DAĞITIK PARALEL SİSTEMLERDE PERFORMANS ANALİZİ

Soydan SERTTAŞ

Yüksek Lisans Tezi

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Haziran - 2006

DAĞITIK PARALEL SİSTEMLERDE PERFORMANS ANALİZİ

Soydan SERTTAŞ

Dumlupınar Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Elektrik-Elektronik Mühendisliği Anabilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman : Yrd. Doç. Dr. Ahmet ÖZMEN

Haziran - 2006

KABUL ve ONAY SAYFASI

Soydan SERTTAŞ'ın YÜKSEK LİSANS tezi olarak hazırladığı "Dağıtık Paralel Sistemlerde Performans Analizi" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

29 / 06 /2006
(Sınav tarihi)

Üye : Prof.Dr. Abdurrahman KARAMANCIOĞLU

Üye : Yrd.Doç.Dr. Hamdi Melih SARAOĞLU

Üye : Yrd.Doç.Dr. Ahmet ÖZMEN

Fen Bilimleri Enstitüsün Yönetim Kurulu'nun ...../...../..... gün ve ..... sayılı kararıyla onaylanmıştır.

.....
Fen Bilimleri Enstitüsü Müdürü

DAĞITIK PARALEL SİSTEMLERDE PERFORMANS ANALİZİ

Soydan Serttaş
Elektrik-Elektronik Mühendisliği, Yüksek Lisans Tezi, 2006
Tez Danışmanı: Yrd.Doç.Dr. Ahmet ÖZMEN

ÖZET

Maliyetten ve zamandan tasarruf eden ve karmaşık hesaplamalar yapabilen bilgisayar sistemlerine olan ihtiyaç her geçen gün artmaktadır. Bu çalışmada, Linux işletim sistemli bilgisayarlar kullanılarak PVM (Parallel Virtual Machine) yardımıyla paralel bir makine oluşturulmuştur. Oluşturulan bilgisayar kümesi ile kullanıcı arasında grafiksel arayüz olarak XPVM kullanılmıştır. Bilgisayarlar arası iletişim Ethernet ile sağlanmıştır. Paralel sistemi oluşturan işlemcilerin sayısı, çözülmek istenen problemin boyutu ve iletişim elemanlarının özellikleri genel performansı etkilediğinden paralel sistemlerde performans analizi büyük önem arzetmektedir. PVM, farklı bilgisayarlarda çalışan işlemlerin birbirleri arasında bilgi transferini sağlayan ve bağlanabilen fonksiyonlardan oluşan bir kütüphanedir. Çalışmada paralel programlar, problem boyutu ve işlemci sayısı değiştirilerek çalıştırılmış, her bir durum için sistemin performansı ölçülmüştür. Daha sonra XPVM ile program aktivite grafiği elde edilmiş ve performans irdelemesi yapılmıştır. Bu şekilde problem boyutu-makine sayısı grafiği elde edilmiş ve buradan performansın en iyi olduğu noktalar bulunmuştur.

Anahtar Kelimeler : Dağıtık Paralel Sistemler, Paralel Sanal Makine, PVM, XPVM

PERFORMANS ANALYSIS OF DISTRIBUTED PARALLEL SYSTEMS

Soydan Serttaş
Electric&Electronic Engineering, M.S.Thesis, 2006
Thesis Supervisor:Asst.Prof.Dr. Ahmet ÖZMEN

SUMMARY

The demand to economic high performance computers that do complex computations very fast is increasing day by day. In this thesis, a parallel machine with the aid of PVM (Parallel Virtual Machine) is set by using computers which are installed Linux operating system. XPVM is used as an interface between the computers and the user. The communication between computers are provided by Ethernet. The general performance of the system is affected by the number of processors and the problem size. Therefore the performance analysis of parallel systems is very important. PVM is a library which has linkable functions to provide information transfer between processes. In this study, parallel programs have been run by changing the problem size and the number of processors. The system performance was observed for each condition. The program activity graph had been obtained, and the performance is examined. The optimum performance points were found in terms of the problem size and the number of processors by looking at the analysis results.

Keywords: Distributed Parallel Systems, Parallel Virtual Machine, PVM, XPVM

TEŞEKKÜR

Çalışmalarım esnasında bana yardımcı olan danışman hocam Yrd.Doç.Dr. Ahmet ÖZMEN'e, her zaman ve her konuda bana destek olan anneme, babama ve kardeşime, fikir alışverişinde bulunduğum Arş.Gör. Arif BAŞGÜMÜŞ, Arş.Gör. Bahadır HİÇDURMAZ ve Arş.Gör. Yıldıray ANAGÜN'e sonsuz teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÖZET iv

SUMMARY v

TEŞEKKÜR vi

ŞEKİLLER DİZİNİ ix

SİMGELER VE KISALTMALAR DİZİNİ xi

1. GİRİŞ 1

1.1. Paralellik ve Paralelliğe Olan İhtiyaç 1

1.2. PVM ve XPVM 3

1.3. Araştırma Hedefleri ve Tezin Yapısı 3

2. PARALEL BİLGİSAYAR MODELLERİ 5

2.1. İşlemcilerin Sınıflandırılması 5

2.1.1. Tekli Komut – Tekli Veri Kümesi (SISD) 5
2.1.2. Tekli Komut – Çoklu Veri Kümesi (SIMD) 6
2.1.3. Çoklu Komut – Tekli Veri Kümesi (MISD) 6
2.1.4. Çoklu Komut – Çoklu Veri Kümesi (MIMD) 6

2.2. Paralel İşlem 8

2.3. Paralel Mimariler 9

2.3.1. Çok İşlemcili Sistemler 9
2.3.1.1. UMA 10
2.3.1.2. NUMA 10
2.3.2. Çok Bilgisayarlı Sistemler 11
2.3.2.1. MPP 11
2.3.2.2. Kümeler 12

2.4. Ölçeklenebilirlik 12

2.4.1. Ölçeklenebilirliğin Kıstasları 12

2.4.1.1. Kaynak Ölçeklenebilirliği 12
2.4.1.2. Uygulama Ölçeklenebilirliği Yönünden 13
2.4.1.3. Teknoloji Ölçeklenebilirliği Yönünden 13

İÇİNDEKİLER (devam)

Sayfa

2.5. Performans …… 13

2.5.1. Enstrumantasyon ve Performans Ölçümü ……... 14

2.5.2. Performans Metrikleri ……. 15

3. PVM…… 17

3.1. PVM Yazılım Paketi ……. 17

3.2. PVM Tarihçesi …….. 19

3.3. Neden PVM?. …… 20

3.4. PVM Ortamında Paralel Program Geliştirme ……... 20

3.5. PVM İçin Grafiksel Arayüz (XPVM)…… 21

3.6. PVM Konsolu ……... 23

4. DENEYSEL ÇALIŞMA …… 25

4.1. Donanım …… 29

4.2. Yazılım ……. 30

4.3. Uygulama Programları.……. 31

5. SONUÇLAR VE ÖNERİLER …….. 43

KAYNAKLAR DİZİNİ ……. 45

EKLER

1. Farklı Matris Boyutu-İşlemci Sayısı Deneme Sonuçları
2. Farklı Problem Boyutu-İşlemci Sayısı Deneme Sonuçları

ŞEKİLLER DİZİNİ

Şekil		Sayfa
1.1.	Süper Bilgisayarların Performans Gelişimi ……	3
2.1.	İşlemci Mimarilerinin Sınıflandırılması……	5
2.2.	SISD ..……	5
2.3.	SIMD ……..……	6
2.4.	MIMD (Paylaşımlı Bellekli) ……	7
2.5.	MIMD (Dağınık Bellekli) ……	7
2.6.	Paralel Bilgisayar Tiplerinin Gruplandırılması ……	9
2.7.	SMP Sistemler …….	10
2.8.	NUMA Sistemler ……..	11
2.9.	Paralel Bilgisayarlar İçin Makromimari ……...	13
2.10.	Derlemenin Çeşitli Safhalarında Enstrumantasyon …. ……	15
3.1.	PVM Fiziksel Görünüm ……...	18
3.2.	PVM Mantıksal (Logical) Görünüm ……	18
3.3.	PVM Programı İçin Ana-Uydu Modeli ..…….	21
3.4.	XPVM Arayüzü ..…….	23
3.5.	Sanal Makineye Bilgisayar Ekleme …….	23
3.6.	Sanal Makine Dağılımı …….	23
4.1.	Beowulf Yapısı …. ……...	26
4.2.	PVM Programı Haberleşme Mekanızması ……...	28
4.3.	Oluşturulan PC Kümesi…….	30
4.4.	Matris boyutu 300 için işlemci sayısı-süre grafiği ..……	32
4.5.	Matris boyutu 600 için işlemci sayısı-süre grafiği ..……	32
4.6.	Matris boyutu 900 için işlemci sayısı-süre grafiği ..……	33
4.7.	Matris boyutu 1200 için işlemci sayısı-süre grafiği ……	33
4.8.	Matris boyutu 1500 için işlemci sayısı-süre grafiği ……	34
4.9.	Matris boyutu 1800 için işlemci sayısı-süre grafiği ……	34
4.10.	Matris boyutu 2100 için işlemci sayısı-süre grafiği ..……..…	35
4.11.	Matris boyutu 2400 için işlemci sayısı-süre grafiği ..……..…	35
4.12.	Matris boyutu 2700 için işlemci sayısı-süre grafiği ..……..……	36
4.13.	Matris boyutu 3000 için işlemci sayısı-süre grafiği ..……..……	36
4.14.	İşlemci sayısı-matris boyutu-icra süresi grafiği ..……	37

ŞEKİLLER DİZİNİ (devam)

Şekil		Sayfa
4.15.	Matris boyutu 2700 ve işlemci sayısı 9 için program aktivite grafiği……..	38
4.16.	Matris boyutu 900 ve işlemci sayısı 9 için program aktivite grafiği……	38
4.17.	Problem boyutu 1000 için işlemci sayısı-süre grafiği ……	39
4.18.	Problem boyutu 2000 için islemci sayisi-süre grafigi ……	39
4.19.	Problem boyutu 3000 için islemci sayisi-süre grafigi ……	40
4.20.	Problem boyutu 4000 için islemci sayisi-süre grafigi ……	40
4.21.	Problem boyutu 5000 için islemci sayisi-süre grafigi ……	41
4.22.	Problem boyutu 10000 için islemci sayisi-süre grafigi ……..	41
4.23.	İşlemci sayısı-problem boyutu-icra süresi grafiği.…… ..	42
4.24.	Problem boyutu 5000 ve işlemci sayısı 8 için program aktivite grafiği……...	42

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	Açıklama
COW	İş istasyonu kümeleri
CPU	Merkezi işlem birimi
HU	Hafıza ünitesi
İU	İşlem ünitesi
KU	Kontrol ünitesi
LAN	Yerel Alan Ağı
MIMD	Çoklu komut- çoklu veri kümesi
MISD	Çoklu komut- tekli veri kümesi
MPI	Mesaj geçiş arayüzü
MPP	Güçlü paralel işlemciler
NOW	İş istasyonu ağı
NUMA	Paylaşımlı bellek erişimi
PAG	Program aktivite grafiği
PVM	Paralel sanal makine
SIMD	Tekli komut- çoklu veri kümesi
SISD	Tekli komut- tekli veri kümesi
SMP	Simetrik çok-işlemciler
TID	Görev kimlik numarası
UMA	Düzenli bellek erişimi
XPVM	PVM garafiksel arayüzü

1. GİRİŞ

Birkaç yıl öncesine kadar sadece gelişmiş ülkelerde bulunan ve milyonlarca dolara malolan yüksek işlem gücüne sahip bilgisayarlar, donanım, yazılım ve algoritmaların hızlı değişimi sebebiyle geçerliliğini çabuk kaybetmeye başlamıştır. Bu nedenle bilişim teknolojisi, aynı işi görebilecek performans/fiyat oranı daha yüksek olan sistemlere doğru kaymıştır. Teknolojinin hızlı ilerleyişi gelişmiş hesaplama tekniklerinin kullanımını zorunlu kılmaktadır. Tek işlemcili bir sistemden yüksek performans elde etmek yerine, çok işlemcili paralel sistemler ortak bellekli ve dağıtık olarak gerçeklenerek daha düşük maliyetle yüksek performans elde edilebilir.

1.1. Paralellik ve Paralelliğe Olan İhtiyaç

Günümüzde teknoloji gücü tüm ülkeler için sosyal ve siyasi güç anlamına gelmektedir. Yüksek teknolojiye sahip ülkeler rekabet güçlerini arttırabilmekte ve her alanda daha da ilerlemektedirler. Bugün elektronik eşya tasarımı, otomotiv, iklim modelleme, ilaç üretimi ve savunma sanayine yönelik çalışmalar gibi birçok çalışma için işlem gücü yüksek bilgisayarlar kullanılmaktadır.

Her yıl yaklaşık 1000 hortum, 5000 sel baskını ve 10000 ağır fırtına, dünya çapında 15 milyar doları bulan maddi hasara neden olmaktadır. Bu yüzden gelecek on, hatta yüzyılı kapsayacak, güvenilir hava tahminlerinin önemi büyüktür. Bu kadar geniş zaman dilimlerine ilişkin hesaplamalar ve iklim simulasyonları da elbette pahalı ve yüksek performanslı süper bilgisayarlara ihtiyaç duyulmaktadır.

Proteinler canlılar için hayati öneme sahiptir. Proteinleri tanımlamak ve onların neden ve nasıl fonksiyon gösterdiklerini anlamak, ilaç sanayine ve diğer biyolojik gelişmelere kapı açacaktır. Cornel Theory Center'da (CTe) araştırmacılar tarıma büyük zarar veren Pseudomonas Syringae adlı bir bakteriyel bitki patojeni üzerine araştırma yapmak istemişlerdir. Böyle bir durumda verimli bir çapraz tür araştırması için bilinen tüm bakteri genomlarına ait bilgilerin bilgisayar belleğine yerleştirilmesi gereklidir. Bu yüzden araştırmacılar 130 milyon nükleotid bazlı bir ağaç bilgi sistemi oluşturmuşlardır. Bu ise 8 Gbyte sistem belleği gerektirmektedir, aynı zamanda bu kadar bilgi işleyebilmesi için çok güçlü sistemlere ihtiyaç vardır.

Barajlardaki çatlaklar, uçak gövdelerindeki yırtılmalar, makine dişlilerindeki bozulmalar gibi kritik öneme sahip malzeme bilimi problemleri geniş tabanlı sonlu elemanlar

simülasyonları ile çözülebilir. İnsan beyninin çalışma yönteminin modellenmesinden, maddelerin manyetik alanlarına atomik düzeyde müdahale edilebilmesine, deprem ve yer hareketlerinin tahmininden ekonomi alanındaki risk analizlerine kadar farklı alanlarda güçlü bilgisayarlara gereksinim duyulmaktadır.

Paralel bilgi işleme büyük bir problemi daha küçük bileşenlere bölerek çözme yöntemidir ve yukarıda anlatılan ve yüksek performans gerektiren yerlerde ekonomik çözümler sunarlar.

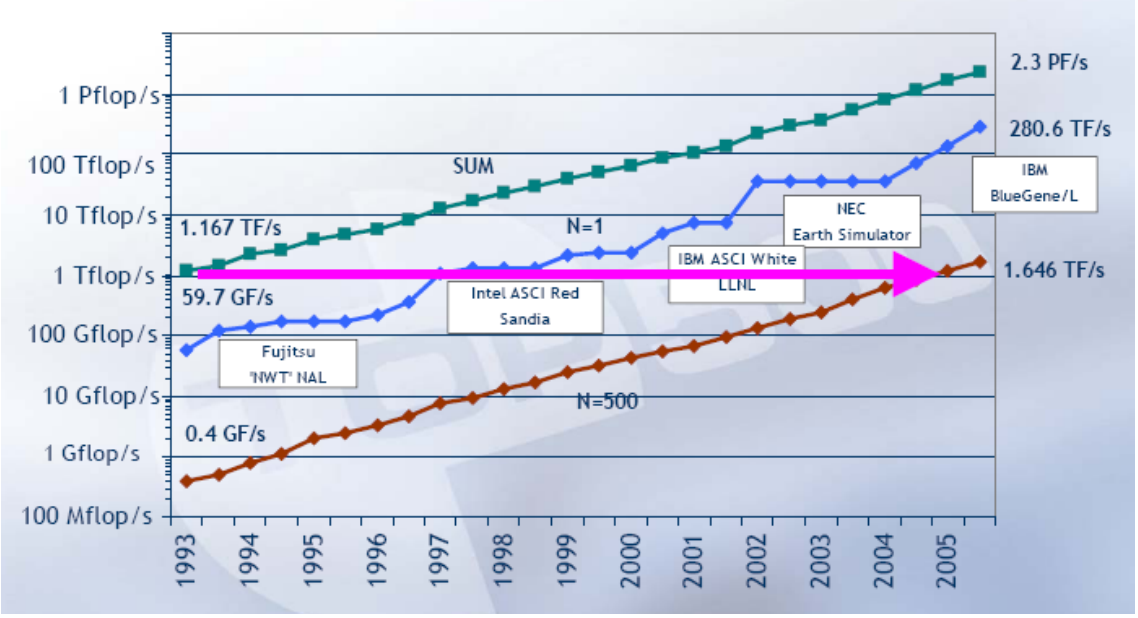
Paralel bilgisayar sistemleri ile ilgili çalışmalar SAGE (Semi-Automatic Ground Environment) ile başlar. SAGE, ilk küme sistemidir ve IBM tarafından Amerikan Hava Kuvvetlerinin hava erken uyarı sistemi için 1950'li yıllarda 12 milyar dolarlık bütçeyle yapılmıştır. Bilişim teknolojisindeki değişimler ile birlikte 1970'li yılların sonlarına doğru ilk kişisel bilgisayarlar (PC: Personel Computer) ile tanışılmıştır. Ethernet'in bilgisayar endüstrisine girişi ve çok kullanıcılı UNIX işletim sisteminin ortaya çıkışı paralel bilgisayar sistemlerinin gelişmesine yardımcı olmuştur.

İlk defa küme (cluster) tanımının kullanımı 1980'li yıllarda 100'den fazla iş istasyonunun DEC (Digital Equipment Corporation) tarafından bağlanmasıyla yapılmıştır. Aynı dönemde paralel işlem modelleri üzerinde çalışmalar başlamıştır. 1990'ların başında, değişik bilgisayarlarda çalışan işlemlerin (process) birbirleri arasında bilgi transferini sağlayan ve bağlantı (link) yapabilen fonksiyonlardan oluşan bir kütüphane olan PVM'in geliştirilmesiyle, 1992 yılında NASA Lewis Araştırma Merkezi'nde jet uçağı motorunun kalıcı durum davranışları simüle edilmiştir.

İlk BeowuIf sınıfı PC kümeleri, NASA Goddard Uzay Merkezi'nde Linux işletim sisteminin ilk sürümleri ile beraber PVM kullanılarak 16 adet Intel 100 MHz 80486 PC ile 1994'te gerçekleştirilmiştir. Bu sistem 2 adet 10Mbps Ethernet LAN içermekteydi. Daha sonra bu konu üzerinde çalışmalar hızlandı ve ilk MPI (Message Passing Interface) standartları Paralel Programlama Grubu tarafından ortaya atılmıştır. Beowulf Sistemi 1996 yılında 1 Gflop'luk performansı ve 50.000$'ın altında maliyeti ile California Teknik Enstitüsü, DOE (United States Department of Energy) Los Alamos Ulusal Laboratuarı ve NASA Jet İtici Laboratuarının ortak çalışmalarıyla kendini ispatlamıştır.

