dağınık sistemlerinkinden daha fazla ön plana çıkmaktadır. Yalnız bilgi paylaşımı değil işlemci gücü de dahil olmak üzere her türlü kaynak paylaşımı yapılabildiği için virüsler ve truva atları göz önünde tutulması gereken çok ciddi tehlikelerdir.

- İşlemler alt işlemlere bölünebilmekte ve bu işlem parçaları farklı düğümlerde paralel olarak çalışabilmektedir. Bu yapı ciddi bir organizasyon tasarımını gerekli kılar. Böylece işlem süreleri azalacak ve servis kalitesi yükselecektir ki zaten bu da grid hesaplama sisteminin oluşturulmasındaki temel amaçtır.

- Büyük boyutlarda veri transferini gerektiren durumlar göz önünde bulundurulmalı, depolama düğümlerinin konumları veri transferi sırasında performansta önemli düşüş göstermeyecek şekilde tasarlanmalıdır.

- Haberleşme trafiğindeki kopukluk, donanım veya program arızası sonucu oluşabilecek kilitlenmeler, elektrik kesintisi veya iş önceliği v.b. durumlar yüzünden ortaya çıkabilecek kesinti ya da işlemin aşırı yavaşlaması gibi durumların kısa sürede belirlenmesi ve gerekli çözümlerin üretilmesi gerekmektedir.

- Önceki kaynak paylaşımlarının verimlerinin istatistiğini çıkararak sistem performansını en iyi seviyeye çıkaracak durumlar tespit edilmeli ve kaynak paylaşımı buna göre düzenlenmelidir.

- Mümkünse kaynak rezervasyonu yapılarak sistem performansı arttırılabilir.

- Belirli aralıklarla bilgilerin ve işlerin yedeği alınmalı, RAID benzeri yapılar kullanılarak hata toleransı arttırılmalıdır.

- Yetki sertifikalarının (public key) haricinde özel anahtarlar (private key) da oluşturulmalı ve bu anahtarlar çok güvenli bir şekilde korunmalıdır.

- Mümkünse yazılımların merkezi bir şekilde yüklenebileceği bir yapı oluşturulmalıdır.

3.1.7. Grid Hesaplama Sisteminin Kurulumu

Bir Grid Hesaplama Sistemi kurulurken sistem yönetimi ve kullanıcı seviyesinde yapılası gereken işlemler vardır. Sistem yönetimi açısından yapılması gereken işlemler şunlardır:

- Grid Yöneticisi (Grid Manager) belirlenir.
- Güvenlik Yöneticisi (Certificate Authority) belirlenir.
- Yeterli band genişliğine sahip bir ağ sistemi kurulmalıdır.
- Grid Klasör ve İndeks Servisleri tanımlanır.
- Kaynakları gridin parçası haline getirecek yazılımlar yüklenir.
- Grid istemcileri (client) belirlenir.
- Yerel ağ adresleri ve tanımlama bilgileri ayarlanır.
- Kullanıcı ve kaynakların gride eklenmesi için yöntemler belirlenir.

Kullanıcı açısından bakarsak yapılması gereken işlemler ise şu şekilde sıralanabilir:

- Kullanıcının kendi bilgisayarının grid hesaplama sisteminde kullanılacak yazılımları çalıştırabilecek özellik ve kapasitede olması gerekmektedir.

- Kullanıcının bağlı olduğu bir sistem varsa bu sistemin güvenlik ayarlarının yeniden gözden geçirilmesi gerekir.
- Kullanıcı, kimliğinin sistem tarafından doğrulanabilmesi için yetkili bir sertifika sağlayıcıdan güvenlik sertifikası almalıdır.
- Sistem yöneticisi tarafından kullanıcıya bir hesap açılır; kullanıcı adı ve şifresi verilir ve yetkileri belirlenir.

Bu ayarlamaların ardından grid hesaplama sistemine üye olan kullanıcı sisteme her bağlanışında güvenlik sertifikasını yönetici programa sunar. Kimliği onaylanan kullanıcı sistem yöneticisinin kendisine verdiği yetkiler dahilinde işlerini yürütebilir, kendisine gönderilen işleri yapar veya başka düğümlere iş gönderebilir. Kullanıcı grid hesaplama sisteminin komutlarını kullanarak başka düğümlere gönderdiği alt işlerin ne durumda olduğunu takip edebilir.

3.1.8. Grid Hesaplama Sistemlerinde Güvenlik

Grid hesaplama sistemleri sadece veri paylaşım için tasarlanmış sistemler olmayıp asıl amacının pek çok bilgisayar sisteminin kaynaklarını ve özellikle de işlemci gücünü paylaşmak olduğunu daha önce belirtilmişti. Bu durum güvenlik konusunu klasik veri paylaşma sistemlerinden çok daha fazla ön plana çıkarmaktadır. Virüsler, truva atları ve hacker saldırıları gibi ciddi problemler dikkate alınırsa güvenliğin grid hesaplamada ne kadar önemli olduğu daha kolay anlaşılır.

3.1.9. Grid Hesaplama Uygulamaları

CERN LHC (Large Hadron Collider) COMPUTING GRIDCERN European Organization for Nuclear Research:

İsviçre-Fransa sınırında, Avrupa'lı yirmi üye devlet ve Türkiye dahil pek çok üye olmayan devlet tarafından desteklenen dünyanın en büyük parçacık fiziği merkezidir. 3000'den fazla çalışanı ve binlerce ziyaretçi bilim adamına 50 yıldan fazla zamandır hizmet vermektedir. 1990 yılında www (world wide web) 'in doğduğu LHC fikrinin ortaya atıldığı merkezdir. Bu merkezdeki her bir deneyde saniyede 40 milyon işlem gerçekleşmekte, saniyede 10 ila 100 Megabayt arası bilgi saklanmakta, yılda yaklaşık 15 petabayt kayıt yapılmaktadır ve bu kayıtlar kullanıcılar tarafından yeniden düzenlenip analiz edilmektedir. Bu bilgilerin 700 MB kapasitesinde CD'ler üzerine kayıt edildiği ve CD'lerin üst üste konulduğu düşünülürse bir yıllık bilgi yaklaşık 20 km yüksekliğinde bir CD dağı meydana getirmektedir ki bu da bir Concord'un uçuş yüksekliğinden yüzde otuz daha fazladır. Sistemin işleyişi için çalışan gride 2004 yılında 22 ülke, 65 bölgede 6100 cpu ile işlem yapılmaktadır.

SPACI (Southern Partnership for Advanced Computational Infrastructure):

Güney İtalya'da üç üniversitenin (Lecce, Calabria, Naples) BT kaynaklarını birleştirmek amacıyla kurdukları griddir. 3M, HP ve yerel yönetimler tarafından da desteklenmektedir.

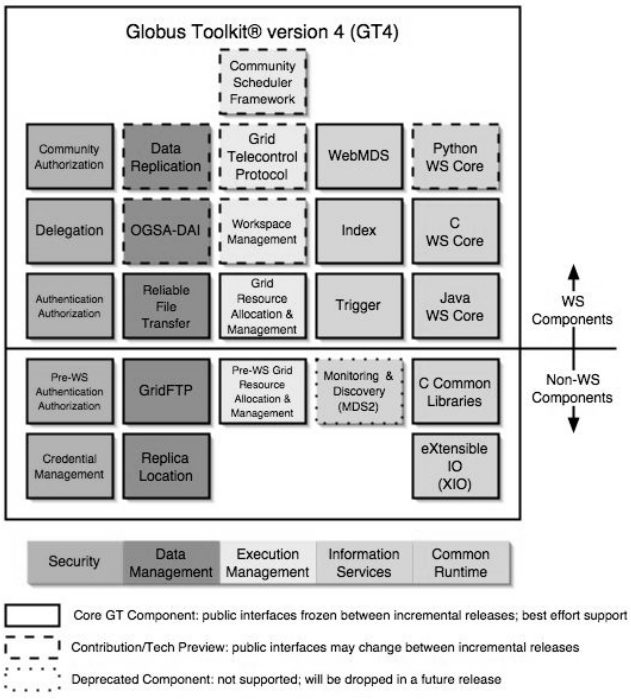
Uygulamadaki projeler:

- Hermes Project (Hematology Research Virtual Medical System)
- AirQuality Grid

- Atmospheric and Climate Modeling
- Computer Aided Medicine
- Biomedical Imaging Grid (Medigrid & BIG)
- Proteomics and Genomics Grid
- Diesel Engine Simulation Grid
- Digital Puglia

3.1.10. Grid Hesaplamada Kullanılan Araçlar

Globus Toolkit: Grid Hesaplama sistemlerindeki bilgisayarların işlem güçlerini, veritabanlarını ve diğer BT kaynaklarını güveli, eş zamanlı ve ortak olarak kullanılmasını sağlayan açık kaynak kodlu yazılımdır. Kaynakların bulunması, izlenmesi, yönetimi, ilave güvenlik sistemi ve dosya yönetimi gibi imkanlar sunmaktadır. Haberleşme için RSL (Globus Resource Specification Language) komutları kullanılmaktadır. GRAM (The Globus Resource Allocation Manager) API 'si sayesinde uzak kaynaklarda aynı yapıya sahip olmasa bile uygulama programları çalıştırılabilmektedir. Şekil 3.7 Globus Toolkit'in içerdiği öğeleri göstermektedir.



Şekil 3.7 Globus Toolkit, www.globus.org

PBS Professional: İş dizileri adı verilen birbiriyle ilişkili işlerin gruplanmasını sağlar. Kullanıcıların bir iş kümesine tek bir birimmiş gibi bilgi göndermesine ve almasına ayrıca işleri düzenlenmesine ve izlemesine imkan tanır.

SmartFrog (Smart Framework for Object Groups)*:* Tanımlama, bilgilerin paketlenmesi, peketlerin birleştirilip hazır hale getirilmesi gibi işlemleri kontrol eder.

Global Grid Form (GGF)*:* Microsoft, SUN Microsystems, IBM, HP, Cisco Systems, Oracle, Intel, Dell, e-Science, Voltaire, Cassata, Nortel gibi pek çok teknoloji devi tarafından desteklenen uluslar arası grid formudur. Grid Hesaplama alanında bilimsel ve endüstriyel uygulamalarda bilgi paylaşımı için tasarlanmıştır.

Web Services Management Framework (WSMF): BT kaynaklarının adreslerini yönetir. Web servislerini ve Grid servislerini içerir. Web servislerini kullanır.

3.1.11. Grid Hesaplamada Performans Hesabı

Bilgisayar sistemlerinde yüksek başarım elde edebilmek birçok parametreye bağlıdır. Kümelenmiş bilgisayar sistemlerinde ise durum ek parametreler ile birlikte daha hassas olmaktadır.

Kümelenmiş sistemlerde bilgisayarın kendi hızı yanı sıra diğer bilgisayarlar ile haberleşmesi de bu hassasiyeti arttırmaktadır.

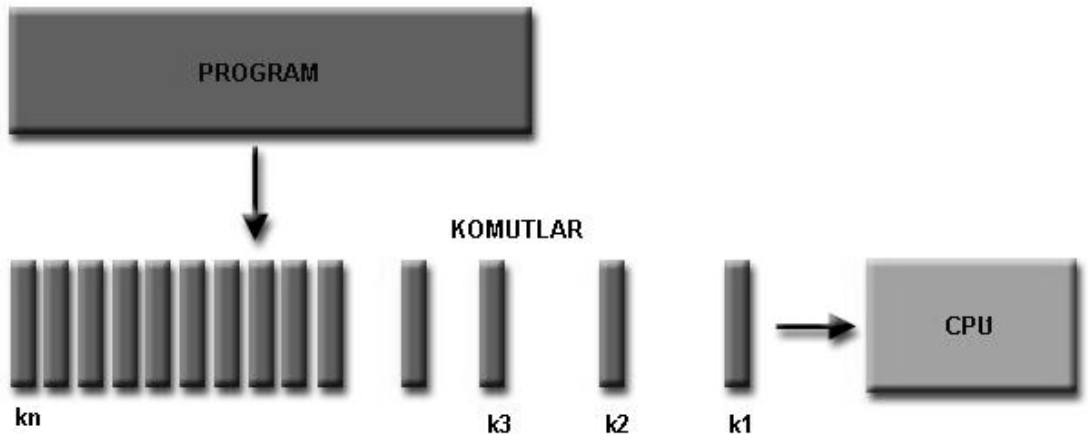
Gecikme olarak adlandırılan bu işlemler arasındaki geçen süre sistemin performansını direk olarak etkilemektedir. DeFanti ve Stevens performans değerlendirilmesinde dokuz akım tipini yedi ölçülebilir birim ile incelemişler ve bir tablo elde etmişlerdir. Bu performans ölçüm kriterleri tablo 3.1'de verilmiştir.

