





**DAĞITIK MİMARİ TASARIMI  
VE YÖNETİM SİSTEMİ GELİŞTİRİLMESİ**

**YÜKSEK LİSANS TEZİ**

**Mustafa Haluk AKGÜNDÜZ**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Bilgisayar Mühendisliği Yüksek Lisans Programı**

**MAYIS 2015**



**DAĞITIK MİMARİ TASARIMI  
VE YÖNETİM SİSTEMİ GELİŞTİRİLMESİ**

**YÜKSEK LİSANS TEZİ**

**Mustafa Haluk AKGÜNDÜZ  
(504011402)**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Bilgisayar Mühendisliği Yüksek Lisans Programı**

**Tez Danışmanı: Prof. Dr. Eşref ADALI**

**MAYIS 2015**



İTÜ, Fen Bilimleri Enstitüsü'nün 504011402 numaralı Yüksek Lisans Öğrencisi **Mustafa Haluk AKGÜNDÜZ**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı “**DAĞITIK MİMARİ TASARIMI VE YÖNETİM SİSTEMİ GELİŞTİRİLMESİ**” başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

**Tez Danışmanı :**      **Prof. Dr. Eşref ADALI** .....  
İstanbul Teknik Üniversitesi

**Jüri Üyeleri :**        **Prof. Dr. Oğuz TOSUN** .....  
Boğaziçi Üniversitesi

**Yrd. Doç. Dr. Gökhan İNCE** .....  
İstanbul Teknik Üniversitesi

**Teslim Tarihi :**        **04 Mayıs 2015**  
**Savunma Tarihi :**    **25 Mayıs 2015**



*Aileme,*



## ÖNSÖZ

Tez çalışmamda benden desteğini ve sabrını esirgemeyen danışmanım Prof. Dr. Eşref ADALI'ya, tezi bitirmemde sağladığı destek ve motivasyondan dolayı arkadaşım Dr. Toygar AKGÜN'e ve tez boyunca sağladıkları manevi desteklerinden dolayı aileme en içten teşekkürlerimi sunarım.

Mayıs 2015

Mustafa Haluk AKGÜNDÜZ



## İÇİNDEKİLER

|                                                                                                    | <u>Sayfa</u> |
|----------------------------------------------------------------------------------------------------|--------------|
| <b>ÖNSÖZ .....</b>                                                                                 | <b>vii</b>   |
| <b>İÇİNDEKİLER .....</b>                                                                           | <b>ix</b>    |
| <b>KISALTMALAR.....</b>                                                                            | <b>xiii</b>  |
| <b>ÇİZELGE LİSTESİ.....</b>                                                                        | <b>xv</b>    |
| <b>ŞEKİL LİSTESİ.....</b>                                                                          | <b>xvii</b>  |
| <b>ÖZET .....</b>                                                                                  | <b>xix</b>   |
| <b>SUMMARY .....</b>                                                                               | <b>xxiii</b> |
| <b>1. GİRİŞ.....</b>                                                                               | <b>1</b>     |
| 1.1 Tezin Amacı.....                                                                               | 2            |
| 1.2 Literatür Araştırması .....                                                                    | 3            |
| 1.2.1 Biyomedikal mühendisliğinde paralel ve dağıtık işleme teknikleri .....                       | 4            |
| 1.2.2 Cluster Juggler - Bilgisayar kümesi tabanlı sanal gerçeklik .....                            | 4            |
| 1.2.3 Zemin-Yapı etkileşimi sorunlarının bilgisayar kümeleriyle paralel<br>çözümü .....            | 5            |
| 1.2.4 RPiCluster - Raspberry Pi tabanlı Beowulf kümesi .....                                       | 5            |
| 1.2.5 Mikroişlemci mimarisinin dağıtık bellekli süper bilgisayarların<br>hızlanmasına etkisi ..... | 7            |
| 1.2.6 HPC sistemlerinde ağ başarımlı çalışmaları .....                                             | 10           |
| 1.2.7 Beowulf kümesi üzerinde paralel SAR işlemcisinin geliştirilmesi .....                        | 12           |
| 1.2.8 Küçük, düşük maliyetli, disksiz Beowulf kümelerinin başarımlı modeli .....                   | 13           |
| 1.3 Tezin Katkısı.....                                                                             | 15           |
| 1.3.1 Güvenilirlik.....                                                                            | 16           |
| 1.3.1.1 Çoklu çalıştırma.....                                                                      | 16           |
| 1.3.1.2 Birebir yedekleme.....                                                                     | 17           |
| 1.3.1.3 Havuz yedeklemesi.....                                                                     | 18           |
| 1.3.1.4 Yedekleme yönteminin seçimi.....                                                           | 19           |
| 1.3.2 Verim .....                                                                                  | 20           |
| 1.3.2.1 Dosya aktarımı .....                                                                       | 20           |
| 1.3.2.2 Düğüm istatistiği.....                                                                     | 21           |
| 1.3.2.3 Önceliklendirme .....                                                                      | 21           |
| 1.3.2.4 Yük dengeleyiciler.....                                                                    | 22           |
| 1.4 Tezin Kapsamı .....                                                                            | 22           |
| <b>2. DAĞITIK MİMARİ VE YÖNETİM YAZILIMLARI.....</b>                                               | <b>25</b>    |
| 2.1 Dağıtık Mimari Çeşitleri.....                                                                  | 25           |
| 2.1.1 Ana bilgisayarlar .....                                                                      | 25           |
| 2.1.2 İstemci / Sunucu mimarisi.....                                                               | 26           |
| 2.1.3 Dağıtık nesne mimarisi.....                                                                  | 27           |