Günümüzde ise IBM'in Blue Gene/L adlı süperbilgisayarı, 131,072 işlemciyle ve 280.6 teraflop hızıyla dünyanın en güçlü bilgisayarıdır. ABD Enerji Bakanlığı'na bağlı Lawrence

Livermore National Laboratory adlı bilimsel kurumda nükleer araştırmalarda kullanılmaktadır. 1 teraflop, saniyede 1 trilyon işleme denk düşmektedir. 280.6 teraflopluk hızıyla Blue Gene/L saniyede 280 trilyon işlem yapabilmektedir.



Şekil 1.1 Süperbilgisayarların performans gelişimi [12]

1.2. PVM ve XPVM

PVM, heterojen yapıdaki bir grup bilgisayarın tek ve büyük bir paralel bilgisayar olarak kullanılabilmesini sağlayan bir yazılımdır. Bu yazılım sanal makine üzerinde otomatik olarak görevler (task) başlatmayı sağlayan fonksiyonlar içerir ve bu görevlerin birbiriyle iletişim kurmasını sağlar. Ağ, bilgisayar ve uygulama alanlarında heterojenliğe destek verir. Yazılan bir programın çalıştığı mimariye göre optimize edilebilmesine ve sanal makinenin çeşitli ağlarla bağlantısına izin verir.

XPVM, PVM programlarının icrası sırasında ağın durumunun ve görev takibinin gözlemlenmesine imkan vermekle birlikte, uygulamaların çalışan işlerinin birbirleriyle etkileşimi hakkında bilgilendirir.

1.3. Araştırma Hedefleri ve Tezin Yapısı

Bu çalışmanın amacı kişisel bilgisayarlar üzerine kurulacak Linux işletim sistemi ve PVM yazılımını kullanarak dağıtık paralel bir PC kümesi (cluster) oluşturmaktır. Daha sonra

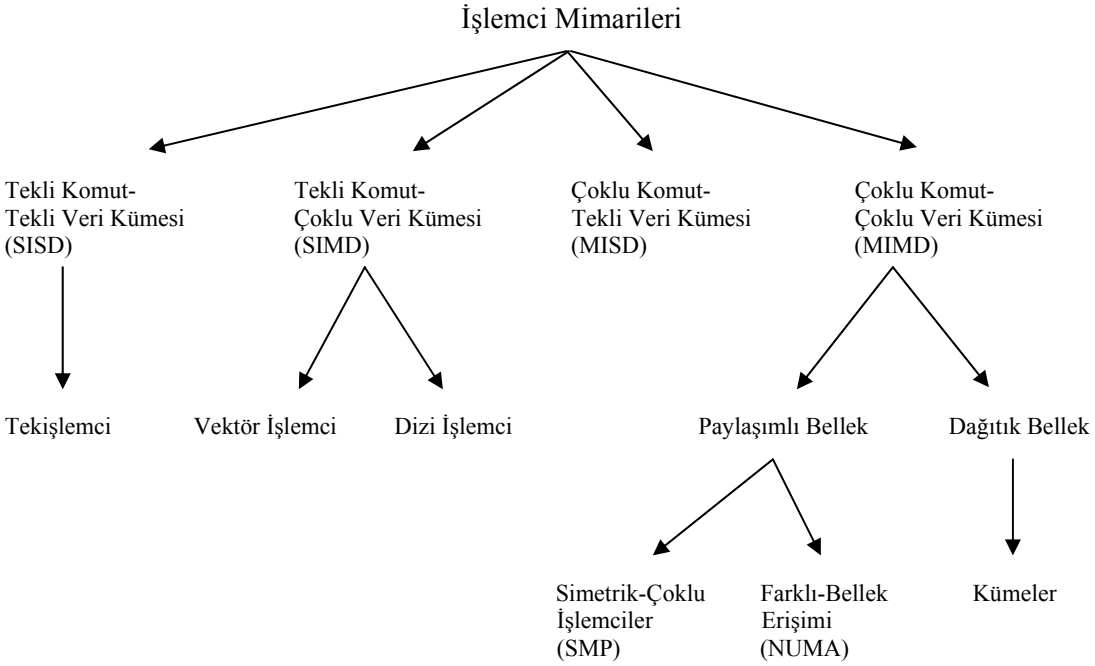
dağıtık paralel programların performanslarının incelenmesi ve problemlerin nasıl giderilebileceğinin araştırılmasıdır.

Bölüm 1'de verilen giriş bilgilerinden sonra, Bölüm 2'de bilgisayar mimarileri, işlemci sınıflandırmaları, paralel işlem ve paralel mimariler incelenmiştir. Bölüm 3'de PVM hakkında bilgi verilmiştir. Bölüm 4'de PVM uygulamalarına yer verilmiş, programların işlemci sayısı-problem boyutuna karşılık icra süreleri ölçülmüştür. Sonuçlar, Excel ile grafiğe dökülmüştür. Grafiklerden performanstaki değişim gözlenmiştir. Bölüm 5'de yapılan çalışmanın sonuçları yorumlanmış ve ileride yapılabilecek yeni çalışmalar ileri sürülmüştür.

2. PARALEL BİLGİSAYAR MODELLERİ

2.1. İşlemci Sınıflandırması

İlk defa Flynn tarafından yapılan ve hala geçerliliğini koruyan bilgisayar mimarisi sınıflandırmasına göre mimari şekil 2.1'de gösterildiği gibi dört farklı kategoride sınıflandırmıştır [3]. Bunlar tekli ya da çoklu veri ve komut kümesine göre değerlendirilir.



Şekil 2.1 İşlemci Mimarilerinin sınıflandırılması [3]

2.1.1. Tekli komut-tekli veri kümesi (SISD)

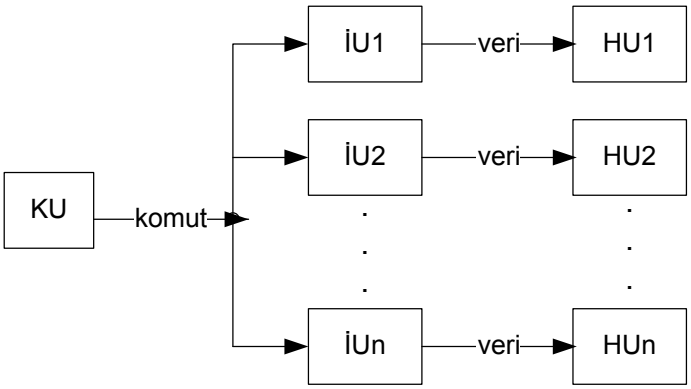
Bu modelde tek bir işlemci herhangi bir anda tek komut işler. Tekişlemciler bu gruba girer. Günümüz kişisel bilgisayarlarında performans için paralellik tercih edilmektedir. Bundan dolayı seri olarak düşünebileceğimiz SISD bilgisayarların üretimi günümüzde azalmıştır.

Şekil 2.2 SISD

Şekil 2.2 SISD mimarisini gösterir. Burada kontrol ünitesi (KU), işlem ünitesine (İU) gerekli komutları aktarır. İşlem ünitesi, hafıza ünitesi (HU) tarafından veri kümesi üzerinde işlem yapar.

2.1.2. Tekli komut-çoklu veri kümesi (SIMD)

Bu model tek bir makine komutunun farklı verilere eşzamanlı uygulanması ile elde edilir. Her işlem elemanı ilgili veri hafızasına sahiptir, böylece her komut farklı işlemciler tarafından farklı veri kümeleri üzerinde işlem yapar [3].



Şekil 2.3 SIMD

Şekil 2.3'de gösterilen SIMD mimarisinde birden fazla işlem elemanına komut kümesi taşıyan tek bir kontrol ünitesi vardır. Her işlem elemanı kendine ayrılan hafızayı kullanabileceği gibi paylaşımlı bellekte kullanılabilir.

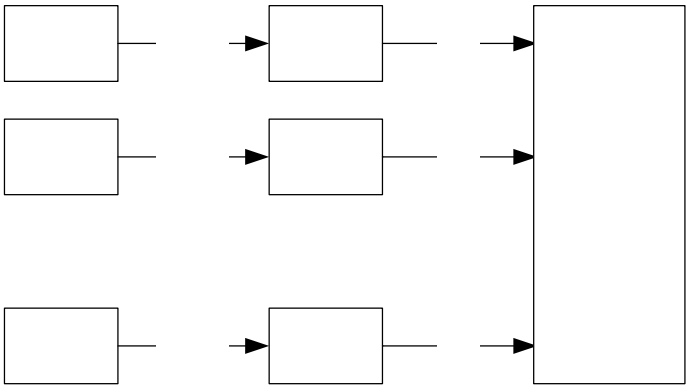
2.1.3. Çoklu komut-tekli veri kümesi (MISD)

Farklı işlemci ünitelerine ait farklı komutların aynı veri üzerinde operasyon yapmasıdır. Bu mimari hiçbir zaman gerçeklenmemiştir[3].

2.1.4. Çoklu komut-çoklu veri kümesi (MIMD)

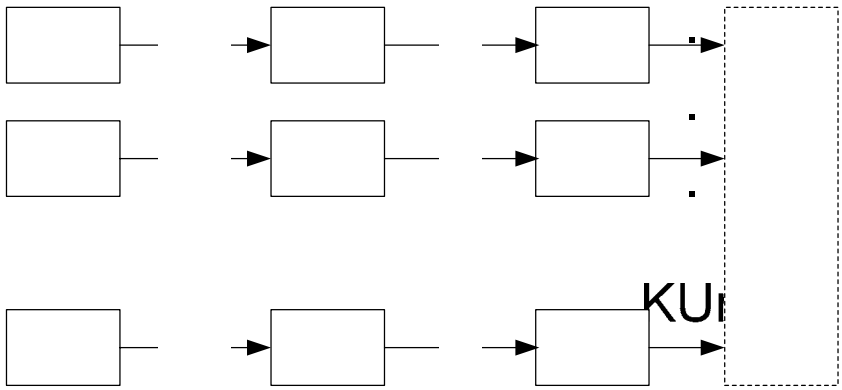
Birden fazla işlemcide eşzamanlı olarak çoklu komutlar çoklu veriye uygulanır. Simetrik işlemciler (SMP: Symmetric Multiprocessor), kümeler ve NUMA (Nonuniform

Memory Access) sistemler bu gruba girer.



Şekil 2.4 MIMD (Paylaşımlı Bellekli)

Şekil 2.4'de görülen MIMD mimarisinde birden fazla işlem ünitesine komut kümesini ileten ayrı ayrı kontrol üniteleri vardır. MIMD, şekil 2.4'de gösterildiği üzere paylaşımlı bellekli olabildiği gibi, şekil 2.5'de gösterildiği gibi dağıtık bellekli de olabilir.



Şekil 2.5 MIMD (Dağıtık Bellekli)

2.2. Paralel İşlem

Paralel işlem (parallel processing) büyük bir problemi paralel olarak çözülebilen daha küçük bileşen, görev veya hesaplamalara bölerek işleme yöntemidir ve son yıllarda özellikle yüksek performans gerektiren bilimsel hesaplamalarda (modelleme, simulasyon, analiz vb.) kullanılmaktadır. Güçlü Paralel İşlemciler (MPP : Massively Parallel Processors) ve kümeler (clusters), paralel işlemi geliştiren teknolojilerdir.

Bilgisayar biliminin paralel işlem alanına girersek ve buradaki teknolojileri kullanırsak daha fazla performans artışı sağlayabiliriz. Paralel işleme, bir uygulamanın çalıştırılması esnasında eşzamanlı işlemcilerin kullanılmasıdır. Bu yöntemde uygulama içerisinde paralel olarak yürütülebilecek kısımlar farklı işlemcilere yönlendirilirler. Böylece uygulama çok daha hızlı çalışır. Örnekleyecek olursak satranç oynayan bir uygulamayı düşünelim. Satranç oynu esnasındaki olası hamlelerin hesaplanması farklı işlemcilere yönlendirilirse daha başarılı bir oyun sergilenebilir. Başka bir örnek verecek olursak görüntü işleme alanının konusu olan, bir resim üzerindeki nesnelerin tespit edilmesi işi farklı CPU'lara yönlendirilebilirse daha kısa zamanda biter.

Paralel bilgisayar yapılarının getirdiği avantajların başında , klasik yöntemler ile ancak çok hızlı işlemciler kullanılarak yapılabilecek işlerin, makul hızlarda işlemciler kullanarak aynı performansı çok daha ucuza yakalaması gelir. Teorik olarak ulaşılabilecek en yüksek hız ise :

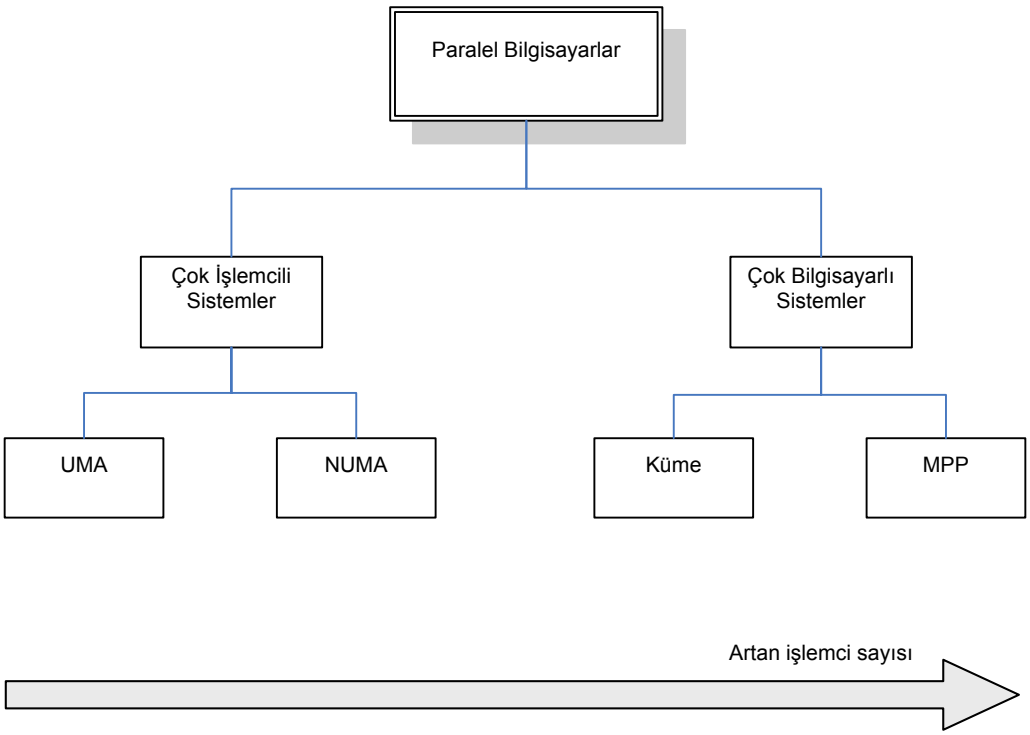
T(paralel) = T(seri) / N(işlemci)

formülüyle hesaplanabilir. Burada T(seri), klasik algoritma çalışma zamanını; T(paralel), paralel çalışma zamanını; N(işlemci) ise işlemci sayısını ifade etmektedir.

Yüksek işlem gücü, kararlı sistem yapısı ve esnek sistem mimarisi, paralel işlemin avantajları arasındadır. Paralel sistemlerin dezavantajlarına bakacak olursak, uygulamalar paralel işleme uygun olmalıdır. Bunun yanında ağ ortamında oluşabilecek sıkışıklık ve güvenlik sorunlarının aşılması gerekmektedir.

2.3. Paralel Mimariler

Eşzamanlı kullanılacak işlemciler hem aynı bilgisayarda hem de ayrı bilgisayarlarda olabilir. Şekil 2.6'da görüldüğü gibi bunlar çok işlemcili sistemler (multiprocessors) ve çok bilgisayarlı sistemlerdir (multicomputers). Aralarındaki fark ise bellek erişimidir. Çok işlemcili sistemlerde, CPU'lar aynı bellek bölgesine doğrudan erişir. Çok bilgisayarlı sistemlerde ise CPU'nun bellek bölgesi ona özeldir. Bu yüzden çok bilgisayarlı sistemlerin birbirleriyle iletişimi ağ üzerinden mesajlaşma yoluyla olur.



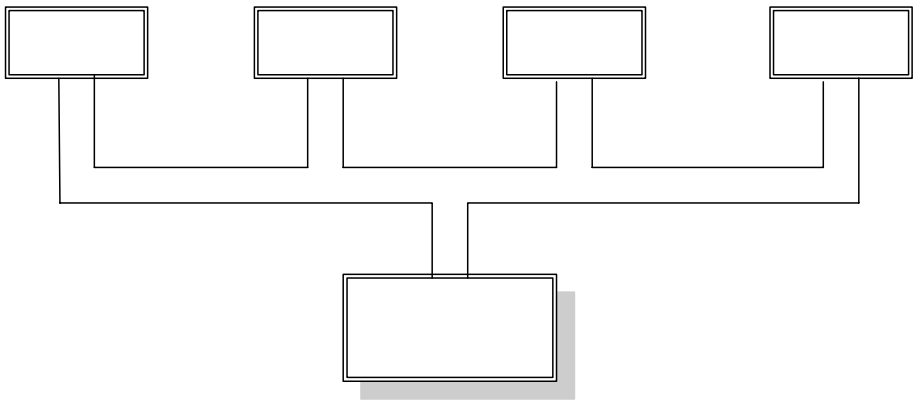
Şekil 2.6 Paralel Bilgisayar tiplerinin gruplandırılması [4].

2.3.1. Çok İşlemcili Sistemler

Çok işlemcili sistemler (multiprocessors), aynı bilgisayar içerisinde birden çok işlemcinin kullanılmasıyla oluşurlar. Çok işlemcili sistemlerde CPU'lar aynı bellek bölgesine erişir. Eğer bu erişim süresi belleğin her bölgesi için aynıysa çok işlemcili sistem UMA (Uniform Memory Access), farklıysa NUMA (Nonuniform Memory Access) olarak adlandırılır.

2.3.1.1. UMA

UMA sistemlerde tüm işlemciler için herhangi bir bellek bölgesine erişim süresi eşittir. En yaygın UMA sistemi Simetrik Çok işlemcili (Symmetric MultiProcessing -SMP) yapılardır (Şekil 2.7). Simetrik kavramını anlamak için önce asimetrik sistemlere bakmamız gerekir. Asimetrik sistemlerde işlemciler eşit durumda değillerdir. CPU'ların bir kısmı sadece girdi/çıktı işlemleri için kullanılırken diğer bir kısmı normal komutların icrası için kullanılır. Simetrik sistemlerdeyse işlemciler hem girdi/çıktı birimlerine hem de belleğe eşit şartlarda erişir. SMP sistemlerde çalışan işlem yükü eşit bir şekilde CPU'lar arasında dağıtılır. Aynı bellek bölgesinin kullanımı sırasında çakışma olmaması için, işletim sistemi tarafından her CPU'nun kullanacağı veri ve kodlar ayarlanmalıdır. Artan işlemci sayısı yüzünden bellek ve CPU'lar arasındaki yol (bus) darboğaz olmaya başlar ve bu sıkışıklık yüzünden SMP sistemlerin performansı düşer.