Tablo 3.1 Akım Performans İnceleme Tablosu, DeFanti ve Stevens

Akım Tipi	*Gecikme*	*Band Genişliği*	*Güvenirlilik*	*Çoklu Yayın*	*Güvenlik*	*Akım*	*Servis Kalitesi*
Kontrol	< 30	64 Kb/s	Var	Yok	Yüksek	Yok	Düşük
Yazı	< 100	64 Kb/s	Var	Yok	Orta	Yok	Düşük
Video	< 30	N x 128 Kb/s	Yok	Var	Orta	Var	Orta
Ses	< 100	N x 5 Mb/s	Yok	Var	Düşük	Var	Orta
İzleme	< 10	N x 128 Kb/s	Yok	Var	Düşük	Var	Orta
Veritabanı	< 100	> 1 GB/s	Var	Belki	Orta	Belki	Yüksek
Benzetim	< 30	> 1 GB/s	Karışık	Belki	Orta	Belki	Yüksek
Dokunuş	< 10	> 1 MB/s	Karışık	Belki	Yüksek	Var	Yüksek
Kaplama	< 30	> 1 GB/s	Yok	Belki	Düşük	Belki	Orta

3.2. SERİ HESAPLAMA

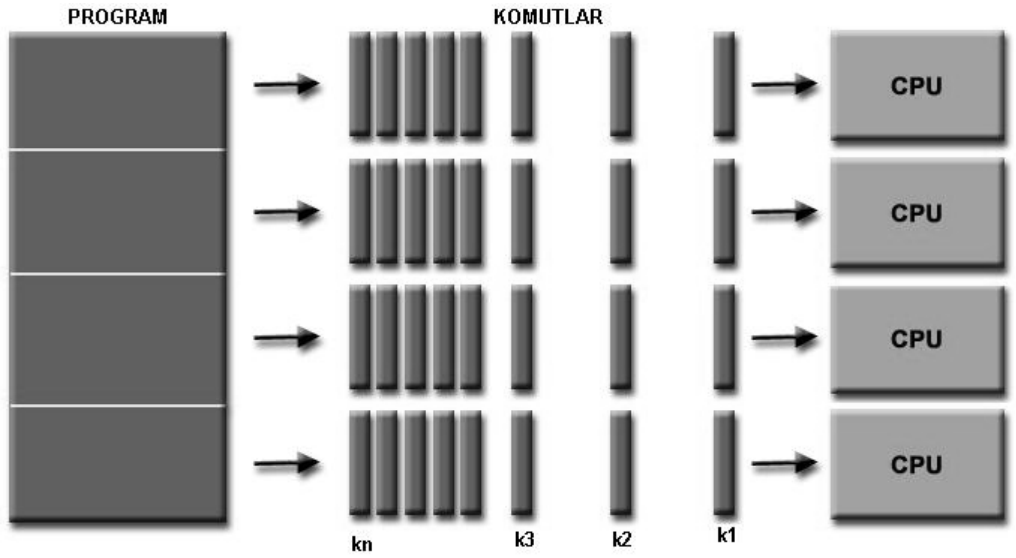
Bilgisayar programları genellikle seri hesaplama yapmak üzere tasarlanırlar. Yazılan programların tek bilgisayarda ve tek işlemcide çalışması durumunda bu doğru bir yaklaşımdır. Böylece ard arda işletilecek komutlar işlem sıralarına göre yazılırlar. İşlem sırası gelen komut mikroişlemciye (CPU) gönderilerek işletilir. Bunun ardından gelen komut ya da komutlar, bu komutun çalışmasından etkilenecek olsunlar veya olmasınlar bu komutun işlemesinin bitmesini beklemek zorundadırlar. Seri hesaplamada komutların yürütülmesi şekil 3.8'de sembolize edilmiştir.



Şekil 3.8 Seri Hesaplama

3.3. PARALEL HESAPLAMA

En sade haliyle paralel hesaplamayı bilgisayara ait bir çok kaynağın sayısal bir problemin çözümü amacıyla eşzamanlı olarak kullanılması olarak tanımlayabiliriz. Paralel hesaplamaya uygun bir algoritmaya sahip program öncelikle eşzamanlı çalışacak komut gruplarına ayrılır. Her komut gurubu eş zamanlı olarak, farklı CPU'lar üzerinde çalıştırılır. Gerekiyorsa bu bağımsız işlemlerin sonuçları daha sonra tekrar bir araya getirilebilir veya bağımsız olarak da anlam ifade ediyor olabilir. Paralel hesaplamanın işleyişi şekil 3.8'de şematik olarak gösterilmiştir.



Şekil 3.9 Paralel Hesaplama

Paralel hesaplama sırasında birden fazla işlemcili bir bilgisayar veya bir ağ üzerinden birbirlerine bağlanmış birden fazla bilgisayar kullanılabileceği gibi bu sistemlerin her ikisi de birleştirilerek daha yüksek işlem hacmine sahip bir sistem de gerçekleştirilebilir.

Paralel hesaplama sırasında bağımsız iş parçaları eşzamanlı olarak çalıştırılabilirler. Yani programımız bir taraftan kompleks bir denklemin kök değerlerini hesaplarken diğer tarafta farklı sayı kombinasyonları üretebilir, bir diğer taraftan da bir resimdeki görüntüyü diğer bir resimle karşılaştırabilir. Her bir işlem

için farklı işlemcilerin eşzamanlı kullanıldığı düşünülürse problemin çözümü için ciddi bir zaman kazancı gerçekleşecektir.

Paralel hesaplama aslında seri hesaplamanın geçirdiği çok önemli bir evrimdir. Gerçek hayat paralel işlemlerle yoğrulmuştur: Aynı anda çayımızı içerken televizyon seyredebilmemiz, müzik dinlerken bir başka iş yapabilmemiz gerçek hayatın aslında paralel işlemlerden meydana geldiğini göstermektedir. Elektronların farklı eksenlerde eş zamanlı olarak hareket etmeleri gibi pek çok gezegenin de hem kendi ekseni hem de yörüngesi üzerinde hareket etmeleri de bize mikrodan makroya tüm evrende de paralel işlem yapısının hâkim olduğunu göstermektedir.

Paralel hesaplama yüksek işlem hacmine sahip problemlerde oldukça iddialı olduğu için pek çok yüksek işlem kapasitesi gerektiren alanda kullanılmaktadır ve daha nicelerinde kullanılmaya adaydır. Meteorolojide hava tahminlerinde, kimyasal ve nükleer reaksiyonlarda, biyolojide canlıların genleri üzerinde, jeolojide sismik aktivitelerin incelenmesinde, elektronik devrelerinde, imalat işlemlerinde, uzay araştırmalarında ve daha pek çok bilimsel hesaplamada paralel hesaplama ile karşılaşmaktayız.

Günümüzde ticari yazılımlarda da daha yüksek işlem hacmine sahip sistemlere ihtiyaç duymaktayız. Web arama motorları, web tabanlı iş servisleri, bilgisayar destekli teşhis sistemleri, paralel veritabanları, veri madenciliği, ileri grafik işlemleri, sanal gerçeklik, sonlu elemanlar analizleri, multimedia, telekonferans ve görüntülü iletişim gibi pek çok alandaki gelişmeler paralel hesaplamanın yaygınlaşmasıyla çok büyük ivme kazanmıştır.

3.3.1. Neden Paralel Hesaplama?

Paralel hesaplamanın bize getirdiği en önemli avantaj hiç şüphesiz zaman kazancıdır. Bu kazanç o kadar ciddi boyutlardadır ki paralel hesaplama tekniği kullanılmadan önce yapılması tahmin bile edilemeyen hesaplamalar mesela bazı gen

analizleri paralel hesaplamayla birlikte mümkün olmakta hem de işlem hacminin büyüklüğüne göre makul sayılabilecek kadar kısa sürede gerçekleştirilebilmektedir. Aynı anda birden fazla işlemin yürütülüyor olması bilgisayarda gerçekleştirmek istediğimiz işlemlerin gerçek hayatla uyumluluğunu da sağlar. Çünkü gerçek hayat paralel işlemlerden meydana gelmektedir.

Paralel hesaplamanın en önemli avantajlarından biri de farklı bilgisayarlara ait kaynakların (CPU, RAM, Disk v.b.) eş zamanlı olarak aynı uygulama içerisinde kullanılabiliyor olmasıdır. Bu işlem için bilgisayarların bir ağ üzerinden haberleşebiliyor olmaları yeterlidir. Böylece çok yüksek maliyetlerle alınacak bir bilgisayar yerine daha düşük maliyete sahip pek çok bilgisayarı kullanarak aynı işi daha ucuza ve daha kısa sürede tamamlamamız mümkün olabilir. Ayrıca birden fazla bilgisayar arasında sadece CPU değil RAM, harddisk ve benzeri pek çok farklı kaynağı tek bir süper bilgisayarmış gibi ve eş zamanlı olarak kullanmak mümkündür.

Bakır telin iletme hızının limiti, mikroçip teknolojisinin çok küçük bir alana çok fazla miktarda transistor yerleştirme ihtiyacı ve ısınmamın önlenmesi gibi problemler yüksek hıza sahip bir işlemcinin maliyetini oldukça yükseltmektedir. Bu da bilgisayarımızın işlemcisini daha hızlı bir işlemciyle değiştirmek yerine, birden fazla işlemcinin veya birden fazla bilgisayarın ya da her ikisinin birleşiminin hem daha ucuz hem de daha hızlı işlem yapabileceğini göstermektedir. İnsanoğlu ucuz ve hızlı işlemciler üretse bile paralel hesaplama sistemleri her zaman bu üstünlüklerini koruyacaklardır. Çok çekirdekli işlemciler bu gerçeği oldukça iyi anlatmaktadır.

3.3.2. FLYNN Klasik Sınıflandırması

Paralel hesaplama sistemlerinin sınıflandırılmasında pek çok metot izlenmiştir. Bunlardan en yaygın olanlarından birisi de 1960'dan beri Flynn klasik sınıflandırmasıdır. Flynn sınıflandırması çok işlemcili bilgisayar mimarilerini komut ve veri olmak üzere 2 boyutta değerlendirir. Bu boyutlardan her biri "tek" veya "çok" durumlarından birine sahip olabilirler.

Buna göre dört farklı durum söz konusudur:

1. Tek Komut Tek Veri (Single Instruction Single Data - SISD): Seri bilgisayarlar olarak da düşünülebilir. Mikroişlemcide aynı anda yalnızca bir tek komut işletilir. Aynı şekilde aynı anda işlenen veri sayısı da en fazla bir olabilir. En eski sistemlerin tamamı bu şekilde işlem yürütmekteydi. Halen pek çok PC, tek işlemcili iş istasyonu ve mainframe bu sistemle çalışmaktadır.

2. Tek Komut Çok Veri (Single Instruction Multiple Data - SIMD): Gerçek bir paralel hesaplama sistemidir. Yürütülen her bir işlem (process) aynı saat sinyaliyle aynı komutu çalıştırmaktadır. Burada üzerinde durulması gereken husus komutların farklı veriler ile çalışmasıdır. Yani her bir paralel işlem aynı saat sinyaliyle aynı komutları farklı veriler için kullanmaktadır. Paralel genişliği yüksek networklerde ve düşük kapasiteli komut kümelerine sahip büyük dizilerde kullanılır. Görüntü işleme gibi yeterince düzenli sistemlerdeki iş yükü bu şekilde kolaylıkla dağıtılabilir. Eş zamanlı ve deterministik işlemlere imkân tanır. İşlemci Dizileri (CM1, MP1, MP2) ve Ardışık Vektörel (IBM 9000, Fujitsu VP) olmak üzere iki türü vardır.

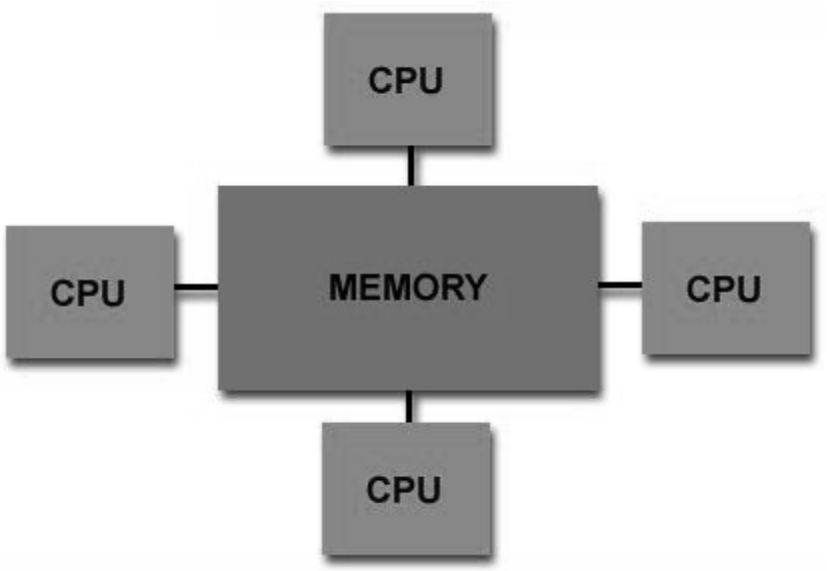
3. Çok Komut Tek Veri (Multiple Instruction Single Data - MISD): Tek bir data grubunun birden fazla işlem tarafından kullanılmasıyla meydana gelir.