|                                                            |           |
|------------------------------------------------------------|-----------|
| 2.1.3.1 Uzak yordam çağrıları .....                        | 29        |
| 2.1.3.2 CORBA.....                                         | 29        |
| 2.1.3.3 DCOM .....                                         | 29        |
| 2.1.4 Bilgisayar kümeleri mimarisi .....                   | 30        |
| 2.1.4.1 Yüksek devamlılık kümeleri .....                   | 30        |
| 2.1.4.2 Yük dengeleyici kümeler.....                       | 31        |
| 2.1.4.3 Yüksek başarımlı kümeler .....                     | 31        |
| 2.1.4.4 Beowulf kümesi .....                               | 32        |
| 2.1.5 Grid mimarisi.....                                   | 32        |
| 2.2 Dağıtık Mimarilerin Değerlendirilmesi .....            | 33        |
| 2.3 Haberleşme Altyapısı .....                             | 35        |
| 2.3.1 Mesaj aktarımı .....                                 | 35        |
| 2.3.1.1 Berkeley soketleri .....                           | 36        |
| 2.3.1.2 Mesaj aktarım arabirimi.....                       | 37        |
| 2.3.2 Uzak yordam çağrısı.....                             | 39        |
| 2.3.3 Dağıtık paylaşımlı bellek.....                       | 40        |
| 2.3.4 Altyapının belirlenmesi .....                        | 43        |
| 2.3.4.1 Internet soketleri .....                           | 44        |
| 2.3.4.2 Unix alan soketleri.....                           | 44        |
| 2.3.4.3 Hat.....                                           | 45        |
| 2.4 Mesajlaşma Altyapısı .....                             | 45        |
| 2.5 Bilgisayar Kümeleri Mimarisi Yönetim Yazılımları ..... | 46        |
| 2.5.1 OSCAR.....                                           | 46        |
| 2.5.2 Rocks .....                                          | 47        |
| 2.5.3 Warewulf.....                                        | 47        |
| 2.5.4 xCAT.....                                            | 48        |
| 2.5.5 Debian küme bileşenleri .....                        | 48        |
| 2.6 Paralleleştirme .....                                  | 48        |
| 2.6.1 Amdahl'ın kanunu .....                               | 49        |
| 2.6.2 Gustafson'un kanunu .....                            | 49        |
| 2.7 Sınama Uygulamasının Belirlenmesi .....                | 50        |
| 2.8 Donanımların Belirlenmesi .....                        | 51        |
| <b>3. MERKEZ YÖNETİM YAZILIMI .....</b>                    | <b>53</b> |
| 3.1 İş Akışı.....                                          | 54        |
| 3.2 Ana Bileşenler .....                                   | 57        |
| 3.2.1 Dağıtıcı .....                                       | 57        |
| 3.2.2 Toplayıcı .....                                      | 62        |
| 3.2.3 Düğüm .....                                          | 64        |
| 3.3 Haberleşme Altyapısının Tasarlanması .....             | 66        |
| 3.3.1 Zamanlayıcı .....                                    | 67        |
| 3.3.2 Mesaj .....                                          | 68        |
| 3.3.2.1 Mesaj tipleri.....                                 | 71        |
| 3.3.3 Arabirim .....                                       | 72        |
| 3.3.4 Net .....                                            | 73        |
| 3.3.5 UnixSocket .....                                     | 73        |