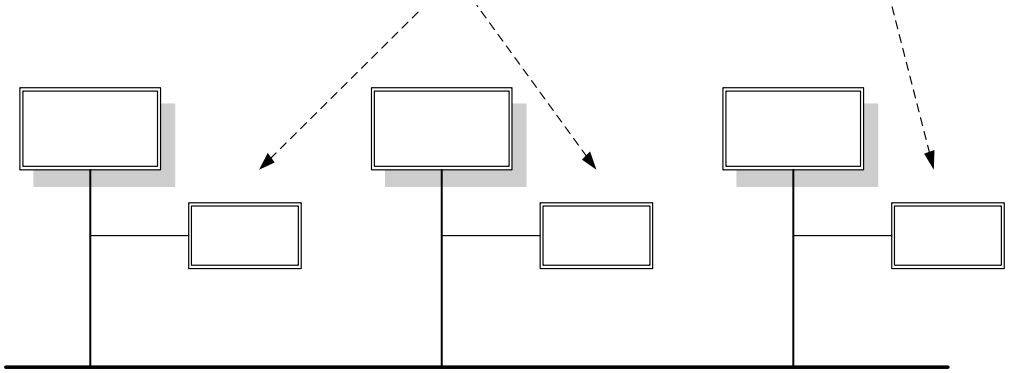


Şekil 2.7 SMP Sistemler

2.3.1.2. NUMA

Şekil 2.8'de artan CPU sayısı sorununa çözüm olan NUMA sistemler gösterilmektedir. NUMA (NonUniform Memory Access) sistemlerde her CPU için yerel bellek ayrılmıştır. İşlemciler kendi yerel belleklerine daha hızlı, diğer bellek bölgelerine ise daha yavaş erişirler. Dolayısıyla NUMA sistemlerde bellek erişimi, UMA'dan farklı olarak, değişik sürelerde gerçekleşebilir. Artan CPU sayısı NUMA sistemler kullanıldığında performans sıkıntısı yaratmayacaktır.

CPU CPU



Şekil 2.8 NUMA Sistemler

NUMA makinelerin bellek erişim hızlarının farklı olabilmesi, kararlı performans isteyen uygulamalar için istenmeyen bir durumdur. Bu yüzden 8 CPU'dan önce NUMA sistemlerle pek karşılaşamayız. Günümüzde bazı sunucu üretici firmalar, SMP ve NUMA karma sistemleri de geliştirmeye başlamışlardır. Bu sistemlerde 2 veya 4 CPU'dan oluşan SMP tabanlı modüller birbirlerine NUMA mantığıyla bağlanarak daha yüksek CPU sayılarına çıkılabilmektedir [13].

2.3.2. Çok Bilgisayarlı Sistemler

Çok bilgisayarlı sistemlerde, her bilgisayardaki bellek tamamen kendisine aittir. Bilgisayarların birbirleriyle veri alışverişi ancak ağ üzerinden mesajlaşma yoluyla olur. Aynı bilgisayar üzerinde, işlemci arttırımına gidilmesi güç ve pahalı olacağından, çok bilgisayarlı sistemler daha iyi bir seçenek olarak ortaya çıkar. Çok işlemcili bilgisayarların yıllar sonra da ulaşamayacağı işlem gücüne çok bilgisayarlı sistemler bugün ulaşmıştır. Şu an mevcut bulunan ve binlerce CPU'dan oluşan çok bilgisayarlı sistemlerin dezavantajı, programlanmasının zor oluşudur.

2.3.2.1 MPP

Çok bilgisayarlı sistemlerden olan MPP'ler çok büyük bir bilgi-işlem gücüne sahiptir. Bu sistemler bir oda içerisinde yüzlerce sayıda işlemci ve yüzlerce giga-byte'lık bellek içerirler. Küresel iklim modelleme ve deprem tahmini gibi büyük hesaplama gücü gerektiren bilimsel ve mühendislik hesaplamalarında kullanılır.

Maliyetlerinin yüksekliği MPP'ler için dezavantajdır. Aynı hesaplama gücünü daha düşük bir maliyetle elde etmek için, dağıtık bilgi-işlem kapsamındaki "küme bilgi işlem (cluster computing)" teknolojisi ciddi bir alternatiftir.

2.3.2.2 Kümeler

Ortak bir problemi çözmek amacıyla birbirinden bağımsız bir grup bilgisayarlardan oluşan ve birbirlerine bir iletişim ağıyla bağlanan yapıya "küme (cluster)" adı verilir. Ethernet benzeri ağ bağlantı elemanları ve standart PC/İş istasyonları kullanılarak oluşturulurlar. Maliyetleri MPP sistemlere göre daha düşüktür. Linux Beowulf küme sistemi bu grubun en bilinen örneklerinden biridir.

2.4. Ölçeklenebilirlik

Bir bilgisayar sistemi ve bunun tüm yazılım ve donanım kaynakları, performans artışı için kaynaklarını arttırabiliyorsa veya maliyeti aşağı çekmek için bu kaynaklar azaltılabiliyorsa ölçeklenebilirdir.

2.4.1. Ölçeklenebilirliğin Kıstasları

Sistemin kaynaklarının daraltılması veya genişletilmesinin (ölçeklenebilirlik) değişik açılarından çeşitli tanımları vardır:

2.4.1.1. Kaynak Ölçeklenebilirliği

Kaynak ölçeklenebilirliği makine boyutunu arttırarak yüksek performans elde etmektir. Makine boyutu; işlemci sayısı, depo birimlerine yatırım (cep bellek, sabit disk) veya yazılım geliştirilmesi gibi konulardır. Makine sayısını arttırmak bilgisayar sistemini geliştirmenin en kısa yoludur. Burada programlama ve haberleşme başlıca sorunlardır. Sisteme işlemci eklemek yerine belleği arttırmak veya yeni diskler eklemek sistemi geliştirmek için başka bir yoldur. Yazılım geliştirme yolları ise, daha fazla fonksiyonelliğe sahip işletim sistemi, daha iyi derleyici, daha verimli kütüphaneler ve daha kolay kullanımlı uygulama programlarıdır.

2.4.1.2. Uygulama Ölçeklenebilirliği Yönünden

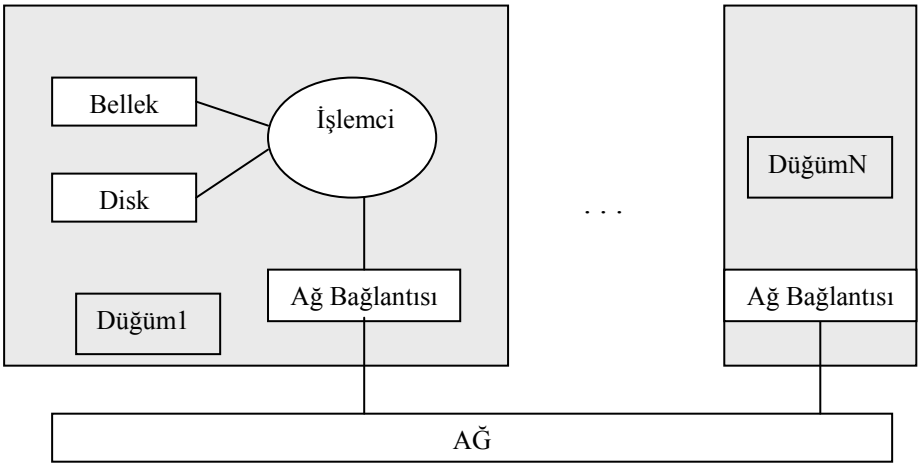
Uygulama programlarının ölçeklenebilir olması paralel bilgisayarların gücünü tam olarak ortaya çıkarmak için önemlidir. Bunun anlamı aynı program yükseltilmiş sistemde de orantısal olarak daha performanslı çalışabilmelidir. Makine boyutu ve problem boyutu önemli ölçütlerdir. Yeni işlemciler eklendiğinde sistemin performansının ne kadar geliştiğini ve daha büyük veri boyutunu ne kadar işleyebileceğini gösterir.

2.4.1.3. Teknoloji Ölçeklenebilirliği Yönünden

Sistemin teknolojik gelişmelere ne kadar uyumlu olduğunu gösterir. Daha hızlı bellek, daha hızlı işlemci ya da daha güçlü derleyiciler sistemi yükseltelebilir. Sistemi gelişmelere uyarladığımızda tekrar kullanılabilir olmalıdır. Bilgisayarların kapladıkları alan da ölçeklenebilirlikte önemli bir yer taşımaktadır. Sistemler bir odada veya birkaç binada olabilirler. Diğer bir özellik heterojenliktir. Değişik donanım ve yazılım bileşenleriyle ne kadar iyi yükseltgenebildiğini gösterir. Yazılım alanında bu duruma portatiflik (portability) denir.

2.5. Performans

Performans için iki temel ölçüt cevap süresi ve üretilen iş miktarıdır. Cevap süresi, uç kullanıcının performansıdır. Üretilen iş miktarıysa, fazla işlem gören disk, ağ kartı gibi bileşenlerin veya çok kullanıcılı sistemlerin performans ölçümünde kullanılır. Bilgisayar performansı, şehiriçi trafiğe benzer. Sıkışık yollar, darboğaz haline gelen bileşenler gibidir.



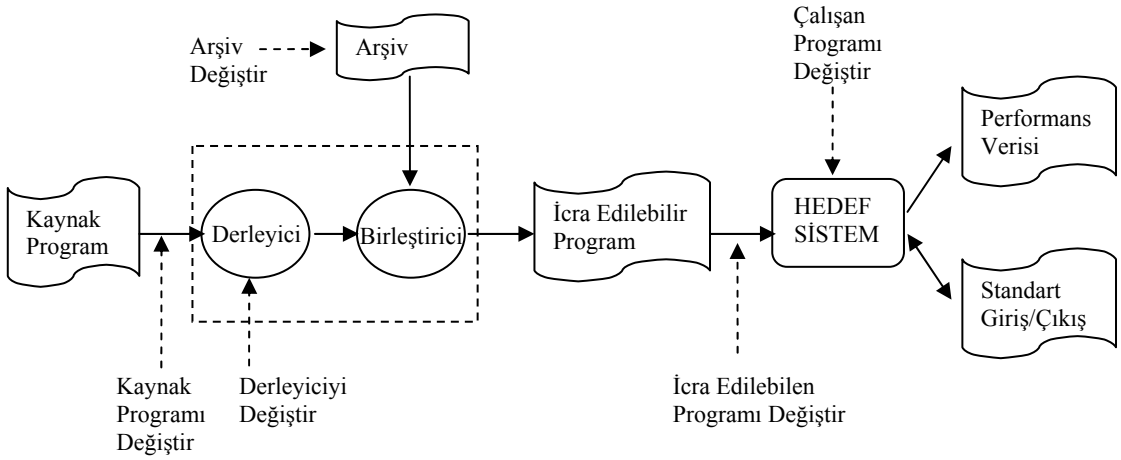
Şekil 2.9 Paralel Bilgisayarlar için Makromimari

Performans için Makromimari ile Mikromimari kavramları arasındaki farkı incelemek gerekir. Bilgisayar sistemlerinin tümleşik yapısına (Şekil 2.9) "Makromimari", işlemci çevresindeki birimlerin mimarisine de "Mikromimari" denir [10]. Kurulacak sistemlerin öncelikle mikromimari yönden hızlı olması gerekmektedir. Bu olay seri bağlanmış bir pil grubuna benzetirsek, seri olarak bağlı 4 pilin herbiri 1,5 volt gerilim üretsin. Seri bağlı bu sistemden 6 volt gerilim elde edilir. Makromimarinin önemi anlamak için örneğimizi açalım: Pillerin herbirinin birbiriyle bağlandığı noktada bir bağlantı direnci olsun ve 0,1 voltluk bir gerilim düşümü olsun. Sistemin verdiği toplam gerilim (1,5-0,1) * 4 = 5.6 volttur. Burada pillerin voltajı mikromimari; toplam voltajIar ve buradaki kayıplar ise makromimari ile temsil edilebilir.

2.5.1. Enstrumantasyon ve Performans Ölçümü

Performans verisi, çalışmakta olan bir programdan ancak enstrumantasyon yardımıyla elde edilebilir. Yazılım ağırlıklı gözlemleme sistemlerinde, yazılım parçalarından oluşan ilavelere "sensör" denilir [5]. Şekil 2.10'da görüldüğü gibi, sensörler, derleme işlemi öncesinde, sırasında veya sonrasında hedef sisteme ilave edilirler.

Performans verisi ilgilenilen olay olduğunda sensör tarafından oluşturulur. Sensör uyarı politikası olay-uyarıcılı veya zaman-uyarıcılı olabilir (bunlar sırası ile izleme ve örnekleme olarak bilinir) [6]. Olay-uyarıcılı politikada, ikincisinin aksine, performans verisi, olayın oluşumu ile eşzamanlı olarak üretilir. Her iki politikada da veri, üretildiğinde geçici bir bellekte olay-kaydı olarak saklanır. Genel olarak olay-kaydı, olayın olduğu zamanı belirten zaman-etiketi, olayın olduğu yeri bildiren yer-etiketi ve sistemin o anki durumuyla ilgili bazı ilave bilgileri içerir. Yazılım sensörleri orijinal sisteme sonradan ilave edildiğinden, sistemin karakterini değiştirecek etkiler yaratabilir, bu nedenle sensörler performans verisi toplanacak bir programa dikkatlice ilave edilmelidirler [9].



Şekil 2.10 Derlemenin çeşitli safhalarında enstrumantasyon [9]

2.5.2. Performans Metrikleri

Sistem özelliklerini ölçmek için kullanılan metrikler ölçmenin bir standardır. Ölçülen özelliğin nesnel bir şekilde değerlendirilmesine yardımcı olur. Metrikler, doğru kararlar alabilmek için bilgi sağlamanın yanı sıra diğer sistem konfigürasyonlarını karşılaştırmada kullanılabilirler.

Performans metriklerin kullanımıyla paralel sistemler daha verimli çalışacak şekilde ayarlanabilirler. Az sayıda giriş ve işlemci konfigürasyonuyla denenen bir paralel programın, farklı sayıda giriş büyüklüğü ve farklı sayıda işlemcili sistemlerde nasıl çalışacağı öngörülebilir ve performansının maksimum olacağı çalışma bölgesi bulunabilir.

Aşağıda seri veya paralel performans analizinde çok kullanılan bazı metriklerin tanımları ve nasıl kullanılabileceği anlatılmıştır:

Cevap süresi (Response time) : Bir programın başlangıcından o programa ait son işlemin(proses) bitişine kadar geçen süreye denir. Ölçülen bu süre programın gerçek performans ölçüsüdür.

Kapasite (Throughput) : Bir sistemin birim zamanda tamamladığı görevlerin(task) sayısına kapasite denir.

Kullanım oranı (Utilization) : Bir sistemin meşguliyetinin zamanın fonksiyonu olarak gösterilmesidir. Kısaca, meşgul geçen sürenin toplam süreye oranıdır.

Hızlanma (Speedup) : Paralel bir sistemde hızlanma, genellikle bir işlemci ile olan icra süresinin, P işlemcide ile olan icra süresine oranı olarak tanımlanmıştır. Kısaca hızlanma her işlemci için performans artışının bir ölçüsüdür.

Verimlilik (Efficiency) : Bir paralel sistemdeki verim, elde edilen hızlanmanın kullanılan işlemci sayısına oranı ile elde edilir. Ancak bu metrik, paralel bir programda gecikmelerin nerede olduğu ya da programın etkinliğini arttırmak için programda ne tür değişiklikler yapılması gerektiği hakkında bilgi vermez.

Aşağıda verilen diğer metrikler Program Aktivite Grafiği (PAG) yardımıyla hesaplanabilirler. PAG bir paralel program icrası sırasında oluşan hesaplama ve senkronizasyon bileşenlerinin zamana göre çizildiği bir grafiktir. PAG'nin bulunabilmesi için ilgili paralel programa sensörler ilave edilmeli ve zaman bilgisi toplanmalıdır.

Kritik Yol : Bir paralel programa ait kritik yol, hesaplama ve senkronizasyon temel parçalarından oluşan, icra süresi en uzun olan yoldur. En iyi performans artımı ancak bu parçaların optimizasyonu ile mümkündür.

Slack : Slack metriği kritik yoldaki parçaların optimizasyonu ile performansın ne kadar artabileceğini gösterir. Slack değeri, kritik yola en yakın ikinci yolun bulunmasıyla hesaplanır.

Lojik Sıfırlama : Lojik Sıfırlama Kritik Yol metriğinden türetilmiş metriktir. Lojik Sıfırlama metriği kullanılarak, Kritik Yol üzerindeki bir parçanın iyileştirilmesi sonucu performansın ne kadar iyileşeceği öngörülebilir . Bu, önce PAG üzerindeki ilgili parçanın icra süresinin sıfırlanması ve sonra eski ve yeni Kritik Yol arasındaki farkın hesaplanması ile bulunur.

Karşılaştırmalı çalışmalar göstermiştir ki kritik yol metriği paralel programların performans analizi için en değerli metriktir[9].

3. PVM

Bir küme, kendi yerel belleklerine sahip makinelerin bir araya getirilmesi ile oluşur. Dolayısıyla herhangi iki düğümün birbiriyle haberleşmesi ancak bu makineleri birbirine bağlayan ağ üzerinden olabilir. Bu nedenle küme mimarisinde iletişim işlevlerini yerine getirecek, yani bir düğümden gelen bir mesajı bir diğer düğüme aktaracak bir yazılım katmanına ihtiyaç vardır. Bu işlevi paralel iletişim kütüphaneleri (ya da iletişim arayüzleri) yerine getirir. PVM, ağ elemanları ile birbirine bağlanmış, işlemci ve işletim sistemi bakımından tamamen heterojen bir yapı oluşturan bilgisayarların, bir arada tek bir paralel makine gibi kullanılmasına imkan veren mesaj geçme yazılım paketidir [7].

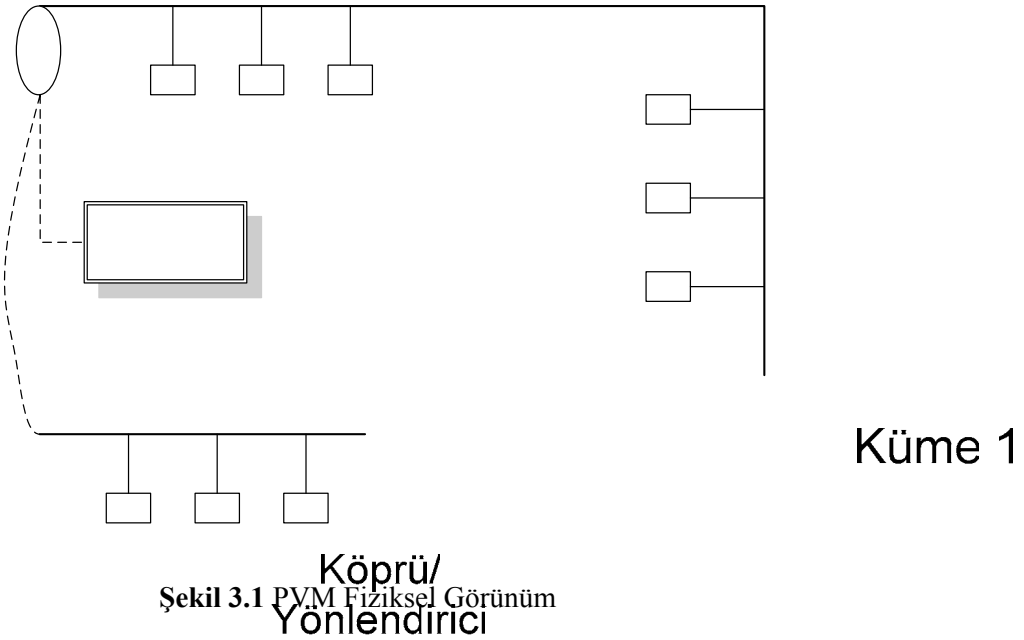
PVM, standart hale gelen “Message Passing” mekanizmasıyla çalışmaktadır. Farklı işlemciler arasındaki senkronizasyon, görev dağılımı ve veri iletimi mesaj göndererek sağlanır. Mesaj gönderme ve alma mekanizmaları hemen her ortam için zaten gerçeklenmiş olduğundan, PVM farklı işlemci ve işletim sistemleri ortamlarına taşınabilir bir yazılımdır.