4. Çok Komut Çok Veri (Multiple Instruction Multiple Data - MIMD): Paralel hesaplamanın en yaygın türüdür. Modern bilgisayarlar bu türe girerler. Her bir işlemci farklı bir komut setini çalıştırabilir. Her işlemci farklı bir data setiyle çalışabilmektedir. İşlemler senkron veya asenkron olabileceği gibi deterministik veya nandeterministik olabilirler. Süper bilgisayarlar, grid hesaplama sistemleri, çok işlemcili SMP bilgisayarlar ve bazı PC'ler bu türe girmektedir.

3.3.3. Paralel Bilgisayar Hafıza Modelleri

Paylaşılan Hafıza (Shared Memory) Modeli

Paylaşılan hafızalı paralel bilgisayarlar oldukça yaygın kullanılmaktadır. Bu tür hafıza kullanan işlemlerden her biri hafızanın global adres uzayının tamamını kullanabilme yeteneğine sahiptir.



Şekil 3.10 Paylaşılan Hafıza Modeli

İşlemciler bir taraftan bağımsız olarak işlemlerini yürütürken diğer taraftan aynı hafızayı paylaşabilmektedirler. Böylece bir işlemcinin değişiklik yaptığı hafıza bölgesi diğer işlemciler tarafından da görülebilmekte ve değiştirilebilmektedir.

Paylaşılan hafızalı bilgisayarlar hafızaya ulaşma zamanlarına göre iki sınıfa ayrılırlar:

- **Uniform Memory Access (UMA):** Genellikle eşzamanlı (symetric) çokişlemcili (multiprocessor) bilgisayarlar tarafından kullanılır. Sınırlı sayıda işlemcilerle kullanılabilir. İşlemciler aynı anda ve aynı

miktarda hafızaya ulaşabilirler. Bu tür hafıza paylaşımına CC-UMA (Cache Coherent UMA) adı da verilmektedir. Cache Coherent eğer bir işlemci, paylaşılan bir hafıza bölgesinin değerini değiştirirse tüm diğer işlemcilerin bu değişimden haberleri olur demektir. Bu haberleşme donanım seviyesinde sağlanmaktadır.

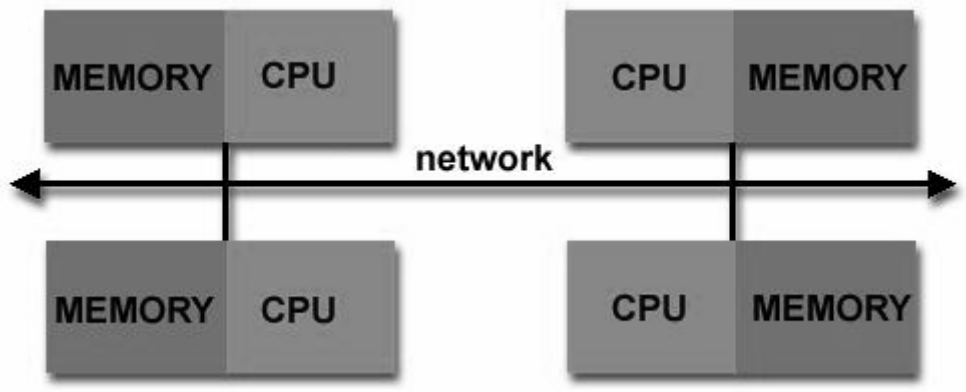
- **Non-Uniform Memory Access (NUMA):** Genellikle iki veya daha fazla eşzamanlı çokişlemcili (SMP / Symetric Multiprocessor) bilgisayarın fiziksel olarak bağlantısıyla gerçekleştirilmektedir. SMP bilgisayarlardan her biri diğerinin hafızasına doğrudan ulaşabilmektedir. Her işlemci her hafızaya eşit ulaşım süresine sahip değildir. Özellikle karşılıklı SMP bilgisayarların birbirlerinin hafızalarına ulaşması işlemi daha yavaş olmaktadır. CC bu hafıza paylaşım türünde de mevcuttur ve Cache Coherency NUMA (CC-NUMA) olarak adlandırılır.

Hafızada ortak adres uzayı kullanmak programcılık açısından büyük kolaylıklar sağlamaktadır. Ayrıca hafızanın mikroişlemci ile yakınlığı işlemciler arasında hız ve aynı oranda veri paylaşımına imkân tanımaktadır.

Tabii ki bir takım dezavantajları da vardır. Bunlardan ilk olarak hesaba katılması gereken hafıza ile mikroişlemci arasındaki ölçeklenebilirliktir. Mikroişlemci sayısındaki artış mikroişlemci ile hafıza arasındaki trafiği geometrik olarak arttırmaktadır. Cache Coherent sistemler için hafıza yönetimi de geometrik olarak artan trafikten etkilenmektedir. Programcı açısından da bir takım zorluklar ortaya çıkmaktadır. Ortak paylaşılan hafızaya ulaşma sırasında senkronizasyonun güvenlik altında tutulması tamamen programcıların sorumluluğuna bırakılmıştır. Bu problem mikroişlemci sayısının artması ile daha da karmaşık bir hal almaktadır.

Dağıtılmış (Distributed Memory) Hafıza Modeli

Paylaşılan hafıza sistemlerinde olduğu gibi dağıtılmış hafıza sistemleri de geniş bir kullanım alanına sahiptir. Dağıtılmış hafıza sistemlerinde işlemciler arası verilerin paylaşılabilmesi için bir haberleşme ağına ihtiyaç vardır.



Şekil 3.11 Dağıtılmış Hafıza Modeli

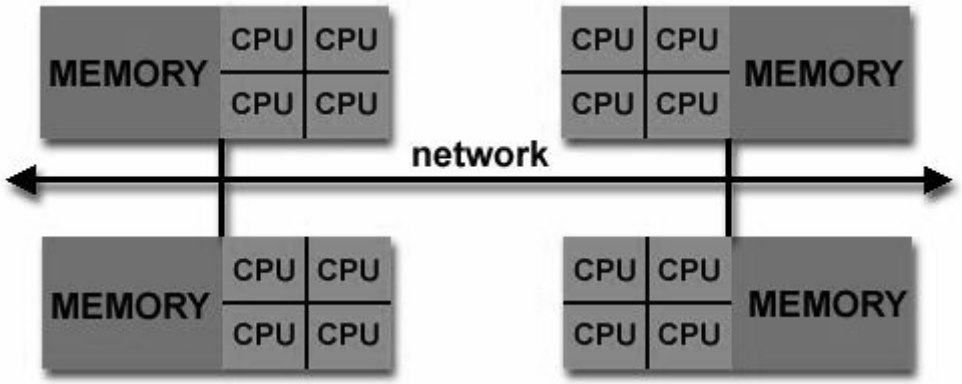
Her işlemcinin kendi lokal hafıza sistemi vardır. Bir işlemcinin lokal hafızasına diğer işlemciler tarafından ulaşılamaz. İşlemciler arasında paylaşılan ortak bir hafıza adres uzayı yoktur. Her işlemci kendi lokal hafızasına sahip olduğu için diğer işlemcilerden bağımsız olarak işlem yapabilmektedir. İşlemcilerin kendi lokal hafızalarında yaptıkları değişiklikler diğer işlemcilerin lokal hafızalarını etkilememektedir. Bu sebepten hafıza tutarlığı sistemine ihtiyaç duyulmamaktadır. İşlemcilerden birisi bir başka işlemcinin datasına ulaşmak isterse bu haberleşmenin ne zaman ve ne şekilde olacağının tanımlanması programcının işidir. Görevler arasında senkronizasyonun sağlanması da programcının sorumluluğundadır. Veri transferi için genellikle ethernet teknolojisi kullanılmasına rağmen pek çok farklı türden teknoloji de kullanılmaktadır.

İşlemci sayısı artırıldığı oranda hafıza miktarı da orantılı olarak artacaktır. İşlemciler her hangi bir ara yüz veya hafıza tutarlılık sistemleri kullanmadan kendi lokal hafızalarına ulaştıkları için bu işlem çok hızlı gerçekleşmektedir. Çok işlemcili tek bir bilgisayara göre maliyeti ve verimi daha düşüktür.

Dağıtılmış hafıza sisteminin bir takım dezavantajları da vardır. İşlemciler arasında veri haberleşmesinin tüm detayları tamamen programcının sorumluluğu altındadır. Bu tür hafıza organizasyonlarında veri yapılarına ulaşmak daha zor ve zaman alıcı olmaktadır.

Hafıza kullanım modellerini genel olarak ele alırsak UMA sistemlerde hafıza ulaşımı çok hızlıdır fakat işlemci sayısı 10'larla ifade edilebilmektedir. NUMA sistemlerde hafızaya ulaşım UMA sistemlerine göre daha fazla zaman almaktadır ve işlemci adedi 100'lerle ifade edilebilmektedir. Dağıtılmış hafıza sistemlerinde ise program geliştirmek ve yönetmek oldukça zahmetli bir iştir ve hafızalar arası veri paylaşımı daha çok zaman almaktadır. Ancak işlemci adedi binler hatta on binlerle ifade edilebilmektedir.

Günümüzde dünyanın en gelişmiş ve hızlı bilgisayarları paylaşılmış ve dağıtılmış hafıza mimarilerini bir arada kullanmaktadır. Bu yapı şekil 3.12'de temsil edilmiştir.



Şekil 3.12 Çok İşlemcili Dağıtılmış Hafıza Modeli

Bu sistemlerde her birim paylaşılan hafıza (shared memory - SM) hafıza tutarlılık sistemlerine sahip bir senkron çok işlemcili SMP bilgisayardan meydana gelirken bu bilgisayarlar bir network üzerinde birleşerek dağıtılmış hafıza sistemini

(distributed memory - DM) sistemini meydana getirmektedir. Tabiidir ki böyle bir sistem hem paylaşılan hafıza sisteminin hem de dağıtılmış hafıza sisteminin avantaj ve dezavantajlarına sahiptir. Bu sistemlerin programlanması da bağımsız olarak her iki sistemin programlanmasında daha karmaşıktır.

3.4 PARALEL PROGRAMLAMA MODELLERİ

Günümüzde kullanılan pek çok paralel programlama modeli mevcuttur. Bunlardan başlıcaları şunlardır:

- Paylaşılan Hafıza (Shared Memory)
- Thread
- Mesaj Aktarmalı (Message Passing)
- Data Paralel
- Simple Program Multiple Data (SPMD)
- Multiple Program Multiple Data (MPMD)
- Hibrit

Bu modellerden hiç biri özel bir bilgisayar veya hafıza modeline mahsus değildir. Hatta bu modellerin bir arada kullanıldığı pek çok örnek mevcuttur. Söz gelişi Kendall Savare (KSR) ALLCACHE yaklaşımında dağıtılmış hafıza modeli üzerine paylaşılan hafıza kullanılmıştır. Bilgisayarların hafızaları fiziksel olarak dağıtılmış olmasına rağmen programcıya tek bir paylaşılan hafıza gibi görünmektedir. Bu yaklaşıma "sanal paylaşılan hafıza" (virtual shared memory) denilmektedir. Bir diğer örnekte ise SGI Orijin üzerinde MPI kullanılarak paylaşılan hafıza modeli üzerinde mesaj aktarmalı model kullanılmıştır. Burada ise SGI Orijin CC-NUMA mimarisi kullanılmaktadır. Böylece her görev global hafızaya doğrudan ulaşabilmektedir. Buna ilave olarak MPI (Message Passing Interface) yardımıyla network üzerinden mesaj gönderme ve alma işlemini gerçekleştirmekte, böylece dağıtılmış hafızaya ulaşabilmektedir.

Görüldüğü gibi “en iyi” olarak niteleyebileceğimiz bir model mevcut değildir. Yaptırılacak işin ve algoritmanın karmaşıklığını dikkate alarak uygun modeli seçmek en doğru davranış olacaktır.

3.4.1. Paylaşılan Hafıza Paralel Programlama Modeli

Paylaşılan hafıza programlama modelinde görevler aynı adres uzayını paylaşabilmektedir. Böylece aynı hafıza bölgesinden asenkron olarak okuma ve yazma yapılabilmektedir. Veri güvenliğini sağlamak için kilitler veya semaforlar kullanılır. Böylece farklı görevlerin aynı hafıza alanına aynı anda ulaşması ile ortaya çıkabilecek veri kaybı önlenmiş olur. Bu da verilerin anlık olarak görevlere tahsis edilip daha sonra serbest bırakılması yoluyla görevler arası veri haberleşmesi yükünü ortadan kaldırmaktadır. Elbette bu tür kontroller hem programlamaya tasarım ve yönetim zorluğu getirmekte hem de performans düşüklüğüne sebep olmaktadır.