|                                                     |            |
|-----------------------------------------------------|------------|
| 3.3.6 Hat .....                                     | 74         |
| 3.3.7 Bağlayıcı.....                                | 74         |
| 3.3.8 Mesaj akışı.....                              | 74         |
| 3.3.8.1 Mesaj Gönderimi .....                       | 75         |
| 3.3.8.2 Mesaj Alımı .....                           | 76         |
| 3.4 Kuralların Tanımlanması .....                   | 76         |
| 3.4.1 İçerik.....                                   | 77         |
| 3.4.1.1 Dosya içeriği.....                          | 77         |
| 3.4.1.2 Değiştirge içeriği .....                    | 77         |
| 3.4.1.3 Çalıştırıcı içeriği .....                   | 78         |
| 3.5 Kullanıcı Arayüzünün Tasarımı.....              | 79         |
| 3.5.1 WXWidgets arayüzü.....                        | 79         |
| 3.5.1.1 Dağıtıcı arayüzü.....                       | 80         |
| 3.5.1.2 Toplayıcı arayüzü.....                      | 81         |
| 3.5.1.3 Düğüm arayüzü.....                          | 81         |
| 3.5.2 Terminal arayüzü .....                        | 82         |
| 3.6 Matris Uygulamasının Geliştirilmesi .....       | 83         |
| <b>4. SINAMALAR ve SONUÇLAR .....</b>               | <b>85</b>  |
| 4.1 Platform .....                                  | 85         |
| 4.2 Uygulama .....                                  | 87         |
| 4.3 Sınamalar.....                                  | 87         |
| 4.3.1 Sınama 1 .....                                | 89         |
| 4.3.2 Sınama 2 .....                                | 91         |
| 4.3.3 Sınama 3 .....                                | 92         |
| 4.3.4 Sınama 4 .....                                | 94         |
| 4.3.5 Sınama 5 .....                                | 95         |
| 4.3.6 Sınama 6 .....                                | 96         |
| 4.3.7 Sınama 7 .....                                | 96         |
| 4.4 Maliyet Analizi.....                            | 97         |
| 4.5 Sonuç .....                                     | 101        |
| 4.5.1 Atanmış bilgisayar başarımlar sonuçları ..... | 104        |
| 4.6 Gelecekteki Çalışmalar.....                     | 105        |
| <b>KAYNAKLAR.....</b>                               | <b>107</b> |
| <b>EKLER .....</b>                                  | <b>111</b> |
| EK A.1 .....                                        | 113        |
| 1.1 Raspberry PI Model B .....                      | 113        |
| 1.2 Raspberry PI 2 Model B .....                    | 113        |
| 1.3 BeagleBone Black Model C .....                  | 114        |
| 1.4 HardKernel ODROID U3 .....                      | 115        |
| <b>ÖZGEÇMİŞ .....</b>                               | <b>117</b> |