3.1. PVM Yazılım Paketi

Ağa bağlı bilgisayarları bir araya getirip hesap yeteneği güçlü sanal paralel bir bilgisayar oluştururan PVM’de sistemi oluşturan makinelerin hepsi tek tip (homojen) olabileceği gibi farklı tiplerde (heterojen) mimarilere sahip olabilir. Ayrıca PVM, sanal makinenin çeşitli farklı ağlarla bağlantısına da izin verir. PVM sistemi aynı odada bilgisayarları bağlayan bir yerel ağ (LAN) kadar küçük veya dünya çapında bilgisayarları bağlayan İnternet kadar büyük olabilir. PVM, sanal makine üzerinde otomatik olarak görevler (task) başlatmayı sağlayan fonksiyonlar sunduğu gibi bu görevlerin birbiriyle iletişim kurmasını ve senkronizasyonlarını da sağlar. Merkezi bir kontrol altında farklı bilgisayar mimarilerini bir araya getirmeye yönelik bu kabiliyet sayesinde PVM kullanıcısı, bir problemi alt görevlere bölebilir ve her birini icra edilmek üzere en uygun işlemci mimarisine atayabilir. Görev boyutuyla ilgili bir sınırlama olmamakla birlikte iki bilgisayar arasında veri göndermek için gerekli zaman nedeniyle görevlerin boyutlarına göre PVM'in etkin kullanılması önerilir [7].

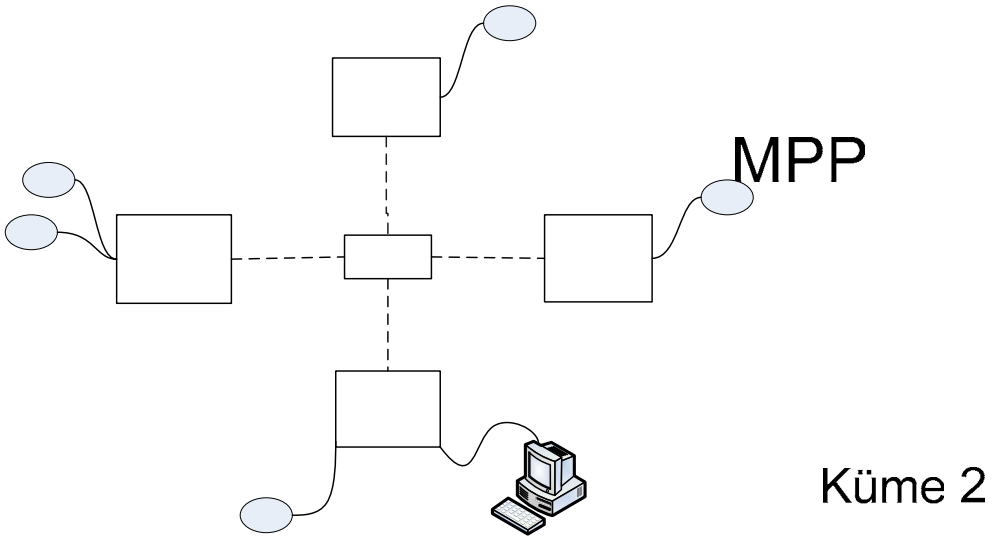
PVM sistemi iki parçadan oluşmaktadır. İlk parça pvmd3 adı verilen daemon süreçtir. Bu süreç sanal makineyi oluşturan tüm makinelerde çalışmaktadır. Bir kullanıcı bir PVM uygulamasını çalıştırmak istediğinde önce PVM'i (daemon süreci) çalıştırarak bir sanal makine yaratmalıdır. Daha sonra PVM uygulamasını komut satırından başlatabilir. Birden çok kullanıcı

aynı anda farklı konfigürasyonda sanal makineler oluşturabilir ve üzerlerinde aynı anda birden fazla PVM uygulaması çalıştırabilirler.



Şekil 3.1 PVM Fiziksel Görünüm

Şekil 3.1'de paralel sanal makine sisteminin bir küme kadar küçük veya dünya çapında bilgisayarları bağlayacak kadar büyük olabileceği görülmektedir. Şekil 3.2 de ise PVM sisteminin sanal makine mantığı görülmektedir.



Şekil 3.2 PVM Mantıksal(Logical) Görünüm

PVM sisteminin ikinci parçası PVM arabirim rutinlerini içeren kütüphanedir (libpvm3.a). Bu kütüphane kullanıcılar tarafından çağrılabilen mesaj iletme, yeni süreçler yaratma, görevlerin koordinasyonunu sağlama ve sanal makineyi düzenleme rutinlerini içerir. Tüm PVM uygulamaları, PVM ortamını kullanabilmek için bu kütüphane ile ilişkilendirilmiş olmalıdır. Uygulamalar Fortran veya C dilleriyle geliştirilebilmektedir.

3.2. PVM Tarihçesi

PVM, paralel bir bilgisayar gibi görünecek şekilde bir ağa bağlanan bilgisayarların heterojen olarak toplanmasını sağlayan bir yazılım paketidir. Oak Ridge Ulusal Laboratuarı'nda geliştirilen PVM, günümüzde endüstriyel ve deneysel amaçlar için kullanılmaktadır. PVM'in ilk sürümü 1989 yılında yapılmış fakat sadece Oak Ridge Laboratuarlarında kullanılmış ve daha sonra, Tennesse Üniversitesinin de katkılarıyla ikinci ve üçüncü sürümleri 1991 ve 1992 yılında yapılmıştır.

PVM ücretsiz dağıtılan, sabit diskte birkaç MByte yer tutan ve kolayca çalışır hale getirilebilen bir pakettir. PVM yazılımı ile çeşitli bilgisayarların işlemci ve belleklerinin ortak kullanımı sağlanabilmekte ve bu yolla, büyük hesaplama yeteneği gerektiren problemler çok fazla para ödemeden çözülebilmektedir. Yazılım paketi hemen hemen tüm işletim sistemlerinde ve bilgisayar mimarilerinde çalışabilmektedir. Bu şekilde PVM, kullanıcılarına, ellerindeki donanımı kullanarak en az maliyetle daha büyük problemleri çözebilecek yüksek performanslı bir bilgisayar oluşturmada yardımcı olmaktadır. PVM paralel programlama eğitiminde bir araç gibi kullanılmasının yanında, günümüzde bilim, sanayi ve tıp alanında da kullanılmaktadır. PVM, C ve Fortran dilinde paralel programlar yazılmasını desteklemektedir. Paralel program tasarımında, görevler en uygun mimariye atanarak performans iyileştirilebilmekte, zaman içinde değişen kaynaklar adapte edilebilmektedir.

3.3. Neden PVM?

PVM'nin genel özellikleri aşağıda sıralanmıştır:

- PVM'de sanal makine kavramı vardır. Yani PVM, heterojen bilgisayar sistemlerinin tek bir paralel sanal makine gibi görünmesini sağlar. PVM bütün mesaj yönlendirmelerini ve iş paylaşımlarını birbirleriyle uyumsuz bilgisayar mimarilerinden oluşan bilgisayar ağları boyunca ele alabilir. Kısacası, heterojen bilgisayarlardan oluşan NOW (Network of Workstations - İş İstasyonları Ağı)'ların veya MPP'lerin kullanıldığı paralel sistemlerin, tek bir sanal makine gibi görünmesini sağlar.
- PVM programları, birbirleriyle koordineli bir şekilde çalışabilir.
- PVM, kendi içinde yük dengelenmesi yapabilir. Mesela, paralel işletim sırasında en uygun düğümü, sizin için seçebilir ve ona görevlendirme yapabilir. Bir düğümün üzerindeki yükün fazla olduğu durumda, diğer düğümlere iş paylaşımı yapabilir.
- Sanal makineye yeni düğümler ekleme ya da çıkarma işlemlerini gerçekleştirir.
- PVM, C, C++ ve Fortran dillerinde kütüphanelere sahiptir.
- PVM, hata tolerasyonuna uyumludur. Sanal makine düğümlerdeki hataları otomatik olarak tespit edip gerekli ayarlamaları yapar.

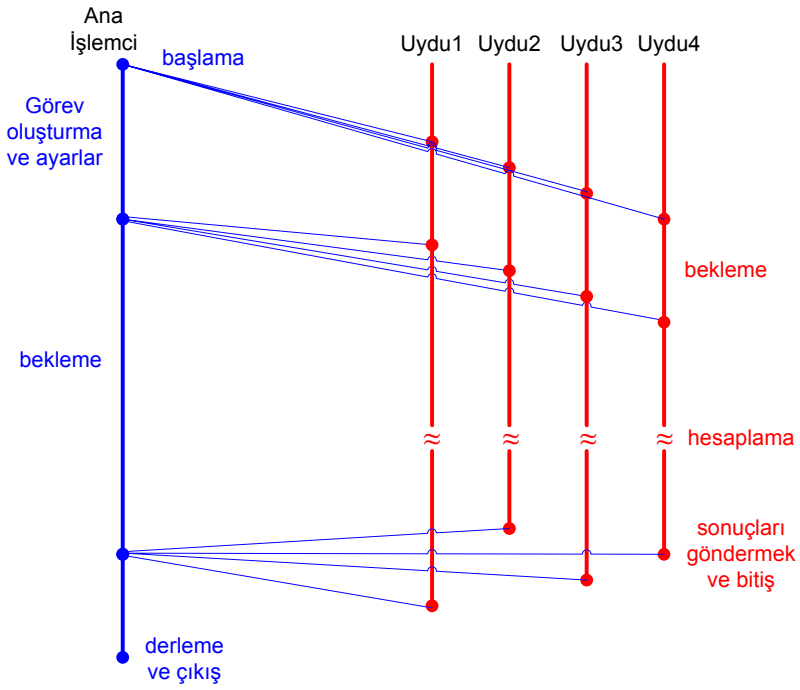
Ayrıca, PVM'nin ücretsiz olması, dünyada birçok üniversitede kullanılıyor olması PVM için önemli özelliklerdir.

3.4. PVM Ortamında Paralel Program Geliştirme

PVM ortamında çözülecek bir problem önce analiz edilmeli ve ortamı oluşturan hesaplama kaynakları arasında görev dağıtımı yapılmalıdır. Dengeli bir çalışma için dağıtılan görevlerin icra sürelerinin işlemciler arasında çok farklı olmamasına dikkat edilmelidir. Bunun için ortamı oluşturan işlemcilerin performansları da dikkate alınmalıdır. Örneğin yüksek performanslı bir işlemciye daha çok görev, eski ve düşük performanslı veya aşırı yüklü işlemciye ise daha az görev verilebilir. Ayrıca, işlemciler arası mesajlaşma, görev süreleri mümkün mertebe büyük seçilerek en aza indirilmelidir. Ayrıca sanal paralel ortam değiştiğinde (işlemci sayısı veya yüksek performanslı bir işlemci yerine düşük performanslı bir işlemci konması gibi), programın

tekrar elden geçmesi gerekebilir. Bu nedenle PVM gibi dağıtılmış paralel ortamlarda verimli çalışacak program yazmak oldukça zahmetli bir iştir [1].

PVM ortamında çalışan programlar genelde Ana-Uydu :"Master-Slave" modelindedir. Bu modelde ana işlemci uydu işlemcilere görev dağıtımı yapar, başlangıç verilerini ulaştırır, onları çalışmaya sevk eder, gerekiyorsa belirli aralıklarla senkronize eder ve en sonunda sonuçları uydu işlemcilerden toplayarak genel sonucu oluşturur. (Bkz. Şekil 3.3)



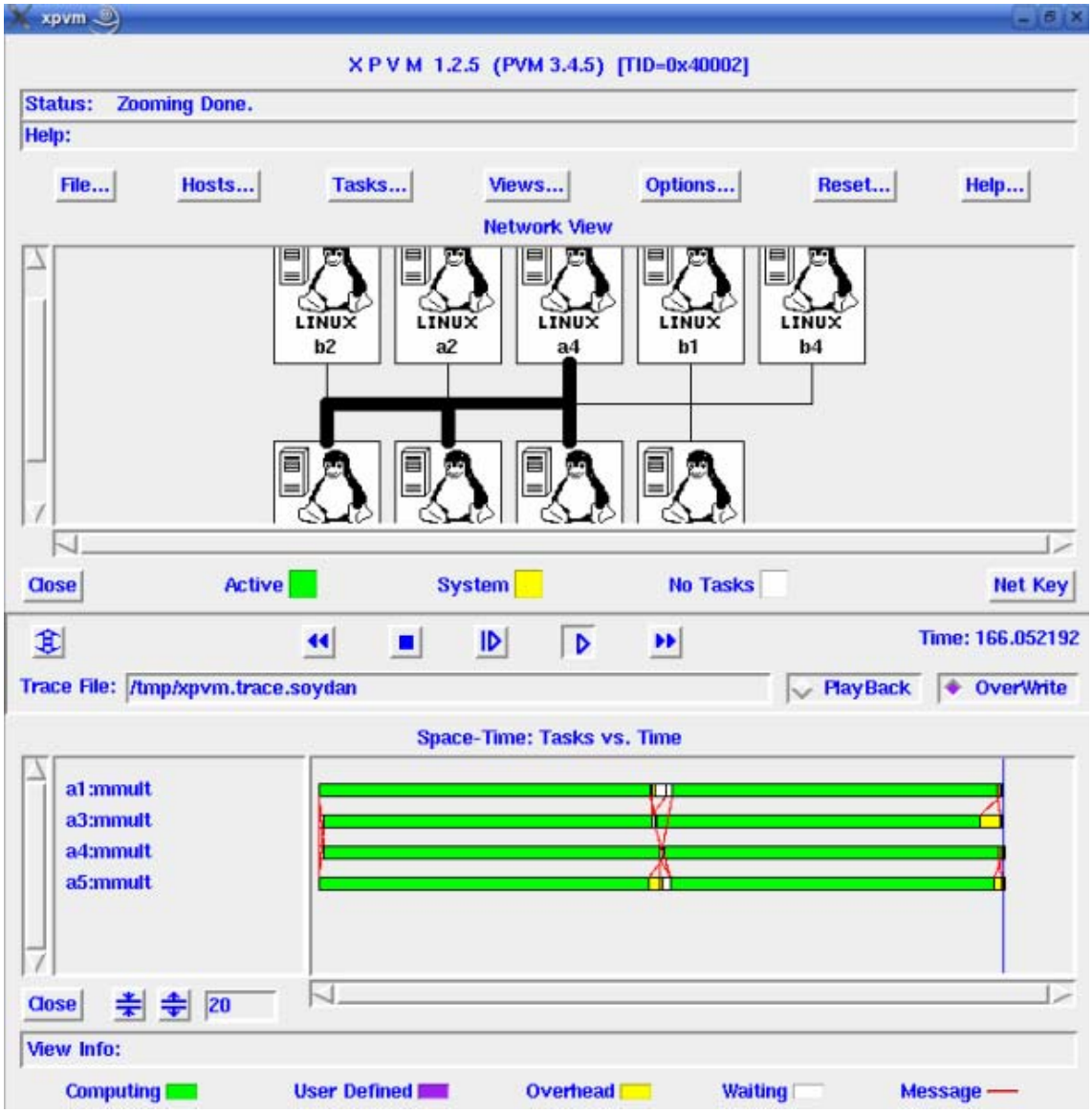
Şekil 3.3 PVM Programı için Ana-Uydu Modeli [1]

3.5. PVM için Grafiksel Arayüz (XPVM)

Paralel programların dağıtılmış kaynaklar üzerinde icrasının görüntülenmesi daha önce bahsedilen dağıtık sistemlerin programlama güçlüklerinden dolayı tek işlemcili sistemlere göre daha önemlidir. Tek işlemcili sistemlerde performans problemlerinin yerleri ve sebepleri kolayca tespit edilebilse de, paralel sistemlerde bu pek kolay değildir. Sorun, paralel programın eşzamanlı olarak farklı makinelerde icrasının sebep olduğu karmaşıklıktır. Pahalı hesaplama kaynaklarının verimli kullanılabilmesi için paralel programlar mutlaka performans monitörü ile izlenmelidir[5]. Bu amaçla görsel yazılım veya donanım araçlarına ihtiyaç vardır. PVM grubu

da paralel PVM uygulamalarının icrasını görsel olarak ekrana yansıtan XPVM'i geliştirmişlerdir.

XPVM, PVM konsol komutlarına grafiksel bir arayüz sağlamaktadır(Şekil 3.4). PVM uygulamalarının icrası sırasında ağın durumuna, görev takibi gibi bazı animasyonların gözlemlenmesine olanak sunar. Performansın belirlenmesi veya programın performans hatalarından arındırılması için bu görsel araçlar faydalıdır.



Şekil 3.4 XPVM arayüzü

3.6. PVM Konsolu

PVM konsolu uygulama programlarının düzenlenmesi ve çalıştırılması için kullanılır. Konsol ile görevlerin başlatılması ve durdurulması, bilgi ve hata mesajlarının alınması yapılabilir. Konsol, PVM'in çalıştığı herhangi bir makinede başlatılabilir veya durdurulabilir. PVM'i başlatmak için komut satırına 'pvm', sanal makinenin işi bittiğinde ise tüm PVM görevlerini sonlandırmak ve sanal makineyi kapatmak için 'halt' komutu girilmelidir.

Bir uygulamanın PVM'de çalıştırılması için uygulamanın tüm bilgisayarlar için derlenmesi gerekir. Derleme işleminden sonra sanal makine ayarlanır. Sanal makineye dahil edilecek bilgisayarların(host) listesini içeren bir dosya kullanıcının 'home' dizininde ve adı '.rhosts' olacak şeklinde oluşturulmalıdır. Konsoldan sanal makineye bilgisayar eklenip çıkarılabilir, aynı şekilde sanal makine dağılımı istenebilir ve çalışan görevlerin durumu kontrol edilebilir. Sistem içerisindeki bilgisayar düzenini görmek için 'conf' komutu kullanılır. Bu komutla birlikte bilgisayar ismi, PVM yönetici görev kimliği (DTID), mimari tipi ve bilgisayarlann hız oranı görülebilir.

Yukarıda değinilen komutların yanında; 'help' konsoldan yardım almak için, 'add' sanal makineye bilgisayar eklemek için, 'delete' sanal makineden makine çıkarmak için, 'ps' çalışan görevleri listelemek için, 'reset' bütün görevleri öldürmek için, 'quit' pvm konsolundan çıkmak için, ve 'halt' komutu tüm bilgisayardaki pvm servislerini durdurmak ve konsoldan çıkmak için kullanılır. Şekil 3.5'te, pvm programı başlatma ve sanal makineye makine eklemek için bir örnek verilmiştir. Burada makineye eklenen bilgisayarlar ve yönetici görev kimlikleri (DTID) görülmektedir. Bu örnekte b1 bilgisayarında pvmd başlatılamadığı için sanal makineye eklenemediği görülmektedir. Sanal makinede bulunan b2 bilgisayarı tekrar sisteme dahil edilmek istendiği için bu bilgisayar için de tekrarlama uyarısı vermektedir.