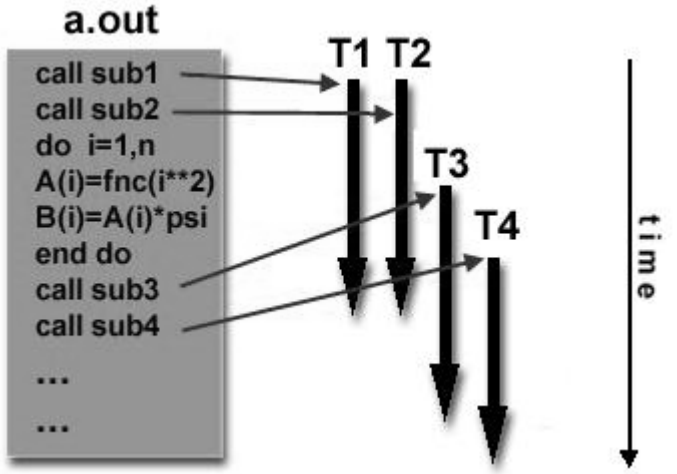
Bu modelde kullanılan dil derleyicileri de programda kullanılan değişkenler için aynı global hafıza adres uzayını kullanmaktadırlar. Yine bu modelde doğrudan dağıtılmış hafıza yapısı kullanılmamakla birlikte KSR ALLCACHE yaklaşımında olduğu gibi fiziksel olarak dağıtılmış hafıza kullanılarak görüntü olarak paylaşılan hafıza yaklaşımı da kullanılabilmektedir.

3.4.2. Thread Paralel Programlama Modeli

Thread paralel programlama modelinde aynı anda yürütülen pek çok modülden meydana gelen tek bir işlem mevcuttur. İşletim sistemi başlangıçta bu tek işlemi yürütür. Bu işlem başlangıçta bir takım seri işler yürüttükten sonra birçok alt rutin oluşturur ve bu işlem parçacıkları da işletim sistemi tarafından eş zamanlı

olarak yürütülürler. Her thread kendi lokal verisine sahip olmakla birlikte ana işlemin tüm kaynaklarını da paylaşabilmektedir. Böylece thread'ler arası veri haberleşmesi için zaman harcanmaz. Ayrıca her bir thread ana işleme ait global hafıza uzayına da ulaşabilmektedir.

Thread'ler kendileri de birer alt rutin olmakla birlikte ana programa ait tüm alt rutinleri de diğer thread'ler ile eş zamanlı olarak çalıştırabilirler.



Şekil 3.13 Thread Paralel Programlama Modeli, Barney, B.

Thread'ler global hafıza uzayı üzerinden haberleştikleri için program tasarımı sırasında birden fazla thread'in aynı hafıza bölgesinde aynı anda değişiklik yapmak istediği durumların iyi izlenmesi ve bu gibi durumlar varsa thread'ler anlık serileştirme teknikleriyle birinin yazma işlemi bitmeden diğeri yazamayacak şekilde işlem sırasına sokulma gerekmektedir.

Programın yürütülmesi boyunca birçok thread başlatılabilir ve bitirilebilir; ancak ana program modülü tüm işlemler sonuçlanana kadar çalışmasını sürdürmek zorundadır.

Thread'ler paylaşılan hafıza mimarisi ve bu mimariyi kullanan işletim sistemleri tarafından desteklenmektedir.

Programın paralelliği ve veri bütünlüğü tamamen programcının sorumluluğu altındadır. Thread kullanımı programcılık ve hesaplama teknikleri açısından hiç de yeni bir teknik değildir. POSIX thread'ler ve OpenMP thread'ler olmak üzere iki tür thread yürütme metodu vardır.

POSIX Thread'ler:

- Kütüphane tabanlıdır.
- Paralel kodlamaya ihtiyaç vardır.
- IEEE POSIX1003.1c standardına sahiptir (1995).
- Sadece C dilinde kullanılmaktadır.
- PThread olarak adlandırılmaktadır.
- Pek çok donanım üreticisi kendi thread araçlarına ilave olarak pthread kullanımını desteklemektedir.
- Programcı açısından oldukça dikkatli kodlama gerektirmektedir.

OpenMP Thread'ler:

- Derleyici tabanlıdır.
- Seri olarak kodlanabilir.
- Başlıca yazılım ve donanım üreticileri tarafından desteklenmektedir.
- 1997'den bu yana FORTRAN ve 1998'den bu yana C/C++ API'leri tarafından desteklenmektedir.
- UNIX ve Windows platformları gibi pek çok platform tarafından da desteklenmektedir.
- Kullanımı oldukça kolaydır.

Microsoft kendisine ait UNIX POSTIX ve OpenMP standartları dışında bir yapıya da sahiptir.

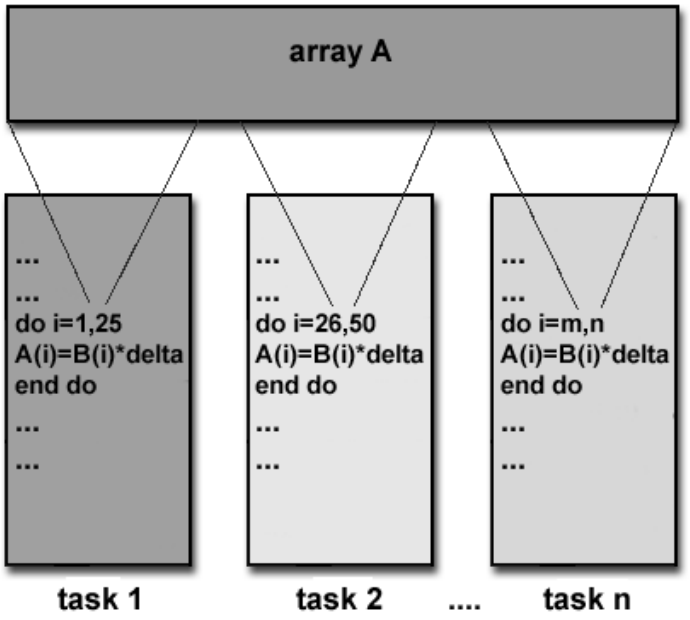
3.4.3. Mesaj Aktarmalı Paralel Programlama Modeli

Bu model de hesaplama işlemi her biri kendi lokal hafızasına sahip birden fazla işlem tarafından yürütülmektedir. İşlemlerin hepsi ayrı birer fiziksel makinede yürütülmektedir. İşlemler arasında veri haberleşmesi mesaj gönderme ve alma yoluyla sağlanmaktadır. Mesaj gönderme ve alma işlemi karşılıklı koordinasyonla sağlanmaktadır. İşlemler arasında mesaj gönderme ve alma işlemlerinin eş zamanlı yürütülmesi ve paralelliğin sağlanması tamamen programcının sorumluluğu altındadır.

Bu model 1980'lerden beri kullanılmaktadır. Ancak ilk modellerde programlama açısından mesaj gönderme ve alma sırasında senkronizasyonun sağlanması oldukça zordu. 1992 yılında mesaj aktarma işlemlerinde standart bir arayüz sağlanması amacıyla MPI Formu oluşturuldu. Günümüzde ise MPI bir mesaj aktarma standardı halini almıştır. Paralel hesaplama sistemlerinin pek çoğu kısmen veya tamamen bu standartları kullanmaktadırlar. MPI'ın paylaşılan hafıza (shared memory) mimarisi ile kullanılması durumunda ise işlemler arası haberleşmede network yerine performans açısından daha verimli çalışma için paylaşılan hafıza tekniği kullanılmaktadır.

3.4.4. Data Paralel Paralel Programlama Modeli

Data Paralel programlama modelinde işlerin her biri aynı veri kümesine odaklanırlar. Veri kümesi diziler, küpler ve benzeri kolay gruplandırılabilecek veri yapılarına sahiptir. Birden fazla görev aynı veri yapısı üzerinde çalıştırılır; ancak her görevin üzerinde işlem yaptığı bölüm şekil 3.14'te temsil edildiği gibi ana verinin farklı bir bölümüdür. Görevler aynı işlemi verinin yalnızca kendisine ait olan bölümünde gerçekleştirirler.



Şekil 3.14 Data Paralel Programlama Modeli, Barney, B.

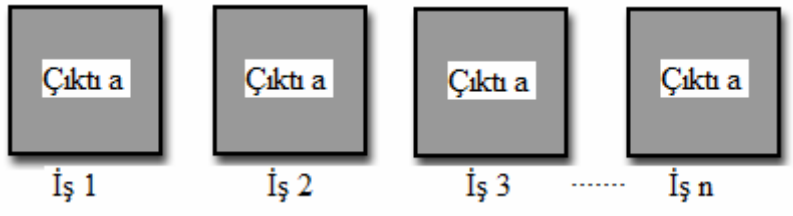
Paylaşılan hafıza modelinde tüm görevler kullanacakları veri parçasına global hafıza uzayından ulaşabilirler. Dağıtılmış hafıza modellerinde ise veri parçalara ayrılır ve her bir görevin kendi lokal hafızasına mesaj aktarma metoduyla aktarılır.

Data paralel programlama modeli data paralel alt rutin kütüphaneleri kullanılarak programlama sırasında veya data paralel derleyiciler tarafından derleme sırasında kullanılabilmektedirler. Derleyici tarafından yapılan işlemlerde mesaj aktarma işlemi tamamen programcıdan gizlenmiştir.

3.4.5. Simple Program Multiple Data (SPMD) Paralel Programlama Modeli

SPMD yüksek seviyeli bir paralel programlama metodudur. Paralel programlama metotlarının bir kombinasyonudur. Tek bir programın eş zamanlı olarak birçok örneği çalıştırılır. Bu programların her biri herhangi bir anda herhangi bir komutu yürütebilir. Duruma göre programlardan her biri içerdiği komutların

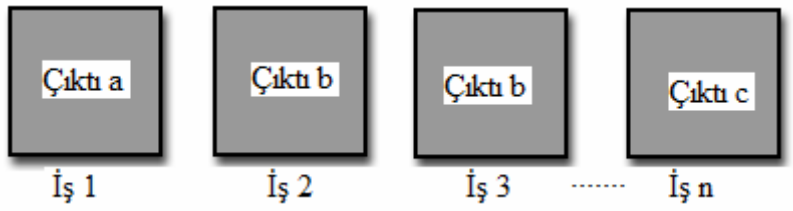
tamamını veya belirli bir bölümünü yürütebilir. Her program aynı veya farklı veriler üzerinde işlem yapabilir.



Şekil 3.15 SPMD Paralel Programlama Modeli

3.4.6. Multiple Program Multiple Data (MPMD) Paralel Programlama Modeli

SPMD gibi MPMD de yüksek seviyeli bir programlama modelidir ve paralel programlama metotlarının bir kombinasyonudur. SPMD modelinden farklı olarak MPMD modelinde birbirinden farklı birçok işlem (program) mevcuttur. Farklı işlemler eş zamanlı çalışırken SPMD modelinde olduğu gibi aynı işlemin de birden fazla örneği çalışıyor olabilir ve her işlem farklı veri kullanabilir.



Şekil 3.16 MPMD Paralel Programlama Modeli

3.4.7. Hibrit Paralel Programlama Modeli

Bu modelde iki veya daha fazla paralel programlama modeli bir arada kullanılmaktadır. Hibrit modele örnek olarak mesaj aktarmalı model (MPI) ile thread model (POSTIX thread) veya paylaşılan hafıza modelini (OpenMP) gösterebiliriz. Bu hibrit model bir network üzerinde birleştirilmiş SMP bilgisayarlardan meydana

gelmektedir. Bir diğer örnek de data paralel model ve mesaj aktarmalı modelin birleşimi olarak düşünülebilir.

3.5 PARALEL PROGRAM TASARIMI

Paralel bir programın tasarımı ve yazımı sırasında neredeyse her şey manuel olarak yapılmaktadır. Paralelliğin program içinde tanımlanması, yürütülmesi ve veri bütünlüğü korunması tamamen programcının sorumluğundadır. Paralel program geliştirmek oldukça karmaşık, zaman alıcı ve hata yapmaya müsait bir işlemdir. Seri programları paralel hale getirmede programcıya yardımcı olabilecek birçok unsur mevcuttur. Bunlardan bazıları seri programları paralelleştiren derleyiciler veya önişlemcilerdir.

Paralelleştirme özelliğine sahip derleyiciler bunu iki türlü yaparlar:

1. Tam Otomatik: Paralelleştirme derleyici kaynak kodunu gözden geçirir ve paralelleştirilmeye uygun kodları tanımlamaya çalışır. Paralelleştirmeye uygun kodlar için performans değerlendirmesi yapar. Böylece kodun bir kısmında paralelleştirme işlemi yapılırsa performansta bir artış olup olmayacağına karar verir. Paralelleştirmeye en uygun kod parçaları genellikle döngülerdir.

2. Programcı Kontrollü Paralelleştirme: Programcı derleyici yönlendirmeleri veya derleyici flagları kullanarak derleyiciye kodun nasıl paralelleştirileceğini belirtir. Bir miktar otomatik paralelleştirmeden de faydalanabilir.

Otomatik paralelleştirme her ne kadar daha eziyetsiz gibi gözükse de her zaman istenilen sonuçları vermeyebilir.