## **KISALTMALAR**

|              |                                                            |
|--------------|------------------------------------------------------------|
| <b>BSD</b>   | : Berkeley Software Distribution                           |
| <b>CORBA</b> | : Common Object Request Broker Architecture                |
| <b>DCOM</b>  | : Distributed Component Object Model                       |
| <b>DSM</b>   | : Distributed Shared Memory                                |
| <b>FIFO</b>  | : First-In, First-Out                                      |
| <b>GPFS</b>  | : General Parallel File System                             |
| <b>GPU</b>   | : Graphics Processing Unit                                 |
| <b>IP</b>    | : Internet Protocol                                        |
| <b>IPC</b>   | : Inter-Process Communication                              |
| <b>JSON</b>  | : JavaScript Object Notation                               |
| <b>LAM</b>   | : Local Area MultiComputer                                 |
| <b>LZO</b>   | : Lempel–Ziv–Oberhumer Lossless Data Compression Algorithm |
| <b>MD5</b>   | : MD5 Message-Digest Algorithm                             |
| <b>MPI</b>   | : Message Passing Interface                                |
| <b>MPICH</b> | : High-Performance Portable MPI                            |
| <b>OSCAR</b> | : Open Source Cluster Application Resources                |
| <b>PBS</b>   | : Portable Batch System                                    |
| <b>PICL</b>  | : Portable Instrumented Communications Library             |
| <b>PVFS</b>  | : Parallel Virtual File System                             |
| <b>PVM</b>   | : Parallel Virtual Machine                                 |
| <b>RFI</b>   | : Radio Frequency Interference                             |
| <b>RMI</b>   | : Remote Method Invocation                                 |
| <b>RPC</b>   | : Remote Procedure Call                                    |
| <b>SAR</b>   | : Synthetic Aperture Radar                                 |
| <b>SGE</b>   | : Sun Grid Engine                                          |
| <b>TCP</b>   | : Transmission Control Protocol                            |
| <b>UDP</b>   | : User Datagram Protocol                                   |
| <b>xCAT</b>  | : Extreme Cluster/Cloud Administration Toolkit             |



## ÇİZELGE LİSTESİ

|                                                                     | <u>Sayfa</u> |
|---------------------------------------------------------------------|--------------|
| Çizelge 1.1 : Kullanılan donanım bileşenleri .....                  | 6            |
| Çizelge 1.2 : CRAFT karşılaştırma sonuçları. ....                   | 7            |
| Çizelge 1.3 : Sistemlerin 64 KB'lık mesaj aktarım hızları. ....     | 11           |
| Çizelge 1.4 : İşlemci karşılaştırmaları.....                        | 13           |
| Çizelge 1.5 : Kullanılan donanımlar.....                            | 14           |
| Çizelge 1.6 : Ölçümü yapılan matematiksel denklem ve sınıfları..... | 14           |
| Çizelge 4.1 : Uygulama bilgileri .....                              | 87           |
| Çizelge 4.2 : Matris bilgileri .....                                | 88           |
| Çizelge 4.3 : Sınama 1 - Tek düğümlü .....                          | 89           |
| Çizelge 4.4 : Sınama 1 - İki düğümlü.....                           | 90           |
| Çizelge 4.5 : Sınama 1 - Üç düğümlü .....                           | 90           |
| Çizelge 4.6 : Sınama 2 - Tek düğümlü .....                          | 91           |
| Çizelge 4.7 : Sınama 3 - Tek düğümlü .....                          | 93           |
| Çizelge 4.8 : Sınama 3 - İki düğümlü.....                           | 93           |
| Çizelge 4.9 : Sınama 3 - Üç düğümlü .....                           | 93           |
| Çizelge 4.10 : Sınama 4 - Üç düğümlü .....                          | 95           |
| Çizelge 4.11 : Sınama 5 - Tek düğümlü .....                         | 95           |
| Çizelge 4.12 : Sınama 6 - Tek düğümlü .....                         | 96           |
| Çizelge 4.13 : Sınama 7 - Tek düğümlü .....                         | 97           |
| Çizelge 4.14 : Masaüstü bilgisayar donanım bileşenleri.....         | 97           |
| Çizelge 4.15 : Diğer donanım bileşenleri.....                       | 98           |
| Çizelge 4.16 : Maliyet indeksi, GY = %0 .....                       | 99           |
| Çizelge 4.17 : Maliyet indeksi, GY = %10 .....                      | 100          |
| Çizelge 4.18 : Maliyet indeksi, GY = %20 .....                      | 100          |
| Çizelge 4.19 : Maliyet indeksi, GY = %30 .....                      | 100          |
| Çizelge 4.20 : Maliyet indeksi, GY = %50 .....                      | 100          |
| Çizelge 4.21 : Maliyet indeksi, GY = %80 .....                      | 101          |
| Çizelge 4.22 : Maliyet indeksi, GY = %100 .....                     | 101          |