Şekil 3.6'te ise örnek bir sistemde bulunan bilgisayaraların 'conf' komutu ile listelenmesi gösterilmiştir. Bu listede HOST bilgisayar ismini, ARCH bilgisayar mimarisini ve SPEED bilgisayarın hız oranını göstermektedir

```
soydan@b2:~> pvm
pvm> add a1 a2 a3 a4 a5 b1 b2 b3 b4 b5
add a1 a2 a3 a4 a5 b1 b2 b3 b4 b5
8 successful
                    HOST     DTID
                      a1    80000
                      a2    c0000
                      a3   100000
                      a4   140000
                      a5   180000
                      b1 Can't start pvmd
                      b2 Duplicate host
                      b3   200000
                      b4   240000
                      b5   280000
```

Şekil 3.5 Sanal makineye bilgisayar ekleme

```
pvm> conf
conf
9 hosts, 1 data format
                    HOST     DTID     ARCH   SPEED       DSIG
                      b2    40000    LINUX    1000 0x00408841
                      a1    80000    LINUX    1000 0x00408841
                      a2    c0000    LINUX    1000 0x00408841
                      a3   100000    LINUX    1000 0x00408841
                      a4   140000    LINUX    1000 0x00408841
                      a5   180000    LINUX    1000 0x00408841
                      b3   2c0000    LINUX    1000 0x00408841
                      b4   300000    LINUX    1000 0x00408841
                      b5   340000    LINUX    1000 0x00408841
```

Şekil 3.6 Sanal makine dağılımı

Bir PVM uygulamasının çalıştırılması için uygulamanın tüm bilgisayarlarda 'aimk' komutu ile derlenmesi gerekir. Derlenen uygulamanın icra edilebilir dosyası pvm kütüphanesi altındaki bin/LINUX dizininde bulunmalıdır. Uygulamayı başlatmak için 'spawn' komutu kullanılır. PVM uygulaması sanal sistemdeki herhangi bir bilgisayarda başlatılabilir. Ana bilgisayar PVM görevlerini çalıştırır. Bu görevler içinde problemi çözmek üzere haberleşip hesaplama yapan birçok aktif görev vardır. Görev, iletişim ve hesaplama arasında gidip gelen bir kontroldür. Görevler, sistem tarafından verilen görev kimlik numaraları (TID) ile etkileşirler. Son olarak görevler işini bitirdiğinde PVM'den çıkarlar.

4. DENEYSEL ÇALIŞMA

Bu çalışmada PC sınıfı bilgisayarların, Ethernet kullanılarak birbirlerine bağlanmasıyla oluşturulan ve açık kaynak kodlu Linux Suse işletim sistemiyle çalışan PC kümeleri ele alınmaktadır. Paralel sistem P4 3.0 işlemcisi ve 256 MB belleği olan makinelerden farklı sayıda kullanılarak oluşturulmuştur.

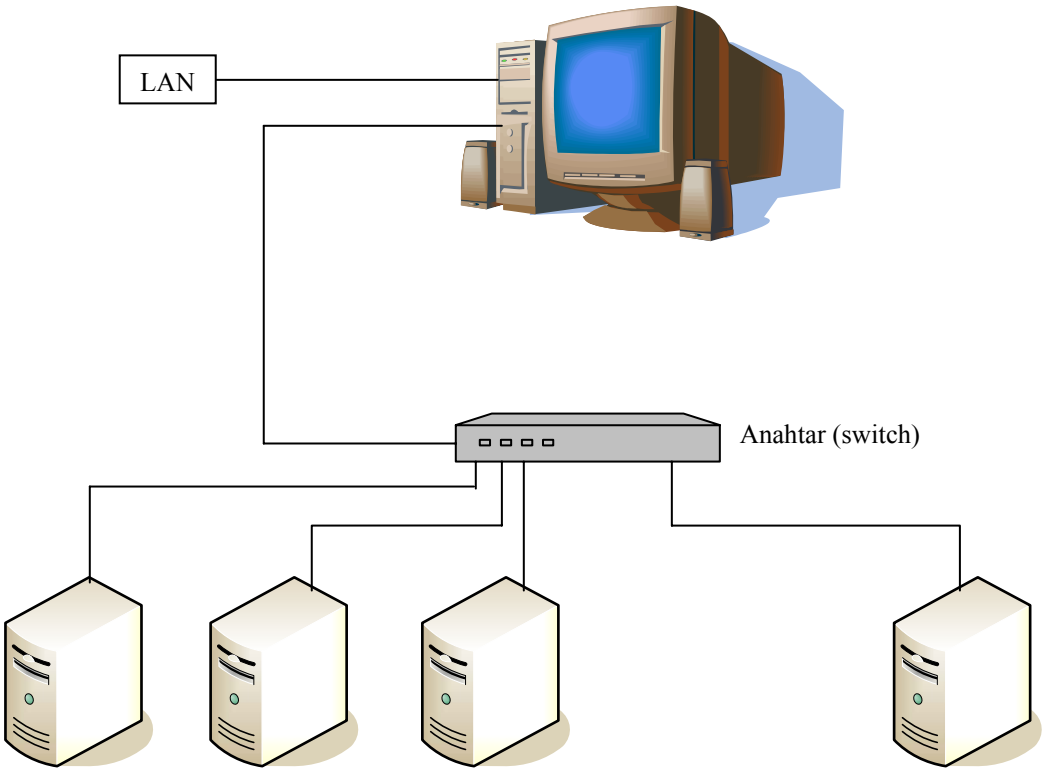
Bilgisayar kümeleri, Flynn'in sınıflandırmasına göre, Çoklu Komut Çoklu Veri (MIMD) sınıfında ve Dağıtık Bellek yapısında makinelerdir. Dağıtık bellek yapısında her makine kendi bellek ve ön-belleğine sahiptir dolayısıyla MIMD sınıfının diğer bir alt sınıfı olan Paylaşımlı Bellek Mimarisi'nin (Shared Memory Architecture) aksine Dağıtık Bellek mimarisinde paylaşılan ortak bir adres uzayı yoktur ve farklı makinelerin birbirleriyle iletişimi Mesaj İletim Kütüphaneleri (Message Passing Libraries) aracılığıyla gerçekleştirilmektedir (örneğin PVM, MPI vs.). Bilgisayar kümeleri, kümeyi oluşturan bilgisayarların ağ bağlantı şekillerine göre iki grupta toplanabilir. Bunlardan ilki iş istasyonu kümeleri, diğeri de Beowulf kümeleridir.

İş istasyonu kümeleri (COW : Cluster of Workstations), bir ağ ile birbirine bağlı, tek başına kullanılabilen iş istasyonu veya kişisel bilgisayarlardan meydana gelir. Genellikle bu kümeyi oluşturan tüm bilgisayarlara diğerlerinden bağımsız ve ayrık IP adresleri atanır dolayısıyla bu makinelerden küme dışı ağlara (örneğin İnternet) veya küme dışı ağlardan bu makinelere doğrudan erişmek mümkündür.

İş istasyonlarının paralel bilgi-işlem dışı çalışmalar için de kullanılabilmesi veya bağımsız olarak erişilebilir olmaları isteniyorsa, Beowulf yapısı daha uygun bir yapıdır. Beowulf kümeleri, bu ismi, ilk olarak NASA tarafından geliştirilen küme mimarisindeki süperbilgisayar sisteminden almaktadır. 1994 yılında Thomas Sterling ve Don Becker adlı iki araştırmacı, NASA için bir proje geliştirmişlerdir. Bu projede süper bilgisayar yapmak için bilgisayarları ethernet ağı üzerinden birleştirdikleri bir bilgisayar kümesi oluşturmuş, belli hesaplamaları yapmak için o an piyasada satılan parçalarla oluşturdukları 16 adet bilgisayardan oluşan bilgisayar kümesine Beowulf ismini vermişlerdir. Çok başarılı olan bu çalışma birçok araştırmacıyı ve firmayı bu konu üzerinde çalışmalar yapmaya yönlendirmiştir. İşlemci teknolojisinin ilerlemesi ve yüksek hızda bilgisayar ağlarının uygulanabilir maliyetlere inmesi, Linux işletim sistemi üzerinde çalışan Beowulf bilgisayar kümesinin yaygın olarak kullanılmasını sağlamıştır. NASA'dan sonra oluşturulan tüm Beowulf sistemler de bu ilk mimariyi temel almaktadır. Bu yapının COW'dan temel farkı, bu yapının bir ana (master)

bilgisayar ve değişen sayıda uydu (slave) bilgisayarlardan oluşmasıdır. Ana bilgisayar yine COW'dakine benzer biçimde bir iş istasyonu veya bir PC olabilmektedir. Ancak uydu bilgisayarlar genellikle klavye, fare, monitör gibi herhangi bir çevre birimi içermeyen PC sınıfı bilgisayarlardır. Dolayısıyla bu bilgisayarların kullanıcılar tarafından bir iş istasyonu gibi kullanılması mümkün değildir. Bu uydu makineler (node ya da düğüm adı da verilir) ana makine tarafından kendilerine verilen komut ya da işlemleri yerine getiren bir işlemci, bellek ve disk paketi olarak düşünülebilir [11].

Beowulf kümelerinde ana bilgisayar genellikle iki ağ arabirim kartı (Network Interface Card) içermektedir. Bunlardan ilki uydu bilgisayarlar ile olan bağlantısını sağlamakta diğeri ise ana bilgisayarı dış dünyaya bağlamaktadır. Uydu bilgisayarların dış dünya ile doğrudan bir bağlantıları yoktur ancak ana bilgisayar üzerinden bu makinelere erişmek mümkün olabilmektedir. Bu doğrultuda tüm uydu bilgisayarlar IP adreslerini ana bilgisayardan almakta ve bu adresler yerel adresler (örneğin 194.27.0.0 gibi) olmaktadır. Şekil 4.1'de tipik bir Beowulf yapısı gösterilmektedir.



Şekil 4.1 Beowulf Yapısı [5]

Kullanıcıların Beowulf kümesini, birden fazla bilgisayardan oluşan bir biçimde değil de tek bir bilgisayar gibi görmeleri bu yapının COW'dan bir diğer bir farkıdır. Kullanıcı ana makineden kodunu derlemekte ve eş zamanda diğer bilgisayarlar üzerinde çalıştırabilmektedir. Ana bilgisayar kümeye erişim için kullanıcı arayüzü görevi görmektedir.

Linux kümelerinin avantaj ve dezavantajlarına bakacak olursak [5]:

Avantajları :

- Güçlü paralel işlemcilere(MPP) göre çok daha düşük maliyetlidirler. MPP'ler milyonlarca dolara malolurlar, buna karşın birkaç yüzbin dolara büyük bir Linux kümesi oluşturulabilmektedir.

- Linux gibi ücretsiz bir işletim sistemi kullanmaktadır.

- Daha fazla işlem gücüne ihtiyaç duyulduğunda istenildiği kadar yeni bilgisayarı sisteme eklemek veya yeni ve daha hızlı modelleriyle değiştirmek mümkündür. Fakat bir MPP için bunlar sözkonusu değildir.

- Herhangi bir donanım üreticisine bağımlı olmayı gerektirmemektedir.

- Herhangi bir bilgisayarda meydana gelebilecek donanımsal bir arıza tüm kümenin çalışmasını etkilemeyecektir.

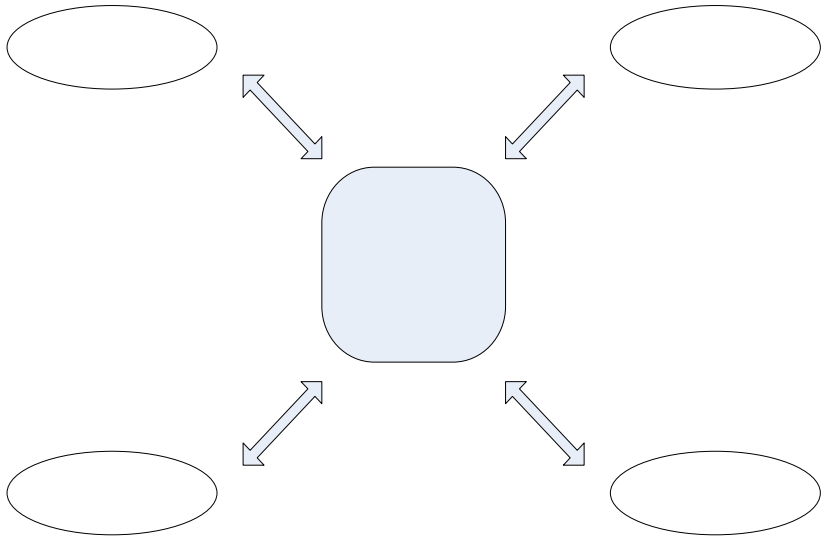
Dezavantajları :

- Çok sayıda bilgisayardan oluşmaları sebebiyle kümede oluşan sorunlarla uğraşmak zor olacaktır.

- Linux kümeleri fazla yer işgal etmektedir.

Bu çalışmada bir Linux kümesi 10 adet bilgisayardan oluşturulmuş ve performans testleri yapılmıştır. Performans ölçümlerinde sadece cevap süresi metriği kullanılmış, programlar konsoldan icra edilmiş ve cevap süresi ölçülmüştür. Deneyler farklı problem boyutu ve işlemci sayısı için yapılmıştır. İcra süreleri her bir parametre için ölçülmüş ve kaydedilmiştir.

Oluşturduğumuz PVM sistemini iki alt sisteme ayırabiliriz. Bunlardan ilki olan "Ana Program", bağlı makinelere veri ve komut dağıtımı yapan yazılımdır. İkincisi ise ana programdan gelen veri ve komutları toplayan ve kendi içinde çalıştırdığı programlara bunları dağıtan ve gerekli çıktıları ana yazılıma geri döndüren yazılımdır ki buna da "Uydu Program" denir. Şekil 4.2'de sistemin haberleşme mekanizması gösterilmiştir.



Şekil 4.2 PVM Programı Haberleşme Mekanizması

Ana programın görevlerini maddeleyecek olursak :

Uydu Program

1. Sistemde bulunan bilgisayarların ana programa tanıtılmasını sağlamak.
2. Ana program ile sistemdeki diğer makineler arasındaki iletişim ağını kurmak.
3. Sisteme tanıtılan makinelerden biri ya da bir kaçında uydu program çalışmıyorsa ya da bu makineler ile iletişim kurulamıyor ise ana programın sonlandırılmasını sağlamak.
4. Alt programların diğer makinelere gönderilmesi ve bir alt yordam gibi çalışmasını sağlayacak komutları göndermek.
5. Alt programlara veri girişi yapılabilmesi için gerekli veri yapılarının diğer bilgisayarlara dağıtılmasını sağlayacak komutları ve verileri göndermek.
6. Alt programlardan çıkacak olan çıktının ana programa gönderme işlemini sağlayacak komutları göndermek.

An

Uydu Program

Uydu programlar, alt programların işletilebildiği ve istenen çıktıların ana yazılıma gönderilebileceği ortamı sağlayan kısımdır. Karmaşık problemlerin çözümünde kendi altında bir çok programı eş zamanlı olarak çalıştırması ve eş zamanlı çalışan programlar arasındaki iletişimi ve zaman uyumunu sağlanması gerekir.

Bu dökümanın izleyen bölümlerinde, bilgisayar kümesinin oluşturulmasında kullanılan, donanım, ağ bağlantısı, işletim sistemi ve kullanılan diğer yazılımlara ilişkin bilgi verilmiş ayrıca gerçekleştirilen testlerle ilgili açıklama, sonuç ve yorumlara yer verilmiştir.

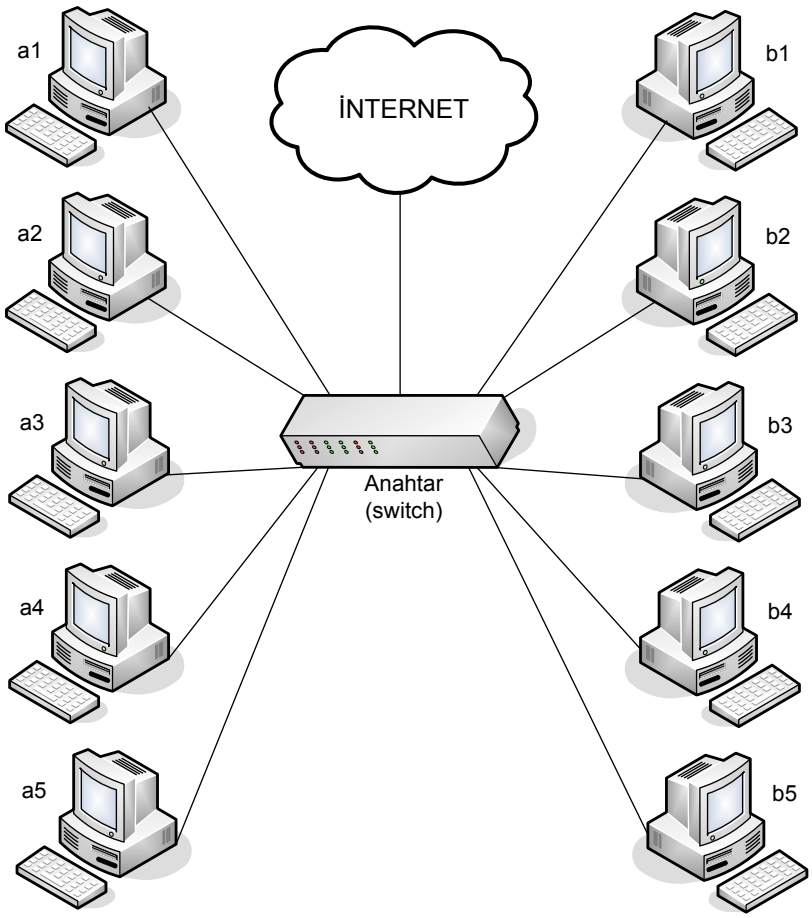
4.1. Donanım

Bilgisayar kümesinin oluşturulmasında kullanılan PC'lerin donanım özellikleri aşağıda verilmiştir :

- İntel Pentium IV 3.0 GHz işlemci
- İntel P4 i810 anakart
- 256 MB DDRAM
- 80 GB 7200 rpm Sabitdisk
- İntel(R) PRO/100 VE ağ bağlantısı

Arabağlantı ağı için 100Base-T (100 Mbps) Full Duplex Fast Ethernet ağ yapısı kullanılmıştır. Fast Ethernet yapısı ortalama bir performans vermekle beraber, donanım ve kablolama açısından düşük maliyetlidir. Bilgisayarların birbirleriyle bağlantısı 24 port kapasiteli, 10/100 Mbps destekli bir Fast Ethernet Switch (CNET CSH-2400) kullanılarak yapılmıştır. Şekil 4.3' de oluşturulan PC kümesinin yapısı gösterilmiştir.

Ethernet, dağınık tek kullanıcılı bilgisayarların yerel ağ olarak istenilen kaynakları kullanma ihtiyacından ortaya çıkmıştır. XEROX firması tarafından geliştirilen, OSI (International Standart Organization) tarafından IEEE 802.3 standardı olarak belirlenen bir standarttır. Yerel bir ağda bulunan bilgisayarların birbirleriyle haberleşmesini sağlar. Bu da hareketli kayıtlar olarak düşünülebileceğimiz 'Mesaj'larla olur. Mesajlar özel bir formatta düzenlenmiş bilgi topluluğudur. Hem gönderen hem de alan makineler bu mesajları yorumlayabilir. Bir kaç byte kadar kısa olabileceği gibi binlerce byte da olabilir. Hızı yaklaşık 10-100Mbps'dir.