- Hatalı sonuçlar üretebilir.
- Performansta düşme olabilir.
- Manuel paralelleştirmeye göre daha esnektir.

- Paralel alt küme sayısı sınırlıdır.
- Gerçekte paralelleştiremeyecek bir kodu yanlış analiz ederek çok karmaşık bir şekilde paralelleştirebilir.

3.5.1. Programın Anlaşılması

Paralel bir program yazmak için atılacak ilk adım doğrusal programlarda olduğu gibi problemin iyice anlaşılması olmalıdır. Ele aldığınız problemi çözmek üzere paralel kod tasarlamaya başlamadan önce problemin paralel çözüme uygun olup olmadığının çok iyi değerlendirilmesi gerekmektedir.

Paralelleştirilebilir Problem Örneği: 3 ile 1.000.000 arasındaki asal sayıların toplamının bulunması

Bu problem paralelleştirilmeye oldukça uygundur. Bu aralıktaki tek sayılar "n" çalıştırılacak işlem sayısı olmak üzere 2*n adım aralığı ile n işleme hesaplattırılır. İşlemlerin döndüreceği n adet değerin toplanması ile sonuç elde edilir.

Paralelleştirilemeyen Problem Örneği: Fibonocci serisinin hesaplanması (Fibenocci Serisi: 1,1,2,3,5,8,13,21…..)

Bu serinin özelliği her elemanın kendisinden bir önceki ve iki önceki elemanların toplamından elde ediliyor olmasıdır. Yani fibonacci serisi f(n) = f(n-1) + f(n-2) serisinin her elemanı kendinden önceki elemanların hesaplanmasına bağlı olarak hesaplanacaktır. Herhangi bir n elemanı kendinden önceki değerler hesaplanmadan hesaplanamaz. Bu sebepten dolayı problem paralelleştirilmeye uygun değildir.

Programda asıl işin yapıldığı kısmı iyi tespit etmek gerekir. İyi düzenlenmiş bilimsel ve teknik programlarda bu birkaç bölümde yoğunlaşmıştır. Bu kısımlardan

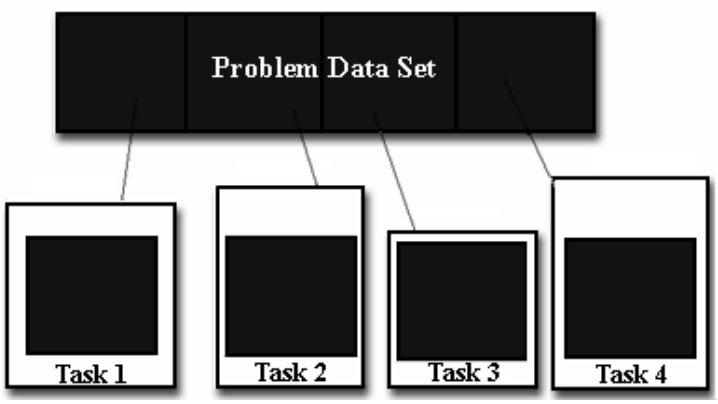
mikroişlemciyi fazla meşgul etmeyenlere dokunmadan, yalnızca zaman alıcı kısımların paralelleştirilmesi üzerinde çalışmak gerekmektedir.

Darboğazlar önceden belirlenmelidir. Program akışının dikkat çekici şekilde yavaşladığı noktalar göz önünde bulundurulmalıdır. Söz geliş giriş-çıkış (I/O) işlemleri bu konuda göz ardı edilmemesi gereken noktalardan birisidir. Bu durumların ortadan kaldırılması için program yapısının yeniden düzenlenmesi veya daha uygun bir algoritmanın seçimi yoluna gidilebilir. Paralelleştirilmeye engel olabilecek durumları dikkatlice tespit etmek gerekir. Bunlardan en başta geleni de fibonocci serisinde olduğu gibi veriye bağlılık durumudur. Mümkünse farklı algoritmalarla çözüme gitme yolu denenmelidir. Aksi halde paralelleştirmede ısrar etmek olumsuz sonuçlar doğurabilmektedir.

3.5.2. Parçalara Ayırma

Paralel algoritma tasarımındaki ilk adım problemin bağımsız işlemler tarafından yürütülebilecek parçalara ayrılmasıdır. Bu işe "bölme" veya "parçalama" adı verilir. Hesaplama işleminin paralel görevler arasında parçalanması için iki temel yol vardır. Domain Decomposition ve Fonksiyonal Decomposition.

- **Domain Decomposition:** Bu tür parçalara ayırma metodunda probleme ait veri bloğu parçalanır ve şekil 3.17'de gösterildiği gibi her parça bir görev (task) tarafından işlenir.



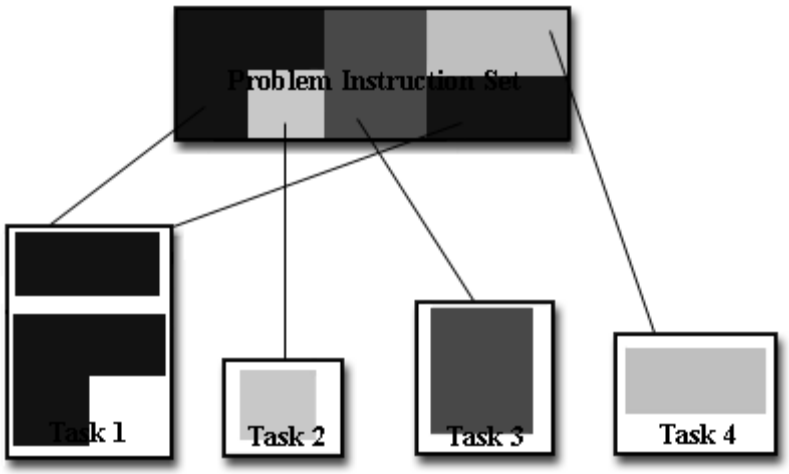
Şekil 3.17 Domain Decomposition Parçalama

Dizi, matris veya küp yapısına sahip veriler için uygun bir yöntemdir.



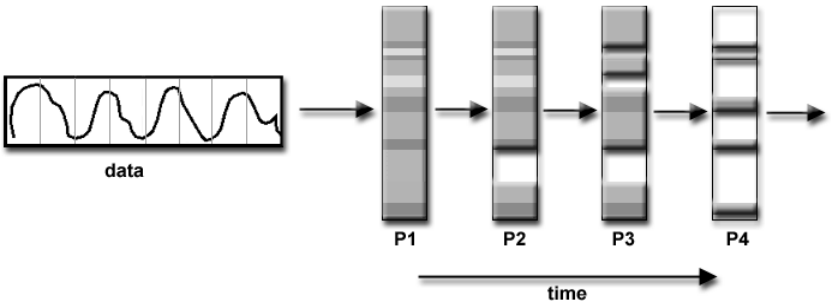
Şekil 3.18 Dizi/Matris/Küp Parçalama

- **Fonksiyonel Parçalama:** Bu yaklaşımda verinin parçalanıp görevlere atanması yerine hesaplama üzerinde yoğunlaşılır. İş parçacıkları ayrı birer fonksiyonel görev olarak çalıştırılır. Her iş parçası kendisine gereken eriye gerektiği zaman ulaşır.



Şekil 3.19 Fonksiyonel Parçalama

Fonksiyonel parçalama metodunu bir sinyal filtreleme işlemi üzerinde inceleyelim. Bir ses sinyalinin dört adet bağımsız filtreden geçirileceğin kabul edelim. Her bir filtre aslında birer hesaplama işlemidir ve diğer filtrelerden bağımsızdır. Bu durumda her filtre verinin kendisini ilgilendiren kısmıyla diğer filtre işlemlerinden bağımsız ve onlara paralel olarak ilgilenebilmektedir. Ses filtreleme işlemi şekil 3.20'de sembolize edilmiştir.



Şekil 3.20 Ses Filtreleme Örneği, Barney, B.

3.5.3. Haberleşme

Görevler arasında haberleşme ihtiyacı tamamen probleme bağlıdır. Bazı görevler sanal olarak tamamen ayrı çalışırlar ve kendi aralarında veri paylaşımına ihtiyaç duymazlar. Örnek olarak siya-beyaz bir fotoğrafın negatifini üreten programı ele alalım. Problem, siyah noktaların beyaza, beyaz noktaların ise siyaha dönüştürülmesi olarak tanımlanmıştır. Bu problemin paralelleştirilmesi oldukça kolaydır ve paralel görevlerden hiç birisi diğerleriyle haberleşme ihtiyacı duymamaktadır. Bu tür problemler "sıkıcı paralel" (embarrassingly parallel) olarak da adlandırılırlar. Paralel işlemlerin neredeyse tamamen ayrık olarak çalışması ve çok az haberleşmeye ihtiyaç duyulması sebebiyle bu isim verilmektedir.

Pek çok paralel programda ise işlem çoğu zaman bu kadar sessiz olarak yürümemektedir. Bu tür programlarda görevler kendi aralarında sık sık veri paylaşımı yaparlar. Örnek olarak üç boyutlu (3D) bir ortandaki ısı geçiş (difüzyon) problemini ele alalım. Bu örnekte her görev komşu görevlerin hesapladığı sıcaklık değerlerinden faydalanmak zorundadır. Herhangi bir görevin verilerindeki değişiklik komşu veriler üzerinde işlem yapan görevin kullanacağı verileri doğrudan etkilemektedir.

Haberleşme sırasında dikkat edilmesi gereken faktörleri şöyle sıralayabiliriz:

Haberleşme Maliyeti: Görevler arası haberleşme genellikle en çok zaman alan işlemlerden birisidir. Bir görevin diğer bir görevle haberleşme senkronizasyonu sağlamak amacıyla işlem yapmak yerine diğer görevin göndereceği veya alacağı veriyi beklemesi ciddi zaman kaybına yol açabilmektedir.

Haberleşme trafiğinin yeterince sağlıklı ve hızlı sağlanabilmesi için önemli bir husus da bilgisayar ağının band genişliğidir (bandwidth). Bilgisayar ağının band genişliğinin yeterli olmadığı durumlarda veri haberleşmesi için harcanan zaman daha da artacaktır. En küçük bir mesajın (1 byte) bir bilgisayardan diğer bilgisayara gönderilebilmesi için geçen süreye "latency" denilir ve mikrosaniyelerle ifade edilir.

Birim zamanda aktarılan veri miktarına ise “band genişliği” adı verilir ve megabyte/saniye ile ifade edilir.

Mesajların birçok küçük paket halinde gönderilmesi yerine, birçok küçük mesajın birleşiminden meydana gelen daha büyük ve az sayıda paket göndermek haberleşmedeki zaman kaybını azaltacaktır. Toplam boyutu aynı olan haberleşmelerden paket sayısı az olan “latency” zaman kaybı az olduğu için daha hızlı tamamlanacaktır.

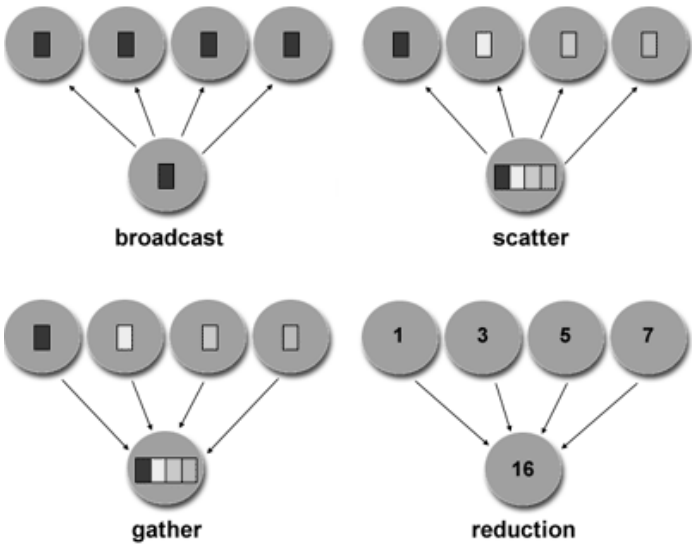
Haberleşmenin İzlenilebilirliği: Mesaj aktarmalı modelde haberleşme programcının kontrolü altındadır ve oldukça izlenebilir durumdadır. Data paralel modelde ise programcı görevler arası haberleşmenin ne şekilde olduğunu bilmek zorunda değildir.

Senkron ve Asenkron Haberleşme: Senkron haberleşmede görevler arası veri paylaşımı sırasında “handshaking” adı verilen karşılıklı haberleşme sinyalizasyonu söz konusudur. Bu işlem genellikle programcı tarafından kod ile gerçekleştirilmektedir. Bazı durumlarda düşük seviyeli olarak kullanıcı kontrolü olmaksızın da handshaking işlemi gerçekleştirilebilir. Senkron haberleşmeye haberleşme tamamlanmadığı sürece işlem akışını durdurmak için sıkça başvurulur. Asenkron haberleşme görevlere diğer görevlerden bağımsız olarak veri transferi imkanı tanır. Buna göre A görevi B görevine bir mesaj göndermesi gerektiğinde mesajı gönderir ve B’nin mesajı alıp almadığına bakmaksızın işine devam eder. Böylece haberleşmenin sonlanmasını beklerken işlem zaman kaybına uğramazlar. Asenkron haberleşmenin en önemli üstünlüğü hesaplamanın haberleşmeyi beklememesidir.