## ŞEKİL LİSTESİ

|                                                               | <u>Sayfa</u> |
|---------------------------------------------------------------|--------------|
| Şekil 1.1 : RPiCluster başarımlı grafiğı.....                 | 6            |
| Şekil 1.2 : RPiCluster çalışma zamanı grafiğı. ....           | 7            |
| Şekil 1.3 : Uygulama 1 hızlanma grafiğı. ....                 | 8            |
| Şekil 1.4 : Uygulama 2 hızlanma grafiğı. ....                 | 9            |
| Şekil 1.5 : Uygulama 3 hızlanma grafiğı. ....                 | 9            |
| Şekil 1.6 : Sıfır kopyalama tekniğı.....                      | 10           |
| Şekil 1.7 : Çoklu çalıştırma.....                             | 17           |
| Şekil 1.8 : Birebir yedekleme.....                            | 18           |
| Şekil 1.9 : Havuz yedeklemesi.....                            | 19           |
| Şekil 2.1 : Ana bilgisayar çalışma yöntemi. ....              | 26           |
| Şekil 2.2 : İstemci / Sunucu mimarisi. ....                   | 28           |
| Şekil 2.3 : Dağıtık nesne mimarisi.....                       | 29           |
| Şekil 2.4 : Bilgisayar kümeleri mimarisi. ....                | 31           |
| Şekil 2.5 : Grid mimarisi.....                                | 33           |
| Şekil 2.6 : Berkeley soketleri. ....                          | 37           |
| Şekil 2.7 : Mesaj aktarım arabirimi.....                      | 39           |
| Şekil 2.8 : Uzak yordam çağrısı.....                          | 40           |
| Şekil 2.9 : Dağıtık paylaşımli bellek.....                    | 41           |
| Şekil 3.1 : Merkez yönetim yazılımının işleyişı. ....         | 55           |
| Şekil 3.2 : Merkez yönetim yazılımının detay görünümü.....    | 55           |
| Şekil 3.3 : İş akışı.....                                     | 58           |
| Şekil 3.4 : Dağıtıcı akış diyagramı.....                      | 60           |
| Şekil 3.5 : Toplayıcı akış diyagramı. ....                    | 63           |
| Şekil 3.6 : Döğüm akış diyagramı.....                         | 65           |
| Şekil 3.7 : Zamanlayıcı akış diyagramı.....                   | 68           |
| Şekil 3.8 : Mesaj aktarım yapısı.....                         | 69           |
| Şekil 3.9 : Mesaj başlık yapısı. ....                         | 70           |
| Şekil 3.10 : Mesaj blok başlığı yapısı. ....                  | 70           |
| Şekil 3.11 : Mesaj akışı.....                                 | 75           |
| Şekil 3.12 : Dağıtıcı kullanıcı arayüzü.....                  | 80           |
| Şekil 3.13 : Toplayıcı kullanıcı arayüzü. ....                | 81           |
| Şekil 3.14 : Döğüm kullanıcı arayüzü.....                     | 82           |
| Şekil 4.1 : Tek bilgisayar - Başarım artışı.....              | 102          |
| Şekil 4.2 : Üç Raspberry PI - Başarım artışı.....             | 103          |
| Şekil 4.3 : Atanmış bilgisayar başarımlı / fiyat grafiğı..... | 105          |
| Şekil A.1 : Raspberry PI Model B. ....                        | 114          |
| Şekil A.2 : Raspberry PI 2 Model B. ....                      | 115          |