Şekil 4.3 Oluşturulan PC Kümesi

4.2. Yazılım

Oluşturulan bilgisayar kümesinde yaygın olması ve kolay kullanımlı olmasından dolayı SuseLinux tercih edilmiştir. İletişim arayüzü olarak PVM 3.4 kullanılmış, program icraları XPVM ile gözlenmiştir. Uygulama programları, iki kare mattrisi çarpan 'Mmult' ve ısı difüzyon denklemi çözen 'Heat' isimli programlardır.

4.3. Uygulama Programları

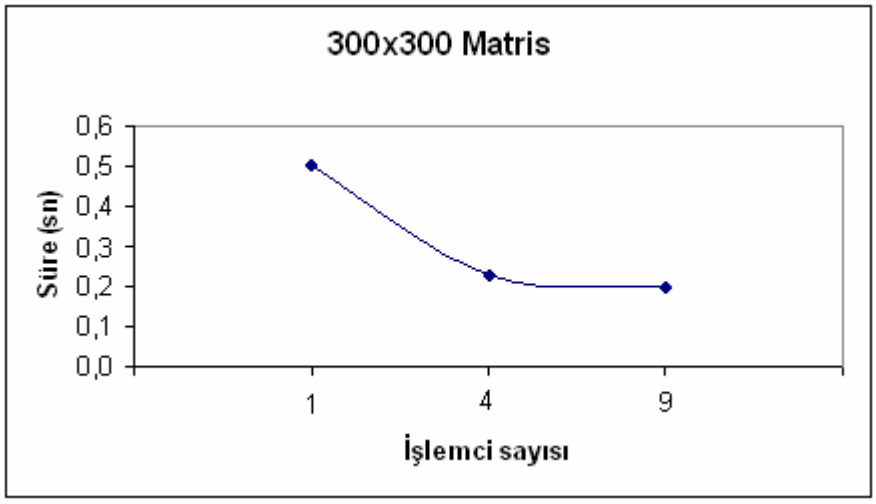
Performans deneyleri için, iki kare matrisi çarpan Mmult ve ısı difüzyonu denklemini PVM ortamında ana-uydu modeliyle çözen Heat adlı programlar seçilmiştir. Bu programlarda ana makine ilk veriyi hazırladıktan sonra uydu makinelerle haberleşir ve sonuçları bekler. Uygulama program kodları internetten indirilmiştir.

Matris çarpım programı, A, B ve C matrisleri kare matris olmak üzere C = AB işlemini Fox algoritmasıyla[15] gerçekleştirir. Bu algoritmaya göre, A ve B matrislerinin boyutları NxN, çarpım işlemini gerçekleştirecek makine sayısı P ise her bir işlemci matrisin $N/\sqrt{P}$ bloğunu işler. Dolayısıyla her makine C matrisinin bir bloğunu hesaplar.

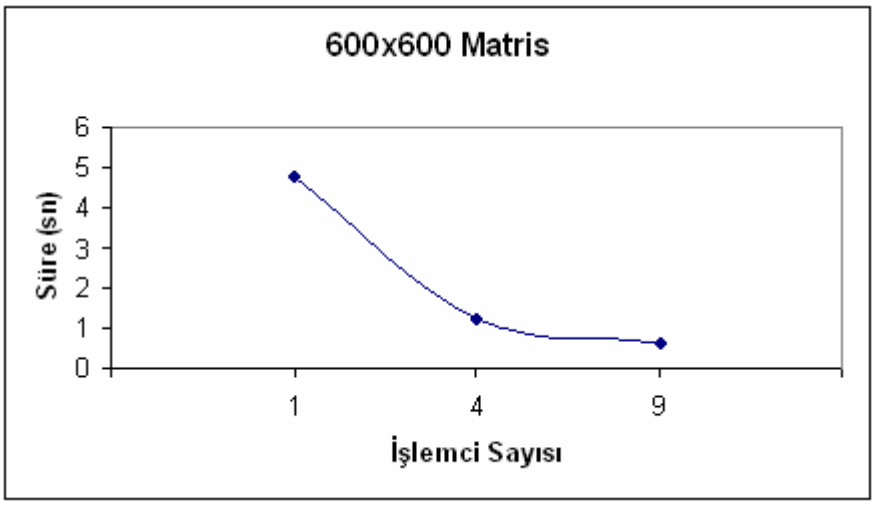
Heat programında ısı dağılımı ile ilgili hesaplamalar, uydu makinelerde çalışan programlar tarafından yapılmaktadır. Uydu programda var olan sonsuz bir döngü ile ilk veriler ana makineden alınır, sınır koşulları ise komşu makinelerle haberleşilerek belirlenir ve sonuçlar hesaplanır. Daha sonra bu kısmi sonuçlar ana makineye geri gönderilir ve teldeki ısı değişimi bulunur.

Deneyler sırasında, matris çarpım programı kodunda bir değişiklik yapılmamış ancak ısı difüzyon denklemi uygulama kodu içerisinde problem boyutu ve işlemci sayısı değişkenleri değiştirilerek her konfigürasyon için derlenmiştir. Her iki program için, konfigürasyonlara göre gerekli sayıda işlemci paralel ortama dahil edilmiştir. Konfigürasyon denemelerinde elde edilen veriler Ek 1 ve Ek 2'de verilmiştir.

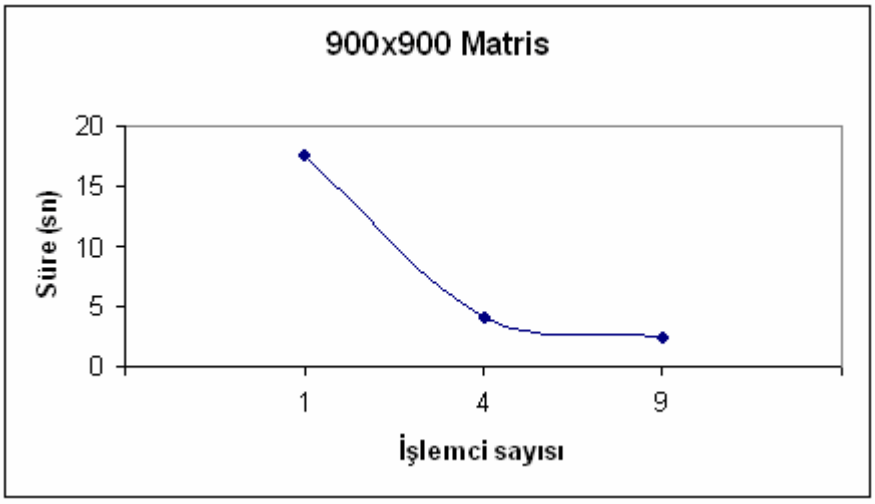
Matris çarpım programı için, farklı matris boyutları ve işlemci sayısı-icra süresi grafikleri şekil 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12 ve 4.13'de gösterilmiştir. Bu grafiklerde görüldüğü üzere artan işlemci sayısı programın icra süresini beklenildiği gibi düşürmüştür. Elde edilen eğriler lineer olmayıp ters logaritmik eğrilerdir.



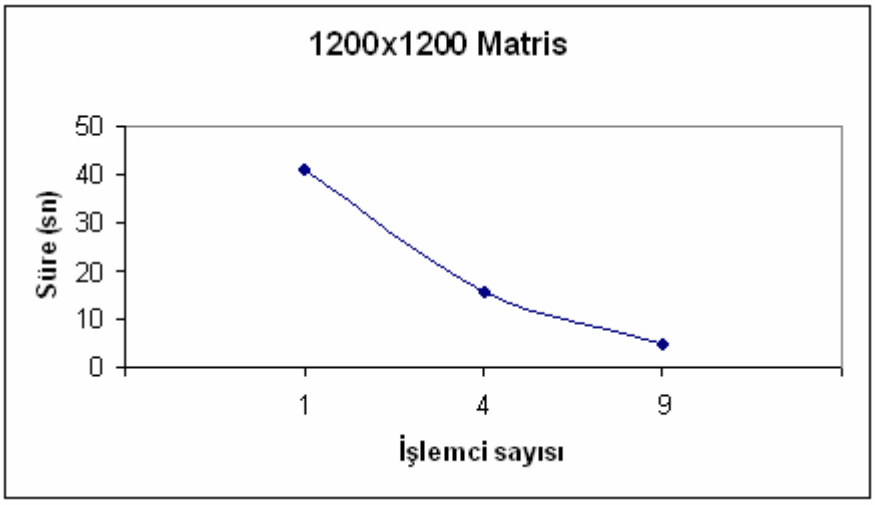
Şekil 4.4 Matris boyutu 300 için işlemci sayısı – süre grafiği



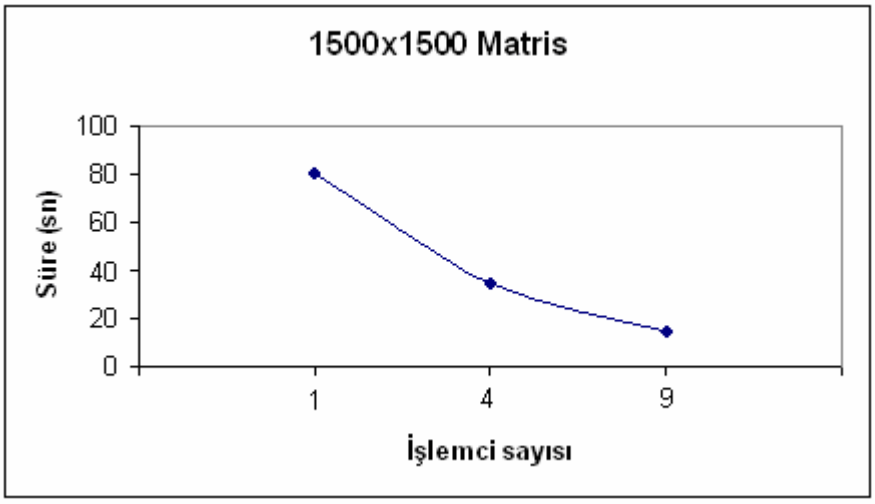
Şekil 4.5 Matris boyutu 600 için işlemci sayısı – süre grafiği



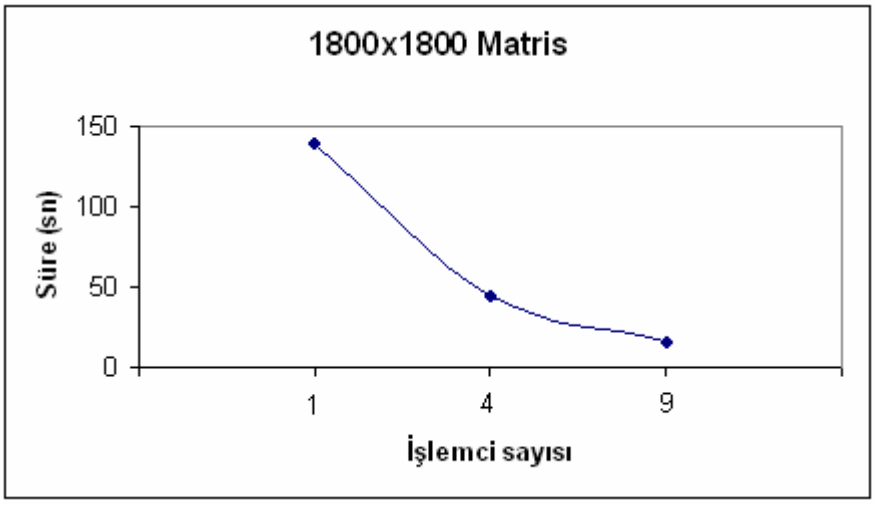
Şekil 4.6 Matris boyutu 900 için işlemci sayısı – süre grafiği



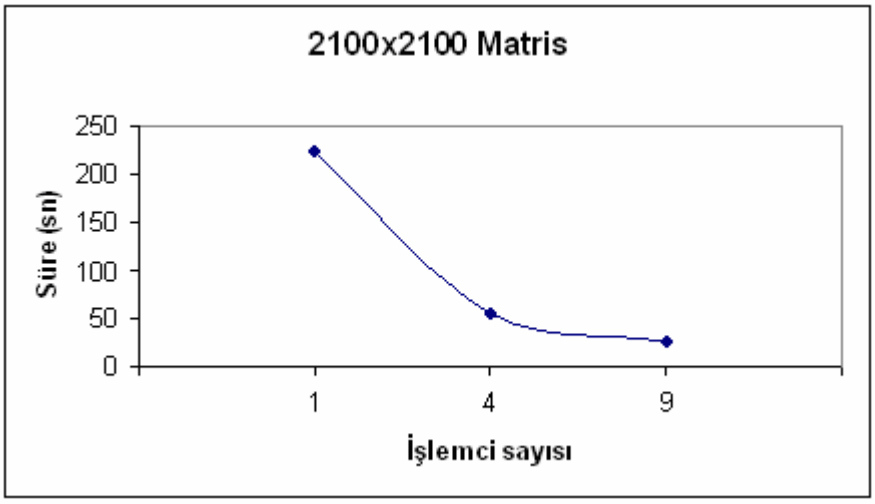
Şekil 4.7 Matris boyutu 1200 için işlemci sayısı – süre grafiği



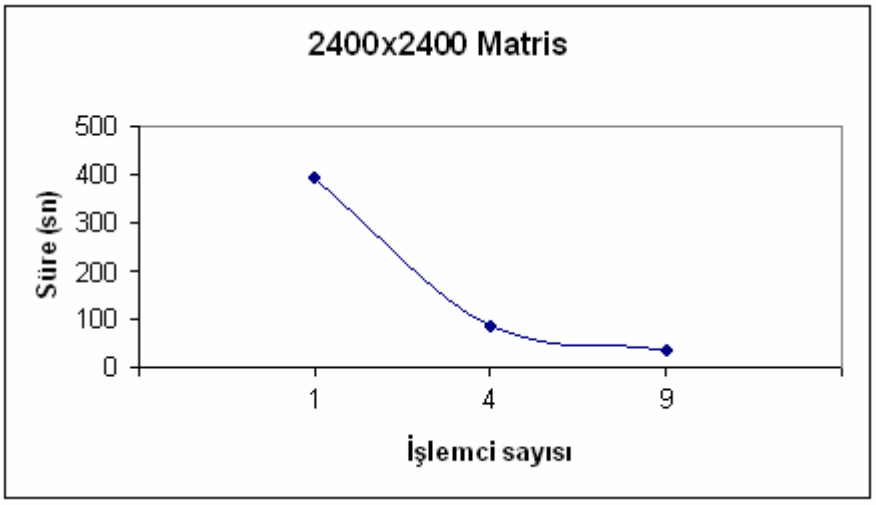
Şekil 4.8 Matris boyutu 1500 için işlemci sayısı – süre grafiği



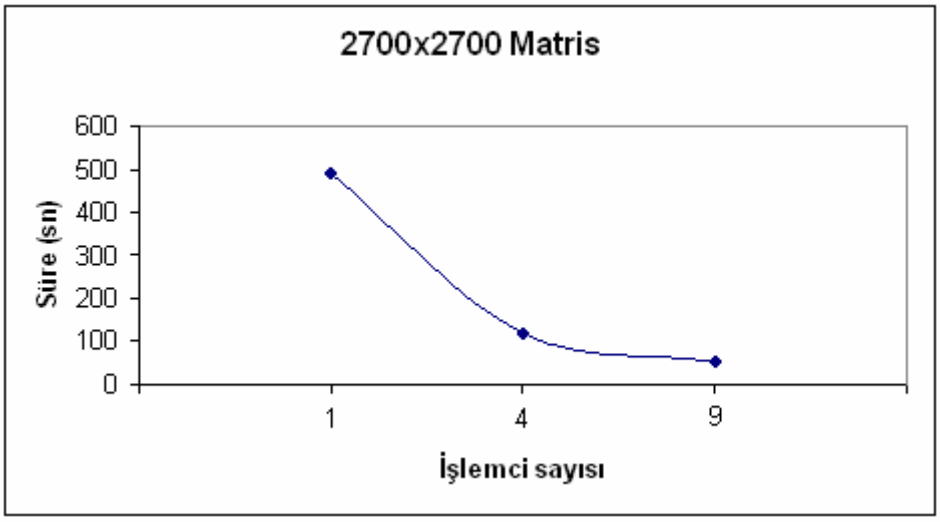
Şekil 4.9 Matris boyutu 1800 için işlemci sayısı – süre grafiği



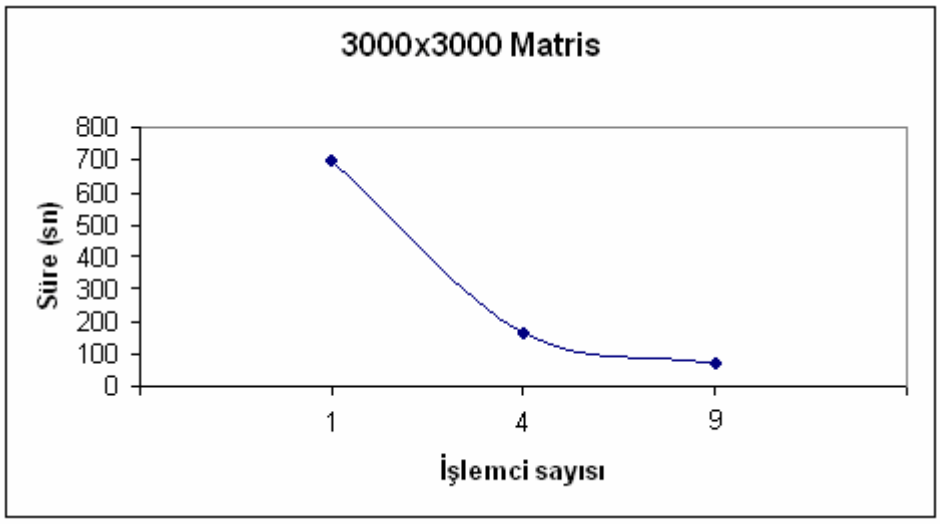
Şekil 4.10 Matris boyutu 2100 için işlemci sayısı – süre grafiği