Haberleşme Bölgesi: Programda hangi görevlerin birbirleriyle haberleşeceğinin bilinmesi paralel kod tasarımı sırasında kritik bir husustur. Aşağıdaki bölge tanımlamalarının her ikisi de hem senkron hem de asenkron olarak uygulanabilmektedir:

1. Noktadan Noktaya (Point-to-Point): İki görevden birisi alıcı, diğeri gönderici olarak çalışır.

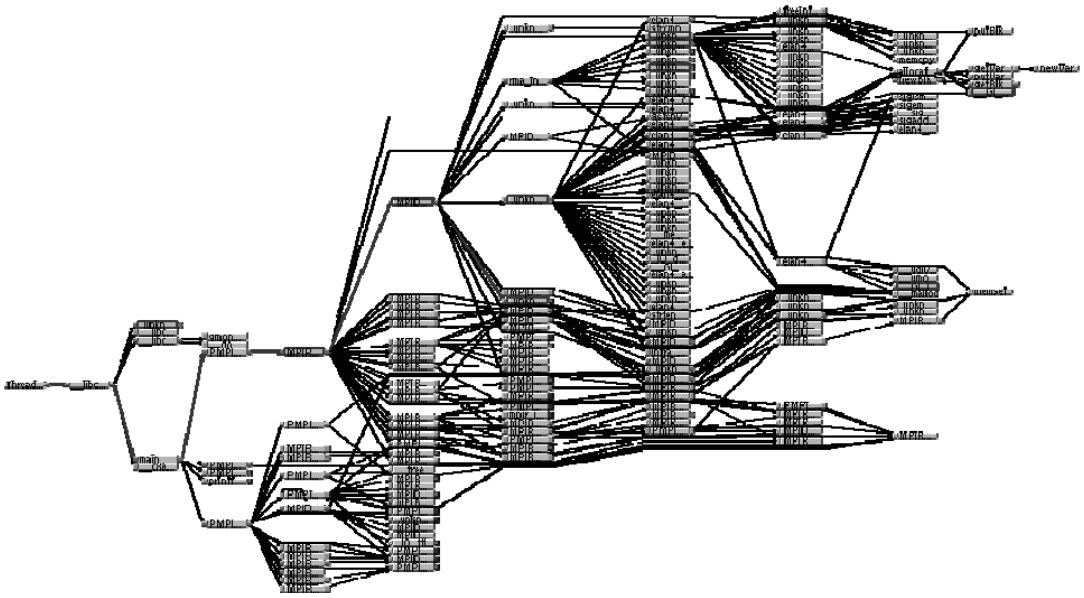
2. Kolektif: Belli bir grubun üyesi olan ikiden fazla görev arasında veri paylaşımı gerçekleşir. Bunun bazı türleri resimde görülmektedir.



Şekil 3.21 Haberleşme Bölgeleri, Barney, B.

Haberleşmenin Verimliliği: Programcılar Haberleşme performansını etkileyen faktörleri dikkate alarak haberleşme teknikleri arasında tercihlerini yaparlar. Bu tercihleri yaparken haberleşme "mesaj aktarmalı mı?" yoksa "hafıza paylaşımlı mı?"; "senkron mu?" yoksa "asenkron mu?" gibi soruların aslında problemin içinde saklı olduğunu unutmamak gerekir.

Karmaşıklık Maliyeti: Haberleşmenin programa maliyeti programın karmaşıklığıyla doğru orantılıdır. Hem paralel program yazmaya karar verirken hem de haberleşme teknikleri arasında tercih yaparken programın karmaşıklığı asla göz ardı edilmemelidir. Yazılacak program bazen şekil 3.22'deki gibi oldukça karmaşık bir yapıya sahip olabilir.



Şekil 3.22 Karmaşıklık, Barney, B.

3.5.4. Senkronizasyon

- **Bariyer:** Bariyerler tüm görevleri etkilerler. Tüm görevler bariyere ulaşıncaya kadar çalışmalarına devam ederler. Bariyere uğrayan görev durdurulur ve bekletilir. Son görev de bariyere uğradığı anda görevler senkronize olurlar. Bu anda genellikle programın serileştirilmesi gereken bölümleri işletilir. Ardından tüm görevler otomatik olarak işlerine kaldıkları yerden devam ederler.

- **Kilit/Semafor:** Her hangi sayıda görevi etkileyebilmektedir. Genellikle belirli bir kod veya veri bölümünü serileştirmek için kullanılır. Aynı anda yalnızca bir tek görev kilidi ele geçirir. Kilide sahip olan görev güvenli bir şekilde, korunmuş veriye veya koda ulaşır. Diğer görevler kilit serbest bırakılıncaya kadar beklemek zorundadırlar. Kilitle bekletilen görevler kuyruk veri yapısıyla sıraya geçerek kilide sahip olmak için beklerler.

- **Handshaking:** Sadece haberleşecek görevlerin arasında gerçekleştirilir. Bir görev diğer bir görevle haberleşmek isterse aralarında bir takım haberleşme sinyalleri ile koordinasyon sağlanır. Bunu "haberleşmeye hazır mısın?" sorusuna cevap vererek "evet" sinyalinin gelmesini beklemek olarak düşünebiliriz.

3.5.5. Veri Bağlılığı

Veri bağlılığı program komutlarının çalışma sırasının programın sonucuna etki etmesi olarak tanımlanabilir. Aynı hafıza bölgesindeki veriye farklı görevler tarafından ulaşılıyorsa veri bağlılığı ortaya çıkar. Veri bağlılığı paralel programlamada çok önemli bir husustur; çünkü paralel programlamanın başlıca engellerinden birisi veri bağlılığıdır.

ÖRNEK:

```
for(i=n ; i<m ; i++)
{
        A[i] = A[i-1] * 2.5;
}
```

Örnekte A[i-1] değeri hesaplanmadan A[i] değerini hesaplayamayız. A[i] değeri A[i-1] verisine bağlıdır. Bu durum paralelleştirmeyi engellemektedir.

ÖRNEK:

Görev 1	Görev 2
X = 2 ; .	X = 4 ; .

. . Y = X * 2 ;	. . Y = X * 3 ;

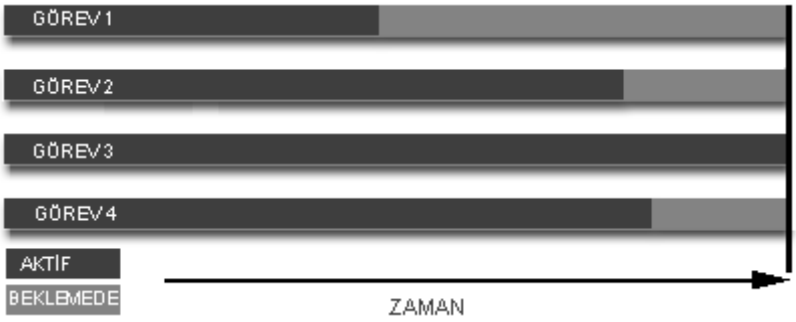
Örnekte Y'nin sahip olacağı değer hangi görevin X değerini en son değiştirdiğine ve hangi görevin en son bittiğine göre farklılık gösterecektir.

Paralelleştirmenin en çok uygulandığı yer döngüler olduğu düşünülürse veri bağlılığının seçilen algoritmadaki önemi daha iyi anlaşılmaktadır.

Veri bağlılığı bulunan durumlarda görevler arası senkronizasyon sağlandıktan sonra veri haberleşmesi yapılmaktadır.

3.5.6. Yük Dengeleme (Load Balancing)

Yük dengeleme görevler arasında iş dağılımı yaparken her işin yaklaşık aynı miktarda iş yükünü alması anlamına gelmektedir. Boşta bekleyen görev zamanını minimuma indirmek, böylece tüm işin bitiş süresini kısaltmak için uygulanır. Paralel programlamada performans açısından çok önemlidir. Program içinde bir bariyer kullanıldığı düşünülürse genel performansı en yavaş çalışan görev belirleyecektir.



Şekil 3.23 Görev-Zaman Dağılımı

Yük dengelemesini sağlamak için aşağıdaki metotlardan herhangi biri kullanılabilir:

1. Yapılacak iş görevler arasında eşit olarak paylaştırılabilir. Dizi, matris ve küp benzeri işlemlerde veri kümesi görevler arasında kolayca paylaştırılabiliyorsa ve her görev aynı işi yapacaksa bu metodu tercih etmek yük dengeleme açısından uygundur. Veri bağlılığı olmayan döngülerin görevler arasında paylaştırılması da uygun bir düşüncedir. Heterojen bilgisayarlardan meydana gelen bir sistemde yük dengeleme yapmak her zaman bu kadar kolay değildir. Bu sistemler için performans analiz yazılımlarına ihtiyaç vardır.

2. İş ataması dinamik olarak yapılabilir. Bazı problemlerde görevlere eşit iş dağılımı yapılsa bile yük dağılımını eşit olarak sağlayamayabiliriz. Dağınık dizilerdeki gibi işlenecek verinin homojen olmadığı durumlar mevcuttur. Bazı görevlere ait verilerin çoğunu işlenmesi gerekirken bazı görevlerin verilerinin pek çoğu işlenmeye ihtiyaç duymayabilir.

İki ayrı renk filtresinin iki görev tarafından yürütüldüğünü düşünelim. Resimde hâkim olan rengi işlemesi gereken görev diğer göreve göre daha fazla iş yüküne sahip olacaktır.

Uyarlamalı grid metotlarında bazı görevler bileşenlerini arıtmaya çalışırken bazıları buna ihtiyaç duymayacaktır.

N-Body simülasyonlarında bazı görevlere ait veriler iş yoğunluğu daha az olan görevlere aktarılabilmektedir. Bazı durumlarda kasten veya algoritma gereği iş yükünün miktarı tahmin edilemeyebilir. Bu gibi durumlarda iş parçacıklarını bir havuzda bekleterek, işi biten her göreve yeni bir iş parçası vermek çok iyi bir teknik olabilecektir.

Dengesiz yük dağılımlarını program akışı sırasında tespit eden ve ona göre dinamik olarak yük dağılımı yapan bir algoritma bu durumlar için en uygun çözümdür. Elbette ki her problem buna imkân tanımayabilir.

3.5.7. Parçacıklılık (Granularity)

- **Hesaplama/Haberleşme Oranı:** Paralel hesaplamada parçacıklılık hesaplamanın haberleşmeye oranını en iyi belirleyen niteliktir. Hesaplama periyotları tipik olarak haberleşme periyotlarına bölünerek eş zamanlı hale gelmişlerdir.

- **Küçük Taneli (Fine-Grain) Paralellik:** Haberleşme olayları arasında nispeten kısa süreli hesaplama işlemleri yapılmaktadır. Hesaplama/haberleşme oranı çok düşüktür. Yük dengeleme işlemi kolaylıkla yapılabilmektedir. Haberleşme fazla olduğu için performans arttırma imkânı azdır. Eğer tanecikli yapı çok fazlaysa haberleşme fazlasıyla ön plana çıkmaktadır ve görevler arası eş zamanlılık hesaplamadan çok daha fazla zaman harcayacaktır.

- **İri Taneli (Coarse-Grain) Paralellik:** Haberleşme olayları arasında nispeten daha büyük hesaplamama işlemleri gerçekleştirilmektedir. Hesaplama/haberleşme oranı yüksektir. Performans arttırma imkânı daha yüksektir. Verimli bir şekilde yük dağılımı yapmak ise oldukça zordur.

Tanecikli yapılardan hangisinin daha verimli olduğunu söylemek çok zordur. Aslında verimlilik kullanılan algoritmaya ve o algoritmanın yürütüleceği donanım birimlerine bağlıdır. Genellikle en yüksek gecikmenin haberleşme ve eş zamanlılık yüzünden olduğu düşünülürse iri tanecikli yapının daha üstün olduğu söylenebilir.

3.5.8. Giriş-Çıkış (I/O)

Giriş-çıkış işlemleri genellikle paralelleştirmedeki en önemli engellerdendir. Paralel giriş-çıkış işlemleri hem henüz yeterince gelişmiş değildir, hem de tüm platformlar tarafından desteklenmemektedir. Aynı dosya alanlarının görevler tarafından ortak kullanıldığı çevre birimlerinde yazma işlemleri veri kaybına sebep olabilir. Eş zamanlı okuma işlemlerinin dosya sürücüsü tarafından desteklenmesi gerekir. Bilgisayar ağı üzerinden gerçekleştirilen giriş-çıkış işlemleri ciddi darboğazlara sebep olabilir. Elbette giriş-çıkış işlemlerine her zaman bu kadar karamsar yaklaşmak doğru değildir. Bazı paralel dosya sistemleri bu olumsuz durumları çoğu zaman ortadan kaldırmaktadır. Bunlardan bir kaçını şöyle sıralayabiliriz:

- GPFS: General Parallel File System for AIX (IBM)
- Luster: Linux Cluster'lar için (Cluster File Systems, Inc.)
- PVFS/PVFS2: Linux Cluster'lar için Paralel Sanal Dosya Sistemleri (Clemson/Argonne/Ohio State)
- PanFS: Linux Cluster'lar için Panasas ActiveScale Dosya Sistemi (Panasas, Inc.)
- HP SFS: HP StorageWorks Scalable File Share (Linux için Global Dosya Sistemi)

MPI-2 için 1996 yılından beri paralel programlama arayüzü mevcuttur.