**Şekil A.3** : BeagleBone Black Model C. .... 116

**Şekil A.4** : HardKernel ODROID U3. .... 116

## **DAĞITIK MİMARİ TASARIMI VE YÖNETİM SİSTEMİ GELİŞTİRİLMESİ**

### **ÖZET**

Dağıtık Sistemler, büyük ölçekli uygulamaların zaman ve alan gereksinimi konusunda karşılaştıkları engelleri aşma yönünde ortaya çıkmış bir mimari türüdür. Dağıtık sistemlere kadar olan zamanda geliştirilen mimariler, uygulamaları hızlandırma yönünde donanım (çok işlemcili sistemler, çok çekirdekli işlemciler vb.) ya da yazılım (istemci / sunucu vb.) alanında bir gelişim gösterse de uygulamaların büyüklüklerindeki artış ivmesini yakalayamamışlardır. Diğer yandan başarımların kriterlerinin sınırlı sayıda fiziki bilgisayarlar üzerinde gerçekleştirilmek istenmesi, ortaya maliyeti yüksek bir çözüm çıkarmaktadır.

Son zamanlarda, akıllı telefon ve tablet bilgisayarların yaygınlaşması, atanmış sistemlere yönelik işlemci, bellek gibi donanımların başarımların / fiyat oranlarının artmasını sağlamıştır. Böylece mevcut geliştirilmiş ya da tasarlanacak düşük maliyetli atanmış sistemlerin birarada kullanılmasıyla, maliyeti yüksek hızlı bilgisayarlardan daha iyi başarımlar / fiyat oranı elde edilebileceği fikri ortaya çıkmıştır.

Hedeflenen çalışmada amaç, büyük ölçekli örnek bir uygulamanın, dağıtık sistem mimarisi vasıtasıyla ayrıştırılıp atanmış bilgisayarlarda çalıştırılmasını sağlamak ve elde edilen başarımların / fiyat oranının mevcut sistemlerle karşılaştırmasını yapmaktır. Bununla birlikte, sistemdeki donanım sayısının artması, sistemin sürdürülebilirliğine ait riskleri, donanımlara dağıtmakta, böylece hem güvenilirlik hem de verimlilik anlamında daha iyi bir sistem ortaya çıkmasını sağlamaktadır. Dağıtık sistemlere yönelik olarak yapılan literatür araştırmalarında; ana bilgisayar, istemci / sunucu, dağıtık nesne, bilgisayar kümeleri ve grid mimarileri incelenmiş, üstün ve eksik yönleri irdelenmiştir. Çalışmada, uygulamaların yüksek başarımlı olarak çalıştırılmak istenmesi ve mevcut atanmış bilgisayarların sınırlı kaynaklara sahip olması nedeniyle, çözüme yönelik en uygun mimarinin bilgisayar kümeleri mimarisi olduğu belirlenmiş ve tez kapsamında tercih edilen mimari olmuştur.

Sayılan üstünlüklerine karşın bilgisayar kümeleri mimarisinin en önemli eksikliği platform esnekliği sağlamamasıdır. Çalıştırılacak uygulamanın atanmış bilgisayarın sahip olduğu işlemci mimarisi ve işletim sistemi ile uyumlu olması gerekmektedir. Ancak bilgisayar sistemlerinde genel olarak platform bağımsızlığı ile başarımlar arasında karşılıklı dışlama durumu hakimdir, yani platform bağımsızlığı beraberinde belirgin bir başarımlar kaybını da getirmektedir. Tezin amacındaki öncelik başarımlar artırımı olduğundan, platform bağımsızlığı değerlendirme dışında kalmıştır.

Bilgisayar kümeleri mimarisinde, ana bir bilgisayar üzerinden merkezi bir yönetim yazılımı vasıtasıyla sistemdeki düğümler yönetilmekte, uygulama alt görevlerinin dağıtılıp çalıştırılması sağlanmaktadır. Bu tez çalışmasıyla bir Merkez Yönetim Yazılımı ve sınamaya yönelik olarak matris çarpımı tabanlı bir matematiksel uygulama

geliştirilmiş, bu uygulamanın geliştirilen Merkez Yönetim Yazılımı vasıtasıyla düğümlere aktarılıp çalıştırılması gerçekleştirilmiştir.