Şekil 4.11 Matris boyutu 2400 için işlemci sayısı – süre grafiği

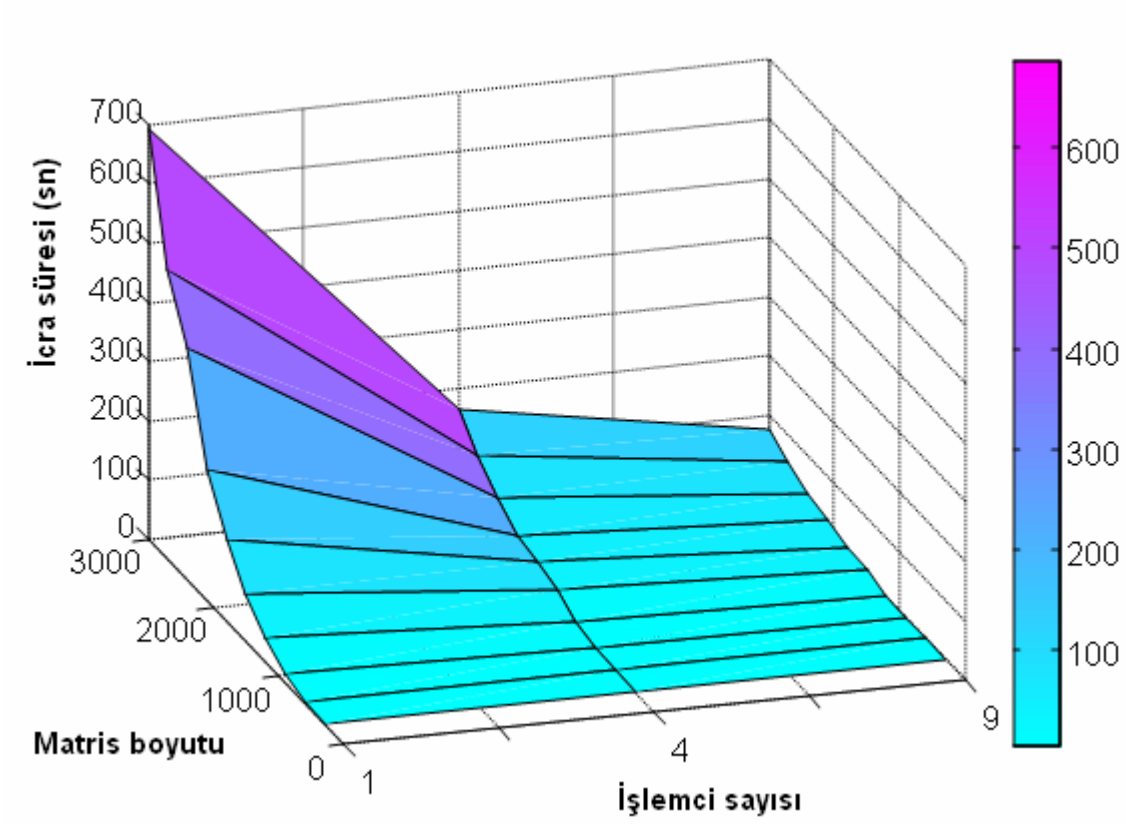


Şekil 4.12 Matris boyutu 2700 için işlemci sayısı – süre grafiği

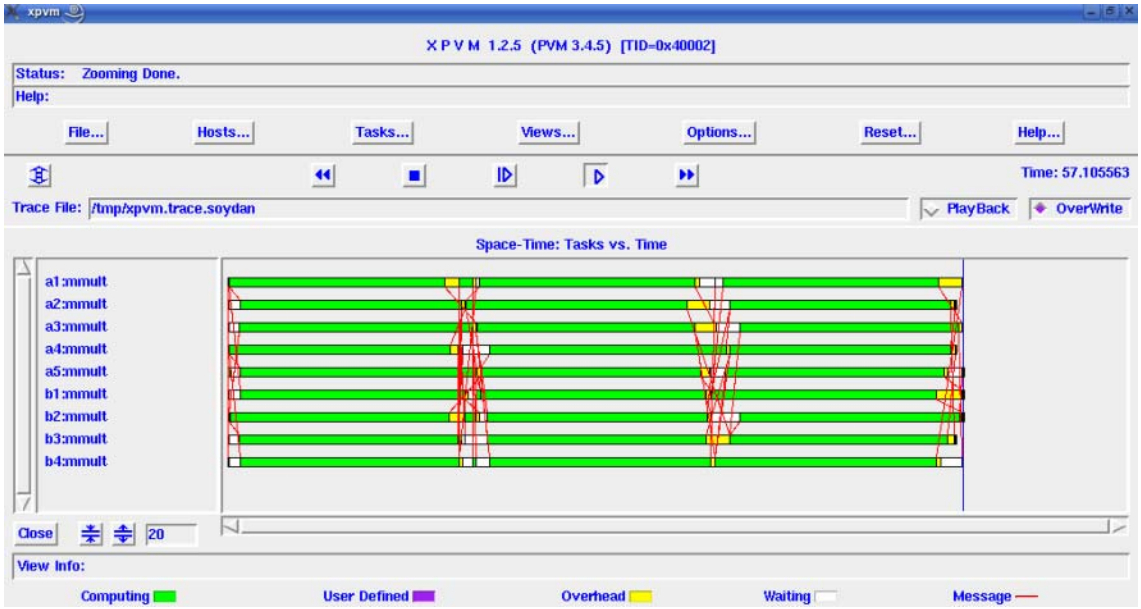


Şekil 4.13 Matris boyutu 3000 için işlemci sayısı – süre grafiği

İşlemci sayısı, matris boyutu ve çarpım işleminin süresini aynı anda görmek için şekil 4.14 çizilmiştir. Bu grafikte de işlemci sayısı artırımının süreyi kısalttığı ve başvurulan paralel hesaplamanın hedefe ulaştığı açıkça görülmektedir.

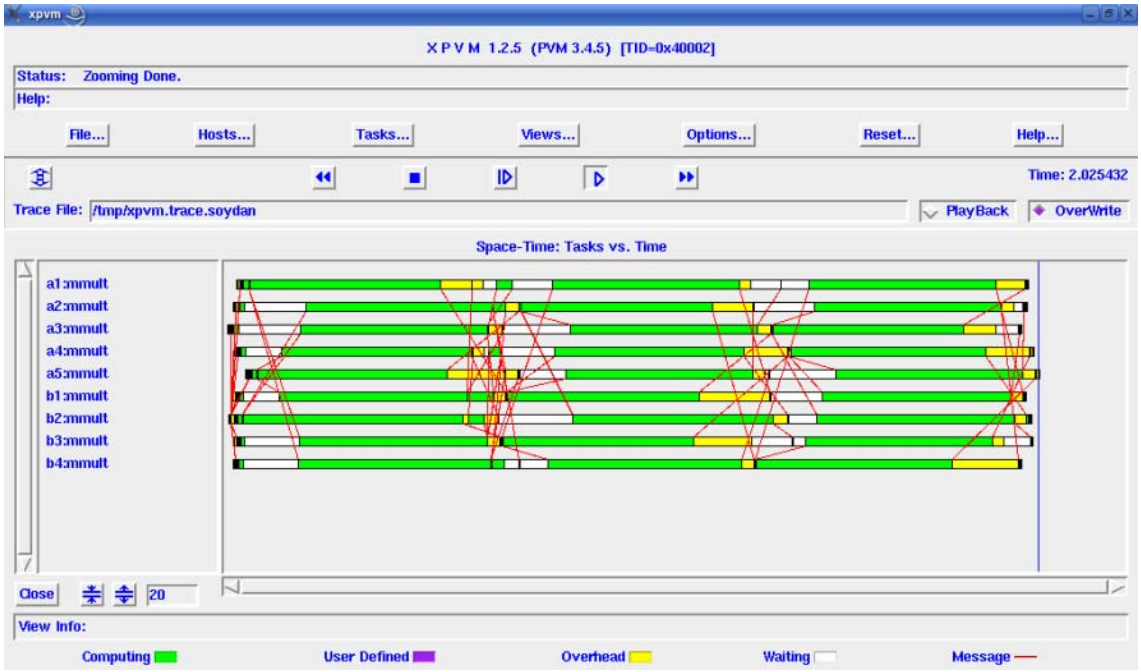


Şekil 4.14 İşlemci sayısı-matris boyutu-icra süresi grafiği



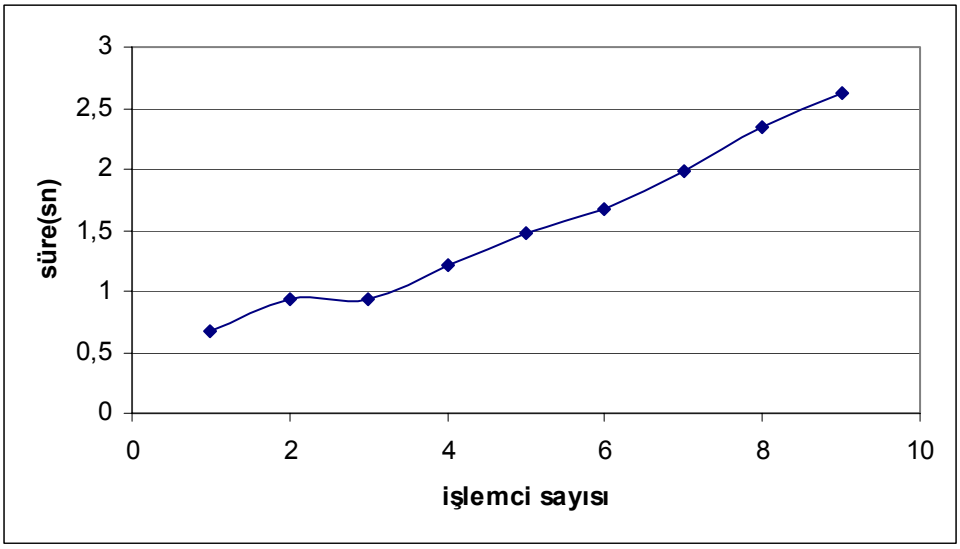
Şekil 4.15 Matris boyutu 2700 ve işlemci sayısı 9 için program aktivite grafiği

Şekil 4.15'te 9 işlemci ile 2700x2700 iki kare matrisin çarpımı sonucu oluşan, Şekil 4.16'da da 9 işlemci ile 900x900 iki kare matrisin çarpımı sonucu oluşan program aktivite grafiği görülmektedir. Matris çarpımının işlem ağırlıklı olması ve bu yükün işlemciler arasında paylaşılması icra süresini önemli ölçüde azaltmaktadır.

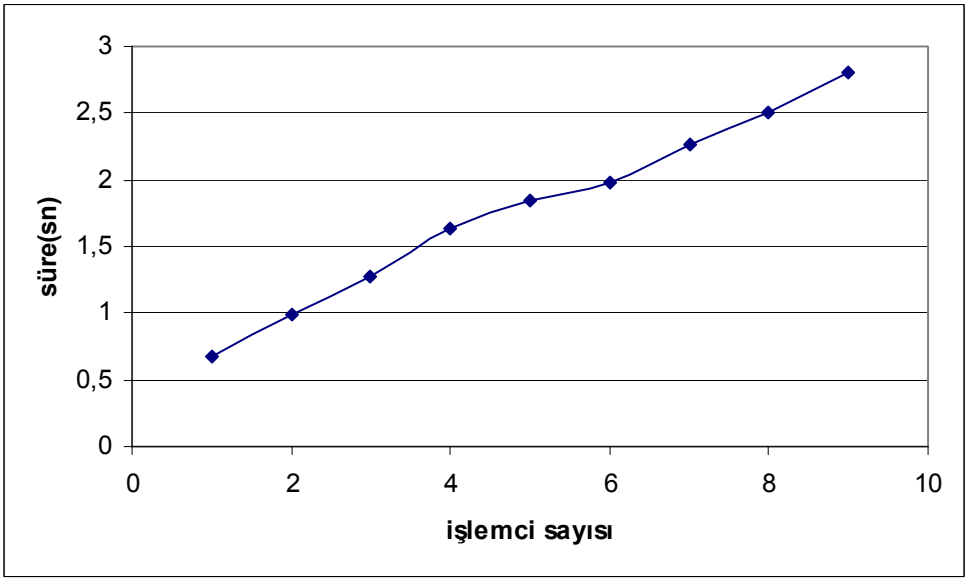


Şekil 4.16 Matris boyutu 900 ve işlemci sayısı 9 için program aktivite grafiği

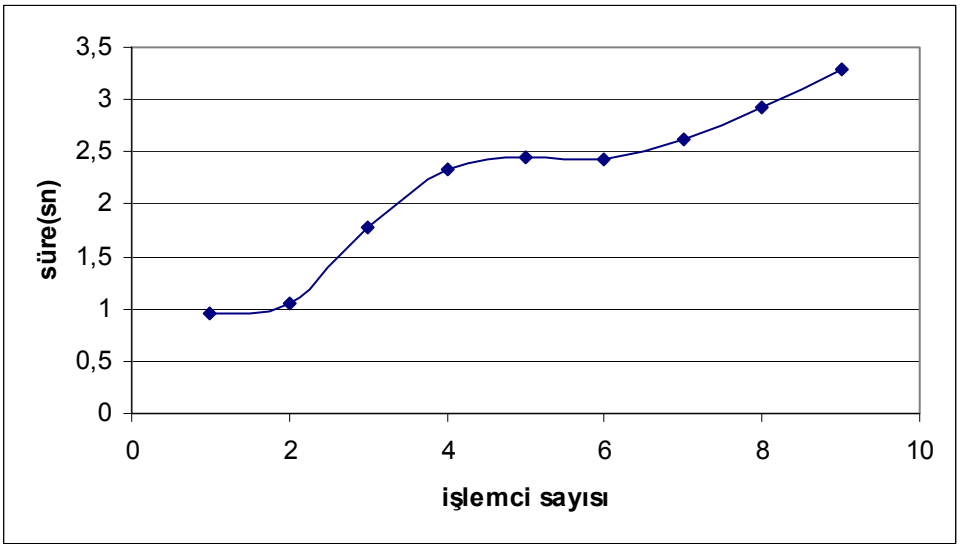
Isı difüzyon programı için, şekil 4.17, 4.18, 4.19, 4.20, 4.21 ve 4.22'de farklı problem boyutlarına göre işlemci sayısı - icra süresi grafikleri gösterilmiştir. Bu grafiklerde görüldüğü üzere artan işlemci sayısı programın icra süresini genellikle artırmıştır. Sadece 3000'den büyük problem boyutunda 2 işlemci kullanıldığında istenilen icra süresi azalmasına rastlanmıştır



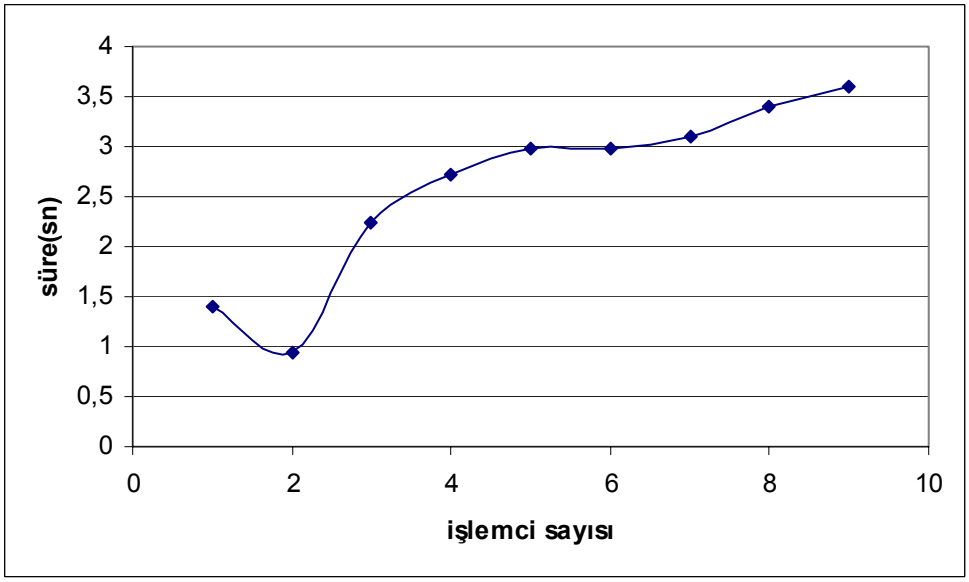
Şekil 4.17 Problem boyutu 1000 için işlemci sayısı – süre grafiği



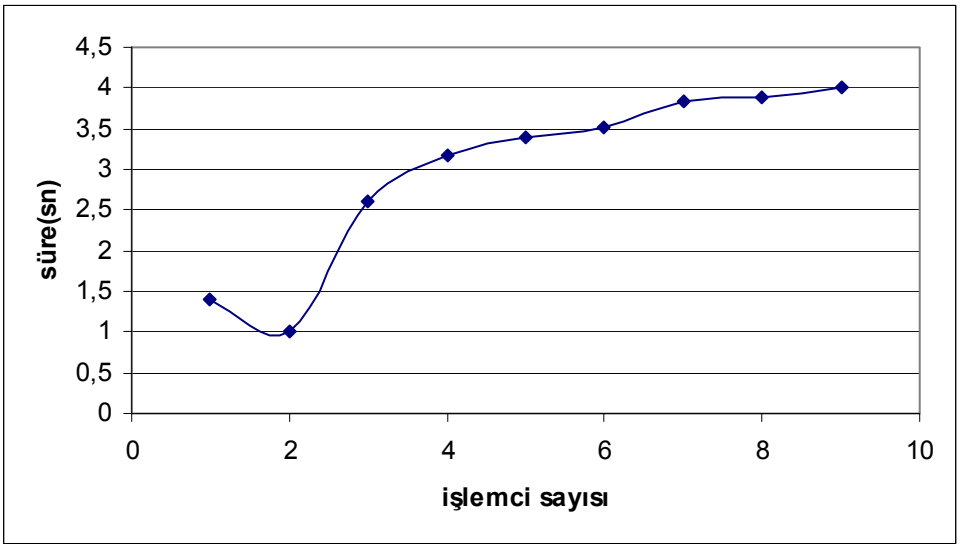
Şekil 4.18 Problem boyutu 2000 için işlemci sayısı – süre grafiği



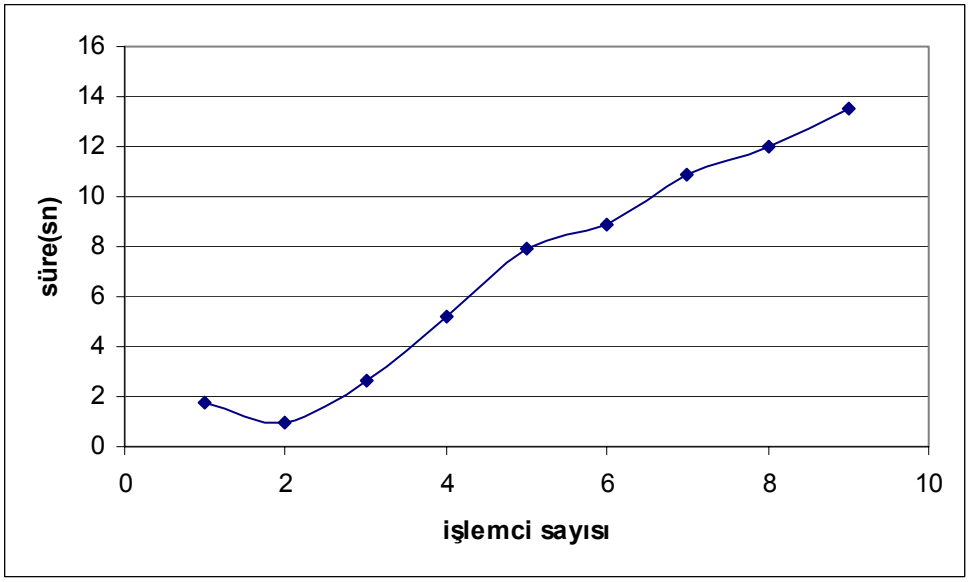
Şekil 4.19 Problem boyutu 3000 için işlemci sayısı – süre grafiği