Giriş çıkış işlemleri için tavsiye edilen bazı teknikler ise şunlardır:

- Program genelinde giriş-çıkış işlemleri mümkün olduğunca azaltılır.

- Seri giriş-çıkış işlemleri bir görev üzerinde toplanır. Diğer paralel görevlere paralel haberleşme yöntemleriyle veri bu görev üzerinden paylaştırılır. Örnek olarak T1 görevi dosyadan okuma işlemini gerçekleştirir ve diğer paralel

görevlere gerekli verileri T1 görevi paylaştırır. Yazma işlemini de benzer şekilde diğer görevlerden topladığı verilerle yalnızca T1 görevi yapar.

- Dağıtılmış hafıza sistemlerinde paylaşılan dosya sistemi kullanılıyorsa giriş-çıkış işlemleri lokal dosya sistemi üzerinde yürütülmelidir. Bu yaklaşım paylaşılan dosya sisteminden verimli olduğu gibi, ağ üzerinde o göreve tahsis edilmiş bir alanı (home directory) kullanmaktan da daha verimlidir.

- Her bir görev için ayrı (unique) dosya ismi kullanılabilir.

3.6. PARALEL PROGRAMLAMANIN LİMİTLERİ VE GİDERLERİ

3.6.1. Amdahl Kuralı:

Program kodunun paralelleştirilen bölümü P olmak üzere hızlanma şu şekilde olmaktadır:

$$\text{Hızlanma} = \frac{1}{1 - P}$$

Eğer kod paralelleştirilemiyorsa P=0 olacağından Hızlanma=1 olacaktır. Eğer kodun tamamı paralelleştirilebiliyorsa P=1 olur. Bu durumda Hızlanma=1/0 olacağından teorik olarak tanımsızdır. Eğer kodun %50'lik kısmı paralelleştirilirse Hızlanma=2 olacak yani programımız iki kat hızlanacaktır.

Paralel işlemleri yürütecek olan işlemci sayısını da hesaba katmamız gerekecektir. Bu durumda hızlanma ilişkimiz "N" işlemci sayısını temsil etmek üzere

$$\text{Hızlanma} = \frac{1}{(P / N) + S}$$

“P” paralel kod bölümlerini, “S” ise seri kod bölümünü temsil etmektedir.

Açıkça görülmektedir ki ölçeklenebilir paralellikte bir takım sınırlar mevcuttur. Mesela P=0.5, 0.9 ve 0.99 değerleri için hızlanma

Tablo 3.2 Parallelleşme Oranına Göre Hızlanma

N	P=0.50	P=0.90	P=0.99
10	1.82	5.26	9.17
100	1.98	9.17	50.25
1000	1.99	9.91	90.99
10000	1.99	9.91	99.02

Bununla birlikte problemin boyutu arttıkça performansın da arttığı açıkça görülmektedir.

2D Grid Hesaplama 85 saniye %85

Seri kısım 15 saniye %15

Grid boyutunu iki katına çıkartarak ve zaman adımlarını yarıya düşürerek problemin boyutunu artırabiliriz. Boyutu büyüdükçe paralellik oranı artan problemler, paralellik oranı sabit olanlara göre daha ölçeklenebilirdirler. Problem boyutu dört kat artırıldığında sonuçlar aşağıdaki gibi değişmektedir.

2D Grid Hesaplama 680 saniye %97.84

Seri kısım 15 saniye %2.16

3.6.2. Karmaşıklık:

Genellikle seri programlarla karşılaştırıldıklarında paralel programlar oldukça karmaşıktırlar. Çünkü olay yalnızca aynı anda çalışan birden fazla komut kümesinden ibaret değildir. Komut kümeleri arasındaki veri haberleşmesi işin asıl zahmetli kısmını oluşturmaktadır.

Karmaşıklık, yazılım geliştirme döngüsünün her adımında programcıya zaman maliyeti getirmektedir. Karmaşıklık tasarım, kodlama, hata ayıklama, iyileştirme ve bakım adımlarının her birinde bize zaman maliyeti getirecektir. Projemize sonradan katılabilecek kişilerle uyum zamanında gecikme yaşamamak için kod sadeliğine özen göstermek gerekmektedir.

3.6.3. Taşınabilirlik:

Standart hale gelmiş API sayesinde paralel programlarda taşınabilirlik eskiye göre çok daha kolay hale gelmiştir. Fakat hala seri işlemler için de geçerli olan bazı problemler paralel işlemlerde de yaşanmaktadır. Her ne kadar API'ler için bir takım standartlar geliştirilmişse de yine de kullanımlarındaki detaylarda farklılıklar vardır. İşletim sistemleri taşınabilirlikte önemli rol üstlenmektedir. Donanım mimarileri de hayli değişkenlik göstererek taşınabilirliğe etki etmektedir.

3.6.4. Kaynak İhtiyaçları:

Paralel programlamanın başlıca amacı programın çalışma süresini kısaltmaktır. Sekiz işlemcili bir sistemde bir saat çalışan bir paralel kod aslında işlemci saatiyle sekiz saat çalışmış demektir.

Paralel kodlar için gerekli hafıza miktarı da seri kodlarınkine göre daha fazladır. Tekrarlanan veriler, paralel çalışan kodlar ve bunlar için gerekli kütüphaneler daha fazla hafızaya ihtiyaç duyacaklardır.

Kısa çalışma süresine sahip paralel programların performansı seri programlara göre daha düşüktür. Paralel ortamlarla bağlantı kurulması, görevlerin oluşturulması, haberleşmesi ve sonlandırılması kısa süre çalışacak paralel kodlar için çalışma süresini oran olarak büyük kısmını teşkil edecektir.

3.6.5. Ölçeklenebilirlik:

Paralel programların performanslarının ölçeklenebilirliği birbiriyle ilişkili pek çok faktör tarafından belirlenir. Sadece sisteme daha fazla bilgisayar eklemek nadiren beklenen performans artışını sağlar. Ölçeklenebilirlik genellikle kullanılan algoritmanın yapısıyla ilgilidir. Bazı durumlarda sisteme daha fazla kaynak eklemek performansta düşüşe sebep olabilir.

Ölçeklenebilirlikte donanım faktörü önemli rol üslenmektedir. Örnek olarak aşağıdaki unsurları sıralayabiliriz:

- SMP sistemlerde hafıza-işlemci arası bus band genişliği
- Haberleşme ağı band genişliği
- Kullanılabilir hafıza miktarı
- İşlemci saat hızı

Paralel arayüz kütüphaneleri ve alt sistemler de programlarımızın ölçeklenebilirliğini sınırlandırabilir.

3.6.6. Performans Analizi ve Ayarları:

Paralel programların çalışması sırasında hata ayıklama, izleme ve analiz işlemleri seri programlara göre çok daha zordur. Birkaç tane paralel program izleme ve analiz programı mevcuttur. Bunlardan farklı platformlar için çalışanlar vardır. Bu programların da yaptıkları iş ölçeklenebilirlik analizinden öteye geçmemektedir.

3.7 WEB ÇİFTLİĞİ

Web sunuculardа yük dağılımını sağlamak amacıyla aynı web içeriğine sahip birden fazla web sunucusunun aynı domain adı altında, farklı IP adresleri adresleriyle tanımlanmasından oluşur.

İstemciler tarafından yapılan dns sorgulamalarında her isteği “Round Robin” algoritması ile sıradaki IP’ye yönlendirir. Böylece web sunucu içinde yük dengeleme (Load Balance) sağlanmaya çalışılır.

3.8. WEB SERVİSİ

3.8.1 Web Servisi Nedir?

Web servisleri, web üzerinde hizmet veren fonksiyonlar veya program parçalarıdır. Standart HTTP protokolünü kullanırlar. İstekler ve cevaplar SOAP (Simple Object Access Protocol) protokolü ile, XML formatında ve http protokolü üzerinden iletilir. Bu işleme RPC (Remote Procedure Call) adı verilir.

RMI’ın java nesneleri dışındaki nesnelere erişememe problemi benzeri problemleri yoktur.

Herhangi bir dille yazılabileceği gibi herhangi bir dille de çağırılabilirler.

HTTP haberleşmesi sırasında 80 numaralı port kullanıldığı için güvenlik duvarları tarafından nadir durumlar dışında problem çıkartılmamaktadır.

Dağınık uygulamalarda parametre veya prosedür değişikliğinde merkezi değerler kullanıldığı için veri bütünlüğü daha rahat sağlanmaktadır.

Veritabanı gibi kaynaklara doğrudan erişime izin vermeyip ancak servis fonksiyonları ile ulaşım sağlandığı için çok daha güvenli istemci uygulamaları yazılabilmektedir. Kullanıcı adları ile şifrelerin bulunduğu komutlar istemci yazılım içinde bulunmayıp sadece servis fonksiyonlarında bulunmaktadır ve ancak programcı tarafından görülebilmektedir.

3.8.2. Web Servislerinde Kullanılan Standartlar

- **XML (Extensible Markup Language):** Verilerin esnek bir şekilde temsil edilmesi için geliştirilmiş protokoldür.

- **SOAP (Simple Object Access Prothocol):** Web servisleri için tanımlanan veri iletim protokolüdür.

- **WSDL (Web Services Description Language):** Web servislerinin tanımlanmasında kullanılan standarttır.

- **WSRF (Web Services Resource Framework):** Modelleme ve web serviskaynaklarına ulaşmak için geliştirilen bir protokoldür.

- **UDDI (Universal Description, Discovery and Integration):** Web Servislerinin tanıtılması ve bulunması için kullanılan protokoldür.

4. UYGULAMA

4.1. Örnek Problem

Uygulamada “En Kısa Yolu Bulma” problemi örnek olarak ele alınmıştır. Konumları sabit “n” adet nokta veya şehir arasında gezinmek için tercih edilecek en kısa yolun bulunması istenmektedir. Noktalar arası mesafe elle girilmeyip veri dosyasından alınan konum değerleri ile doğrusal olarak hesaplanacaktır.

4.2. Mevcut Seri Çözüm

Labirentten çıkış bulmada kullanılan algoritmalara benzer özyinelemeli (rekürsif) bir fonksiyon ile her seferinde bir nokta başlangıç kabul edilerek tüm noktaların gezilmesi için gerekli alternatif yollar ele alınmış; her seferinde en kısa kabul edilenden daha kısa bir yol bulunum bulunmadığı kontrol edilmiştir.

Tüm alternatif yollar değerlendirildikten sonra bulunan en kısa yol sonuç olarak verilmiştir.

4.3. Mevcut Seri Çözümün Paralelleştirilmesi

Her bir paralel iş parçacığı farklı bir noktayı başlangıç noktası kabul ederek alternatif yolları ona göre değerlendirmektedir.

4.4. Uygulama Programının İşleyişi

Program öncelikle kullanıcı arayüzünden problemin kaç nokta için çözüleceğini öğrenir. Ardından noktaların adlarını, x ve y koordinat bilgilerini bir metin dosyasından okur.

Web çiftliği üzerinden kaç adet alternatif web servisi kullanabileceğini öğrenir. Web çiftliğindeki adreslere sırasıyla birer nokta referans olarak atar. Nokta sayısı web çiftliğindeki adres sayısından fazlaysa tüm noktaların bir adrese atanması yapılıncaya kadar "Round Robin" algoritması ile sunuculara yeni birer nokta ataması yapılır.

Başlangıç kabul edilen her nokta için bir thread oluşturulur. Noktaların koordinatları ve referans noktanın hangisi olduğu bilgisi thread tarafından SOAP protokolü kullanılarak XML formatında ve HTTP protokolü ile web servisine gönderilir.

Yönetici program thread'lerden gelecek sonuçların toplanmasını beklerken servisler hesaplama işlemini başlatırlar. Her bir servis kendisine referans olarak verilen noktadan yola çıkarak tüm noktaları gezmek için gerekli alternatif yolları bulur ve en kısa olan yolun izlediği şehirleri ve mesafeyi geri döndürür.