Benzer çalışmalardan farklı olarak, güvenilirliğin artırılmasına yönelik olarak çoklu çalıştırma, birebir yedekleme ve havuz yedekleme yöntemleri incelenmiştir. Havuz yedekleme yönteminde diğer yöntemlere göre daha az sayıda düğümün sürdürülebilirlik kümeleri olarak işaretlenip bekletilmesi, bununla beraber diğer yöntemlerden daha iyi güvenilirlik sağlanması, sistemin güvenilirliğinin sağlanması noktasında yeterli imkanı sağlamış olup, tez kapsamında tercih edilmiştir.

Diğer taraftan verimi artırmaya yönelik olarak, dosya aktarımlarında, MD5 bazlı kontroller konularak dosya aktarımı optimize edilmiş ve aynı dosyaların tekrarlı aktarımı engellenmiş, tasarlanan önceliklendirme algoritması ile düğümler önceliklendirilmiş, böylece düğümlerde çalıştırılacak uygulama alt görevlerine öncelik değeri atama imkanı verilmiş, düğümlerin homojen kullanımı sağlanarak yaşlanma oranları dengelenmiş, böylece sistemin ilk bozulma anı ötelenmiştir.

Haberleşme altyapısının seçimine yönelik çalışmada, Berkeley Soketleri, Mesaj Aktarım Arabirimi, Uzak Yordam Çağrısı ve Dağıtık Paylaşımlı Bellek yapıları incelenmiş, hem kullanılacak atanmış bilgisayarların farklı yapılandırma gereksinimleri, hem de başarımlar ekseninde yapılan araştırma sonuçlarına göre, geliştirilen uygulamada bilgisayar kümeleri mimarisinde çokça kullanılan Mesaj Aktarım Arabirimi yerine TCP / IP tabanlı Berkeley Soketleri tercih edilmiştir.

Haberleşme altyapısı olarak Berkeley Soketlerinin seçilmesinden dolayı, aktarılacak istenen nesnelere ya da dosyalara ilişkin verilerin serileştirilerek mesaj yapısına eklenmesi ve karşı tarafta aynı verinin mesajdan ayrıştırılarak ilgili nesnelerin ya da dosyaların oluşturulması işlemine ilişkin mesajlaşma yapısı ayrıca tanımlanmış ve geliştirilen Merkez Yönetim Yazılımına entegre edilmiştir.

Geliştirilen yazılıma referans olması açısından, OSCAR, ROCKS, WareWulf, xCAT ve Debian Küme Bileşenleri incelenmiş, farklı yönleri karşılaştırılmıştır.

Merkez Yönetim Yazılımı, uygulamanın ayrıştırılarak düğümlerde çalıştırılmasına ilişkin tüm süreci dağıtıcı, toplayıcı(lar) ve düğüm(ler) den oluşan üç ana bileşenle gerçekleştirmektedir.

Dağıtıcı tüm süreci yönetmek yerine sadece toplayıcı ve düğümleri yönetmekte, böylece hem tüm sürecin yönetimi kolaylaşmakta hem de yazılım içerisinde bileşen bazlı yalıtım sağlanmaktadır. Toplayıcı ve düğümler arası aktarım dağıtıcıdan bağımsızlaştığı için haberleşme ve senkronizasyon gibi ek işler ortadan kalkmıştır.

Toplayıcı, uygulamanın bulunduğu bilgisayarda çalışıp, uygulamaya ilişkin paralelleşmiş alt görevlerin tanımlandığı kural dosyalarını çözmekte ve alt görevlerin düğümlere aktarımını sağlamaktadır.

Düğümler ise toplayıcılardan gelen kural ve kuralla ilgili dosyaların çalıştırılmasını sağlamaktadırlar.

Düğümlerde çalıştırılacak uygulamaya ilişkin dosyalar ve değiştirge kümeleri gibi tüm gereksinimler uygulama sahibi tarafından kural listesi halinde tanımlanmakta ve kural dosyalarında barındırılmaktadır. Kurallar dağıtımı yapılacak uygulamanın içeriğine dair bir bilgi barındırmazlar, uygulamaya bakışları sadece dosya ve değiştirge listeleri

ve bunların alıřtırılma řekillerinden ibarettir. Dolayısıyla merkez ynetim yazılımının uygulamaya olan saydamlıđını sađlamıř olurlar.