Şekil 4.20 Problem boyutu 4000 için işlemci sayısı – süre grafiği

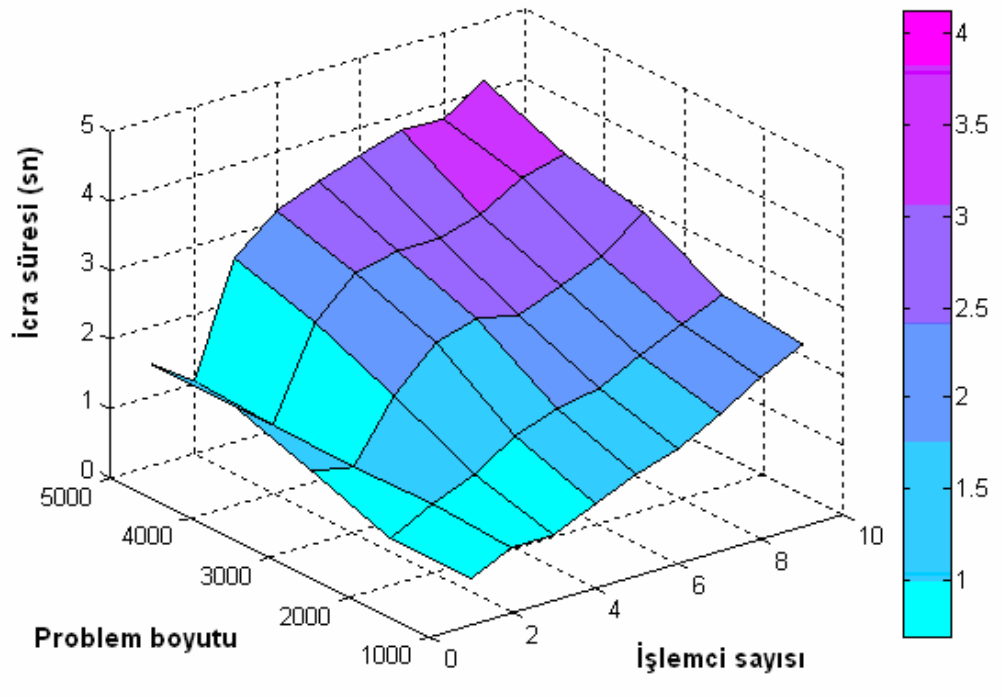


Şekil 4.21 Problem boyutu 5000 için işlemci sayısı – süre grafiği

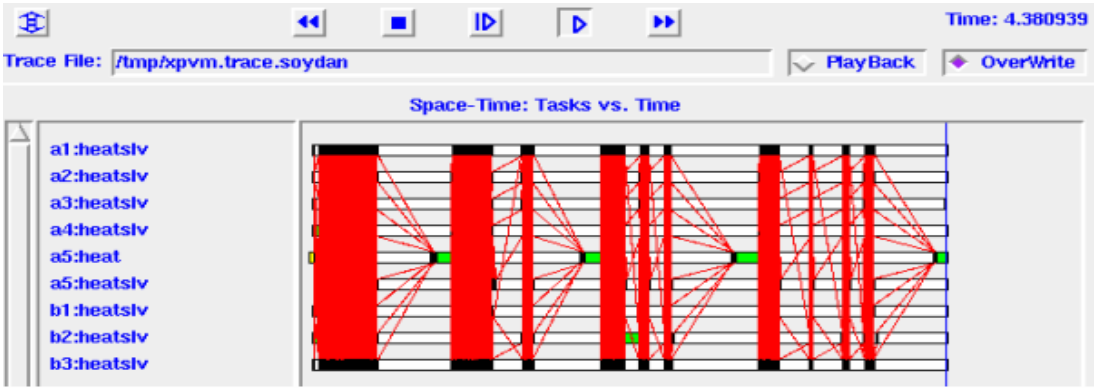


Şekil 4.22 Problem boyutu 10000 için işlemci sayısı – süre grafiği

Heat adlı program için çizilen 3 boyutlu grafikte de görüldüğü üzere(şekil 4.23), işlemci sayısının 2 ve problem boyutunun 3000-10000 aralığı için hedefe ulaşılmıştır. Ancak, diğer durumlarda işlemci sayısının artmasına rağmen icra süresinin azalması gerekirken sürenin arttığı gözlenmiştir. Programa ait program aktivite grafiğinin incelenmesi sonucunda(Şekil 4.24), seçilen uygulamanın haberleşme ağırlıklı olduğu, bu nedenle problem boyutundaki büyümenin beraberinde haberleşme yükünü de arttırdığı ve icra süresini uzattığı anlaşılmıştır.



Şekil 4.23 İşlemci sayısı-problem boyutu-icra süresi grafiği



Şekil 4.24 Problem boyutu 5000 ve işlemci sayısı 8 için program aktivite grafiği

5. SONUÇLAR ve ÖNERİLER

Bu çalışmada, paralel sanal bir makine, dağıtık hesaplama imkanı sunan PVM kullanılarak oluşturulmuştur. Bu makine(küme) üzerinde paralel hesaplama yapan uygulamalar çalıştırılmıştır. Uygulamalarda farklı problem boyutu ve farklı işlemci sayıları için icra süreleri ölçülmüştür. Ölçüm sonuçları grafiğe dökülmüş, paralellikten doğan performans değişimi incelenmiştir.

Tek işlemcili sistemlerde, performans işlemci hızıyla doğru orantılı olarak değişir ve program içinde icrası en uzun süren kısımda bir iyileştirme yapıldığında genel performans da iyileştirilmiş olur. Oysa dağıtık sistemlerde durum bundan farklıdır. Bu tür sistemlerin performansının, sadece işlemcilerin performansına bağlı olmadığı aynı zamanda işlemciler arasındaki koordinasyona ve iletişime de bağlı olduğu görülmüştür. Ağ ortamında oluşabilecek sıkışıklık ve güvenlik sorunlarının aşılması yanında uygulamaların paralel işleme uygun olma gerekliliği anlaşılmıştır.

Yüksek işlem gücü ve esnek sistem mimarisi, paralel işlemin avantajlarındandır. PVM ortamında çözülecek bir problem önce incelenmeli ve kümeyi oluşturan bilgisayarlar arasında görev dağılımı yapılmalıdır. Dağıtılan görevlerin icra sürelerinin işlemciler arasında çok farklı olmamasına dikkat edilmelidir. Bunun için ortamı oluşturan işlemcilerin performansları dikkate alınmalıdır. Örneğin bazı işlemcilerin gücü diğerlerine göre çok fazla veya az olabilir. Eğer bunlara, diğerlerine gönderilen kadar işlem gönderilirse programın sonlanması için az kapasiteli işlemcilerin işlerini bitirmesi beklemek gerekir. Burada amaç bütün işlemcilerin en az zamanda beklemesidir. Ayrıca, işlemciler arası mesajlaşma en aza indirilmelidir. İşlemcinin ya da sayısının değişmesi gibi durumlarda üzerinde çalışılan program tekrar gözden geçirilmelidir.

Paralel performans ölçümleri için en temel performans metriği cevap süresidir. Cevap süresi, bir programın başlangıcından o programa ait son işlemin bitişine kadar geçen süredir. Ölçülen bu süre programın gerçek performans ölçüsüdür.

Paralel sistemlerde performansın gözlemlenmesi bilgisayarların verimli kullanılması için önemlidir. Paralel sistemi oluşturan işlemcilerin sayısı, paralel sistemde çözülmek istenen problemin boyutu ve iletişim elemanlarının hızı genel performansı doğrudan etkilemektedir. Çok fazla işlemci kullanmak performansın artmasından ziyade azalmasına neden olabilmektedir.

Bu çalışmada PVM deneysel ortamı kullanılarak paralel programlarda performans analizleri yapılmıştır. Çalışmamızda problemin boyutuyla kullanılan işlemci sayısı arasındaki ilişki çıkartılmaya çalışılmıştır. Performans ölçümleri için cevap süresi metriğinden yararlanılmış ve uygulamanın farklı boyut ve işlemci sayıları için icra sürelerinin grafikleri elde edilmiştir. Yapılan deneyler sırasında problem boyutunun icra süresini artırdığı, işlemci sayısının ise matris çarpımında icra süresini azalttığı, ısı difüzyon denklemi çözümünde ise icra süresi üzerinde değişmeler gösterdiği gözlemlenmiştir. Artan işlemci sayısının icra süresini bazı durumlarda düşürmeme sebebinin programın haberleşme ağırlıklı olmasından kaynaklandığı görülmüştür. Verim alınabilmesi için uygulamanın performansın optimum olduğu noktalarda çalıştırılması gerektiği anlaşılmıştır. İncelediğimiz ısı programı uygulaması için ideal işlemci sayısının 2 ve problem boyutunun 3000 ve üzeri olduğu bulunmuştur.

Performans ölçümleriyle paralel sistemler daha verimli çalışacak şekilde yönetilebildikleri gibi paralel programlar değişik giriş ve işlemci konfigürasyonlarında daha verimli çalışacak şekilde ayarlanabilirler. Az sayıda giriş ve işlemci düzeniyle denenen bir paralel programın, farklı sayıda işlemcili sistemlerde farklı sayıda giriş büyüklüğü ile nasıl çalışacağı öngörülebilir ve performansının işlemci sayısı ve giriş büyüklüğü açısından en iyi olacağı çalışma bölgesi bulunabilir.

Bu çalışmaya ilave olarak, programın uygun bölümlerine sensörler ilave edilebilir ve performans verisi toplanabilir. Aynı şekilde küme üzerinde farklı programlar incelenerek bu programların performansları da detaylı irdelenebilir. Diğer taraftan PC bileşenlerindeki fiyat düşüşü sürdükçe daha yüksek performanslı kümelerin daha düşük maliyetlerle oluşturulması mümkün olacağından yüksek işlem gücü gerektiren hesaplamalar rahatlıkla yapılabilecektir.

Paralel sistemler gerek devlet gerekse özel sektör tarafından desteklenerek gün geçtikçe gelişmelidir. Yüksek performanslı bilgisayarlara olan ihtiyaç ortadadır. Artık şirketler yüksek ücretlerle sunucu almak yerine, şirket çalışanlarının bilgisayarlarının güçlerini birleştirerek sunucu-üstü bir işlem gücüne ulaşabilirler.

KAYNAKLAR DİZİNİ

[1] Özmen, A., Durmuş, B., "Dağıtık Sistemlerde Paralel Performans ve Karmaşıklık Analizi ". 1. Ulusal Yüksek Performanslı Bilişim Sempozyumu, Gebze-Kocaeli, Ekim, 2002.

[2] Morrison, R.S., 2003, Cluster Computing: Architectures, Operating Systems, Parallel Processing & Programming Languages, revision V2.4, 149p.

[3] Stallings William, Computer Organization and Architecture: designing for performance, Prentice Hall, 5. Edition, 2000.

[4] Karaca M.Selçuk, Bilgisayar Uzmanlığı, Nobel Yayın Dağıtım, Ocak2006.

[5] Angel, R., And Levia G., 2002, Linux Clusters for High Performans Computing, Universidad Autonoma de Madrid, Madrid, 76p.

[6] Özmen, A., A Minimal Overhead Instrumentation System. Proceeding of The Fifteenth International Symposium on Computer and Information Sciences (ISCIS XV), İstanbul, Ekim 2000.

[7] Geist A., Beguelin, A., Dongarra, J., Jiang, w., Manchek, R., Sunderam, V., 1994, PVM3 User's Guide and Reference Manual, Oak Ridge National Laboratory, Tennesse, 159p.

[8] Özmen, A., Paralel Gözlemleme (Monitör) Sistem Mimarisi. Bursa Elektrik-Elektronik ve Bilgisayar Mühendisliği Sempozyumu ve Fuarı (ELECO-2000), Bursa, Kasım 2000.

[9] Özmen, A., Durmuş, B., "Paralel Performans Metrikleri". Elektrik-Elektronik-Bilgisayar 9. Ulusal Kongresi, Kocaeli, 2001.

[10] Hwank, K., Xu, Z., Scalable Parallel Computing Technology, Architecture, Programming, McGraw Hill, USA, 1-800 (1997).

[11] Radejewski, J., Eadline, D., 1998, The Linux Documentation Project : Beowulf HOWTO, http//www.tldp.org

[12] http://www.top500.org

[13] http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0476.html

[14] Sterling, T., Becker, D., Warren, M., Cwik, T., Salmon, 1., Nitzberg, B., "An assessment of Beowulf-class computing for NASA requirements: initial findings from the first NASA workshop on Beowulf-c1ass clustered computing", Aerospace Conıferenıce, Proceedings., IEEE, USA,4: 367 -381 (1998).

[15] G. C. Fox, S. W. Otto, and A. J. G. Hey. Matrix algorithms on a hypercube I: Matrix Multiplication. *Parallel Computing*, 4:17-31, 1987.

EK 1 : "Mmult" Adlı Program için Farklı Matris Boyutu-İşlemci Sayısı Deneme Sonuçları

Tablo 4.1 Matris boyutu 300 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 300	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	0,50	0,50	0,50	0,50
İşlemci Sayısı : 4	0,27	0,21	0,20	0,23
İşlemci Sayısı : 9	0,20	0,20	0,20	0,20

Tablo 4.2 Matris boyutu 600 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 600	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	4,76	4,77	4,80	4,78
İşlemci Sayısı : 4	1,25	1,23	1,27	1,25
İşlemci Sayısı : 9	0,66	0,66	0,68	0,67

Tablo 4.3 Matris boyutu 900 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 900	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	17,45	17,45	17,45	17,45
İşlemci Sayısı : 4	4,20	4,21	4,21	4,21
İşlemci Sayısı : 9	2,48	2,60	2,21	2,43

Tablo 4.4 Matris boyutu 1200 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 1200	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	40,54	40,90	40,88	40,77
İşlemci Sayısı : 4	15,69	15,08	16,15	15,64
İşlemci Sayısı : 9	4,80	5,20	5,04	5,01

Tablo 4.5 Matris boyutu 1500 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 1500	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	80,21	79,46	80,86	80,18
İşlemci Sayısı : 4	31,97	36,73	36,05	34,92
İşlemci Sayısı : 9	15,00	14,36	14,33	14,56

Tablo 4.6 Matris boyutu 1800 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 1800	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	139,19	138,35	139,43	138,99
İşlemci Sayısı : 4	45,36	44,39	44,97	44,91
İşlemci Sayısı : 9	16,66	17,38	16,06	16,70

Tablo 4.7 Matris boyutu 2100 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 2100	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	222,21	222,68	222,34	222,41
İşlemci Sayısı : 4	54,29	55,66	55,21	55,05
İşlemci Sayısı : 9	27,24	26,78	26,16	26,73

Tablo 4.8 Matris boyutu 2400 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 2400	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	393,19	394,02	394,18	393,80
İşlemci Sayısı : 4	86,83	84,46	84,91	85,40
İşlemci Sayısı : 9	38,64	37,72	37,68	38,01

Tablo 4.9 Matris boyutu 2700 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 2700	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	490,29	487,85	489,44	489,19
İşlemci Sayısı : 4	121,91	121,45	121,63	121,66
İşlemci Sayısı : 9	57,01	56,64	56,69	56,78

Tablo 4.10 Matris boyutu 3000 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem Boyutu: 3000	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ortalama
İşlemci Sayısı : 1	713,49	678,48	685,69	692,55
İşlemci Sayısı : 4	166,05	166,41	166,38	166,28
İşlemci Sayısı : 9	76,26	75,51	76,73	76,17

EK 2 : "Heat" Adlı Program için Farklı Problem Boyutu-İşlemci Sayısı Deneme Sonuçları

Tablo 4.11 Problem boyutu 1000 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem boyutu : 1000	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ölçüm 4	Ortalama
İşlemci Sayısı : 1	0,69	0,67	0,65	0,67	0,67
İşlemci Sayısı : 2	0,92	1,08	0,92	0,80	0,93
İşlemci Sayısı : 3	0,95	0,92	0,92	0,93	0,93
İşlemci Sayısı : 4	1,21	1,22	1,23	1,22	1,22
İşlemci Sayısı : 5	1,51	1,53	1,42	1,42	1,47
İşlemci Sayısı : 6	1,90	1,61	1,52	1,65	1,67
İşlemci Sayısı : 7	1,98	1,96	1,99	1,99	1,98
İşlemci Sayısı : 8	2,06	2,36	2,75	2,19	2,34
İşlemci Sayısı : 9	2,51	3,26	2,68	2,11	2,64

Tablo 4.12 Problem boyutu 2000 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem boyutu : 2000	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ölçüm 4	Ortalama
İşlemci Sayısı : 1	0,69	0,65	0,68	0,70	0,68
İşlemci Sayısı : 2	0,88	1,30	0,91	0,91	1,00
İşlemci Sayısı : 3	1,24	1,28	1,20	1,24	1,24
İşlemci Sayısı : 4	1,49	1,61	1,87	1,55	1,63
İşlemci Sayısı : 5	1,86	1,78	1,84	1,92	1,85
İşlemci Sayısı : 6	1,88	2,01	1,98	2,01	1,97
İşlemci Sayısı : 7	2,27	2,28	2,27	2,26	2,27
İşlemci Sayısı : 8	2,90	2,60	2,10	2,56	2,54
İşlemci Sayısı : 9	3,02	2,70	2,90	2,50	2,78

Tablo 4.13 Problem boyutu 3000 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem boyutu : 3000	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ölçüm 4	Ortalama
İşlemci Sayısı : 1	1,03	1,24	1,01	1,04	1,08
İşlemci Sayısı : 2	0,95	0,94	0,88	1,03	0,95
İşlemci Sayısı : 3	2,10	1,86	1,54	1,74	1,81
İşlemci Sayısı : 4	2,98	2,03	2,38	2,21	2,40
İşlemci Sayısı : 5	2,09	3,87	2,26	2,02	2,56
İşlemci Sayısı : 6	2,48	2,39	2,54	2,39	2,45
İşlemci Sayısı : 7	2,53	2,54	2,77	2,96	2,70
İşlemci Sayısı : 8	3,64	2,93	2,49	2,74	2,95
İşlemci Sayısı : 9	4,18	3,12	3,29	3,01	3,40

Tablo 4.14 Problem boyutu 4000 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem boyutu : 4000	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ölçüm 4	Ortalama
İşlemci Sayısı : 1	1,34	1,34	1,87	1,29	1,46
İşlemci Sayısı : 2	1,02	0,91	0,89	1,08	0,98
İşlemci Sayısı : 3	2,06	2,41	2,54	2,11	2,28
İşlemci Sayısı : 4	3,17	2,86	2,77	2,48	2,82
İşlemci Sayısı : 5	2,99	2,98	2,46	3,39	2,96
İşlemci Sayısı : 6	2,54	2,91	3,45	2,97	2,97
İşlemci Sayısı : 7	2,88	3,11	3,54	3,06	3,15
İşlemci Sayısı : 8	4,23	3,24	3,36	3,21	3,51
İşlemci Sayısı : 9	3,62	3,79	3,66	3,57	3,66

Tablo 4.15 Problem boyutu 5000 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem boyutu : 5000	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ölçüm 4	Ortalama
İşlemci Sayısı : 1	1,94	1,27	1,39	1,24	1,46
İşlemci Sayısı : 2	1,10	1,25	0,99	0,90	1,06
İşlemci Sayısı : 3	2,85	2,72	2,59	2,44	2,65
İşlemci Sayısı : 4	3,65	2,82	2,77	3,36	3,15
İşlemci Sayısı : 5	3,46	3,54	3,21	3,39	3,40
İşlemci Sayısı : 6	3,21	3,12	4,48	3,59	3,60
İşlemci Sayısı : 7	4,23	3,69	3,54	3,70	3,79
İşlemci Sayısı : 8	4,12	3,88	3,62	4,38	4,00
İşlemci Sayısı : 9	3,27	3,98	5,04	4,29	4,15

Tablo 4.16 Problem boyutu 10000 ve değişen işlemci sayısına göre icra süreleri(sn).

Problem boyutu : 10000	Ölçüm 1	Ölçüm 2	Ölçüm 3	Ölçüm 4	Ortalama
İşlemci Sayısı : 1	1,86	1,94	1,62	1,70	1,78
İşlemci Sayısı : 2	1,22	0,91	0,81	0,94	0,97
İşlemci Sayısı : 3	2,75	2,88	2,52	2,45	2,65
İşlemci Sayısı : 4	5,35	6,11	4,12	5,10	5,17
İşlemci Sayısı : 5	5,89	6,10	7,01	8,20	6,80
İşlemci Sayısı : 6	8,94	5,21	8,83	11,22	8,55
İşlemci Sayısı : 7	11,26	10,35	11,02	10,89	10,88
İşlemci Sayısı : 8	11,68	10,87	14,78	11,39	12,18
İşlemci Sayısı : 9	16,69	15,60	12,56	13,75	14,65