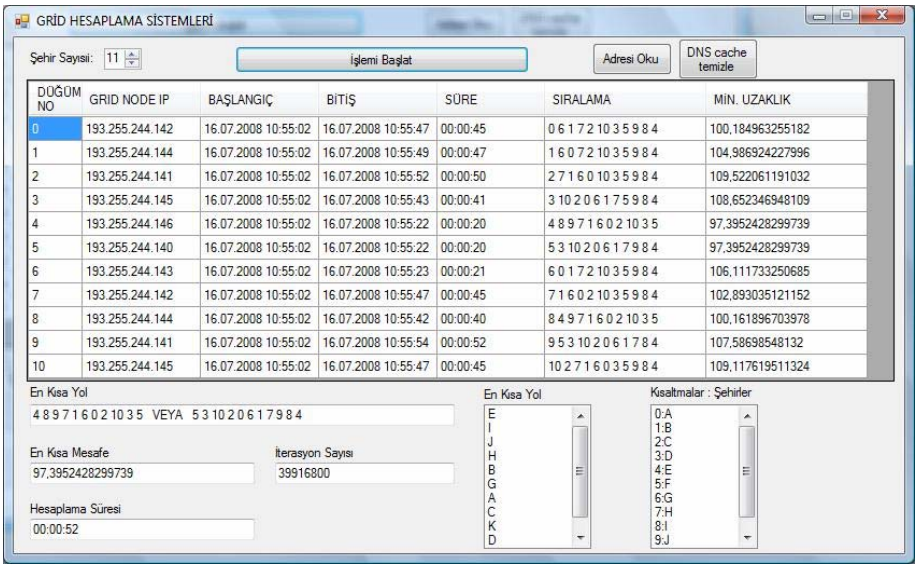
Tüm thread'ler sonuçlarını geri döndürdüğünde ana program değerlendirme yapar ve bulunan en kısa yolu kullanıcıya bildirir.

Şekil 4.1'de 11 nokta ararındaki en kısa yolun tek bilgisayar üzerinde hesaplanmış hali görülmektedir.



Şekil 4.1 Tek Bilgisayarda 11 Nokta İçin En Kısa Yol Hesaplaması

Şekil 4.2'de ise 7 bilgisayarlı grid hesaplama sistemimizde, 11 nokta arasındaki en kısa yol hesaplaması aynı algoritmanın paralelleştirilmiş hali ile hesaplanarak gösterilmiştir.



Şekil 4.2 7 Bilgisayarda 11 Nokta İçin En Kısa Yol Hesaplaması

4.5. Uygulama Programının Sonuçları ve Değerlendirme

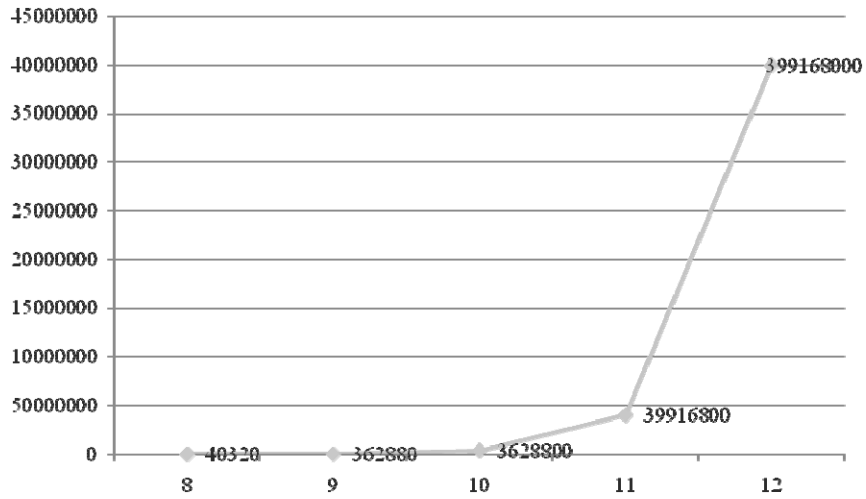
Noktalar arasındaki en kısa yolun hesaplanması için karşılaştırılması gereken alternatif yolların sayısı programın seri ya da paralel olmasına göre farklılık göstermemektedir. Ancak paralel iş parçacıklarının yapacağı kontrol sayısı "n" nokta için değil "n-1" nokta için olacaktır. Bu da n! hesaplama yerine n-1! hesaplama demektir.

Nokta sayısına bağlı olarak alternatif yol sayısındaki artış tablo 4.1'de görülmektedir.

Tablo 4.1 Nokta Sayısına Göre Alternatif Yol Sayıları

Nokta Sayısı	Alternatif Yol Sayısı
8	40320
9	362880
10	3628800
11	39916800
12	399168000
13	6227020800

Şekil 4.1 artışın yapısını daha iyi anlatmaktadır.



Şekil 4.3 Nokta Sayısına Göre Alternatif Yol Sayısındaki Artış

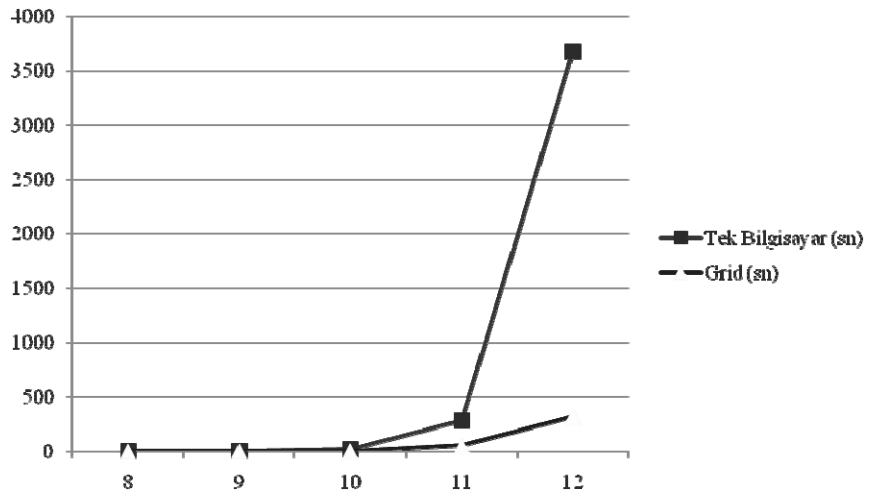
Nokta sayısının her bir artışı için alternatif yol sayısındaki artışın parabolik eğrisi bize bir şehri referans alarak n-1 şehir için kontrolleri gerçekleştirmenin ne kadar zaman kazandıracağını göstermektedir.

İşlem sayısındaki artışın işlem süresine yansıması ise tablo 4.2'de gösterilmektedir.

Tablo 4.2 Nokta Sayısına Göre İşlem Süreleri

Nokta Sayısı	Tek Bilgisayar	Grid
8	0.26 sn	0.51 sn
9	2 sn	1 sn
10	22 sn	4 sn
11	286 sn	52 sn
12	3680 sn	320 sn

Nokta sayısına bağlı olarak işlem süresindeki artışın şekil 4.4'teki grafik görüntüsü ise grid hesaplamanın bu problemdeki avantajı hakkında daha net fikir vermektedir.



Şekil 4.4 Nokta Sayısına Göre İşlem Süresindeki Artış (sn)

Az sayıda işlem için yapılan denemelerde tek bir hesaplama birimi tarafından yürütülen işlemlerin daha hızla sonuç verdiği gözlemlenmektedir. Bilgisayarlar arasındaki haberleşme sırasında ortaya çıkan zaman kaybı bu gecikmeye sebep olmaktadır. Ancak işlem sayısı on milyonlar seviyesi ve üzerinde seyrettiği denemelerde (11 şehir ve fazlası) network haberleşmesinden kaynaklanan gecikmeler ihmal edilebilir seviyede az zaman kaybına sebep olmaktadır. Şehir sayısının artması ile işlem sayısında parabolik bir artış ortaya çıkmaktadır. Bu yüzden çok sayıda işlem için network haberleşmesindeki gecikmeleri ihmal edecek olursak n şehir için yapılan grid hesaplama, n tane n-1 şehir için hesaplama yapan bilgisayarın paralel çalışması şeklinde gerçekleşmektedir.

5. SONUÇ VE ÖNERİLER

Grid Hesaplama sistemleri günümüzün gelişen hesaplama ihtiyaçlarını karşılamak için düşünülmüş en kapsamlı projelerden birisidir. Bu sistemlerin temel fikri, dünya çapında bir bilgisayar ağının tüm kaynaklarını, özellikle de işlem güçlerini tek bir bilgisayarmışçasına bir araya toplamaktır. Böylece dünyanın en güçlü bilgisayarından hem işlem hızı, hem bilgi saklama kapasitesi olarak onlarca kat daha güçlü bir bilgisayar sistemini ortaya çıkartmak hedeflenmiştir. Bu da çok daha düşük maliyetler ve iş gücüyle, daha az zamanda, daha fazla hesaplama işleminin gerçekleşmesi anlamına gelmektedir.

Üniversite seviyesinde uygulanmaya başlayan Grid Hesaplama düşüncesi dünya çapında pek çok devletin ve önde gelen şirketlerin de destekleriyle, ciddi yatırımların yapıldığı büyük projelerde aktif olarak kullanılmaktadır. Gelişen ağ bağlantı hızlarıyla yakın bir gelecekte çok daha büyük miktarda bilgisayar ve bilişim teknolojisi kaynağının katıldığı, daha da büyük Grid Hesaplama sistemleri kullanan projelerin ortaya atılması beklenmektedir. Bu da insanoğlunun yakın bir geçmişte hayal bile edemediği projelerin çok hızlı bir şekilde sonuçlanmasını kaçınılmaz hale getirmektedir.

Her seri çözüme uygun bir paralel çözüm mümkün olmayabilir. Her paralel metot da her paralel çözüme uygun olmayabilir. Paralelleştirilebilen çözümler seri çözümlere göre en azından doğrusal bir performans artışı göstermektedir. Uygulamada kullandığımız örnek benzeri durumlarda ise parabolik performans artışları görülmektedir.

Çalışmamızda yük dengeleme problemini aşmak için web çiftlikleri üzerinde Round Robin algoritması kullanılmıştır. Bundan sonraki çalışmalarda sistem

yoğunluğu izlenerek elde edilen veriler yardımıyla daha verimli yük dağılım teknikleri kullanılabilir. Böylece sistemin verimliliği daha da artacaktır. Ayrıca darboğazların ve arızalı düğümlerin tespiti için uygun çözümler aranmalıdır. Çünkü üretilmesi gereken çözümler programın yapısına göre farklılık gösterebilmektedir.

Türkiye Cumhuriyeti olarak bizim de yer aldığımız, insanlık tarihinin bu en büyük projesinin -daha önceki pek çok projenin aksine- insanlığın huzur ve refah içinde yaşaması için kullanılmaya devam etmesi ise en büyük temennimizdir.

6. KAYNAKLAR

1. Algan, S., 2008, www.csharpnedir.com.
2. Anonim, 2008, www.globus.org.
3. Anonim, 2008, www.llnl.gov.
4. Anonim, 2008, www.ulakbim.gov.tr.
5. Baker, M., Buyya, R., 1999, Cluster Computing At A Glance, s3-47, NY.
6. Barney, B., 2006, Introduction to Parallel Computing.
7. Brown, M., 2004, Bandwidth Management in Grids, IBM Developerworks.
8. Cafaro, M., 2005, Southern Partnership for Advanced Computational Infrastructure.
9. Coomer, J., 2002, Introduction To GRID Computing I, Sun Blueprints.
10. Coomer, J., 2002, Introduction To GRID Computing II, Sun Blueprints.
11. DeFanti, T., Stevens, R., 1999, Teleimmersion, s131-157, Morgan & Kaufmann Publishers.
12. Dr. Wallom, D., 2004, The UK e-Science Programme, Production Grids in the UK and University campus.

13. Ferriera, L. ve Ark., 2002 Introduction to GRID Computing With Globus, IBM Corporation Redbooks.

14. Foster, I., 2006, Designing and Building Parallel Programs.

15. Foster, I. ve Ark., 1998, Managing Security in Distributed Computations, Cluster Computing, 1, 95-107.

16. Foster, I. ve Ark., 1997, A Secure Communications Infrastructure for High-Performance Distributed Computing, 6th IEEE Symp. on High Performance Distributed Computing, pp. 125-136.

17. Foster, I. ve Ark., 1996, The Nexus Approach to Integrating Multithreading and Communication. Journal of Parallel and Distributed Computing, 37, 70-82.

18. Galli, D., 2000, Distrubuted Operating Systems Concept & Practices, Prentice & Hall.

19. Hassaine, O., 2004, SGE Integration with Globus.

20. Hassaine, O., 2004, Steps to Grid Adoption.

21. Hassaine, O. ve Ark., 2004, N1TM Grid Engine 6 Features and Capabilities.

22. Macit, İ., 2003, Grid Computing Systems.

23. Maui High Performance Computing Center, 2006, Parallel Programming Workshop.

24. Microsoft Corporation, 2008, www.microsoft.com.

25. Schulz, M., 2004, Experience Deploying the Large Hadron Collider Computing Grid.

26. Srivastava, N., 2004, HP Grid Dissected.

27. Srivastava, N., 2004, HP Services for Grid Technology.

28. Şenyurt, B., 2008, www.csharpnedir.com.

29. Walker, M.A., 2005, HP and Grid.

30. Welch, V. ve Ark., 2003, Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press.