Sistemin sınanması iki farklı bilgisayar ve drt farklı atanmıř bilgisayar kullanarak gerekleřtirilmiřtir. Sınanan atanmıř bilgisayarlar sırasıyla, Raspberry PI Model B, Raspberry PI 2 Model B, BeagleBone Black Model C ve HardKernel ODROID U3 olup maliyetleri, referans bilgisayarın yaklaşık %6-11 arasındadırlar.

Sistemin sınanması  ařamada gerekleřtirilmiřtir. nce referans bilgisayarda uygulama alt grevleri paralel olarak alıřtırılmıř ve tek iřlemci ve tek depolama birimi kullanımı gibi seri alıřma zorunluluklarından dolayı, toplam bařarımın dđm sayısı ile orantılı artmadıđı grlmřtir.

Ardından atanmıř bilgisayarlarla aynı sına ma tekrarlanmıř, bu sefer toplam bařarımın dđm sayısı ile orantılı arttıđı grlmřtir. Buradan ıkan sonu, seri alıřmaya ynelik tek etken olan ađdan veri aktarımının sistem bařarımına etkisinin ok az olduđudur.

Sonrasında temin edilen donanımlar teker teker sınanmıř ve elde edilen sına ma sonularında, referans bilgisayara gre sırasıyla %4, %21, %11 ve %48'lik bařarım sonuları elde edilmiřtir.

Son olarak yapılan maliyet analizinde, referans bilgisayar ile atanmıř bilgisayarların bařarım / fiyat oranları karřılařtırılmıř, atanmıř bilgisayarların referans bilgisayara alternatif olup olmadıkları sınanmıřtır. Sına ma sonularında, referans bilgisayara gre sırasıyla %58, %289, %156 ve %314'lk bařarım / fiyat oranları elde edilmiřtir. Buna gre ilk versiyon Raspberry PI atanmıř bilgisayarı hari, tm atanmıř bilgisayarlar referans bilgisayardan daha iyi bařarım / fiyat oranı gstermiřlerdir.



# **DISTRIBUTED ARCHITECTURE DESIGN AND MANAGEMENT SYSTEM DEVELOPMENT**

## **SUMMARY**

Distributed System architecture has emerged as a result of the efforts focused on solving the time and space issues of large-scale applications. Prior architectures had taken hardware (multi-processor systems, multi-core processors, etc.) or software (client / server etc.) based approaches to speed up computationally demanding applications, however, these approaches could not keep up with the growth in application scale. On the other hand, achieving the target performance with a constant number of server computers results in high cost solutions.

Recent increases in the use of smart phones and tablet computers have triggered substantial improvements in the cost / performance ratios of embedded grade processors and memory components. This observation has led to the idea that it can be possible to build systems with better cost / performance ratios compared to high-end server based systems using existing or custom designed embedded systems.

The goal of this work is to divide a large-scale application into small chunks of work that are executed on embedded computers and compare the cost / performance ratio of the resulting system with the existing server based systems. An additional advantage of the resulting system is that the increased number of computers that constitute the system helps distribute the risks associated with system sustainability across a larger number of sub-systems, improving the overall system reliability and efficiency. In the literature survey conducted on distributed systems, mainframe, client/server, distributed objects, computer cluster and grid architectures have been examined, advantages and disadvantages of these architectures have been analyzed. Based on the conducted work, it has been concluded that since the target applications are to be run with high performance and since embedded computers have limited computation resources, the architecture that is best suited for large scale applications is computer clusters, therefore it is selected as the preferred architecture in the thesis.

However, computer cluster architecture is not without shortcomings, the most important of which is their lack of platform flexibility. The application to be executed on the computers must be compatible with the processor architecture and the operating system available on the cluster nodes. Within the context of computer systems, platform independence and performance are contradicting design goals, that is, platform independence comes with an apparent performance loss. Since the main goal of this thesis is to increase performance, platform independence will not be taken into account.

Computer cluster architecture is based on a central management software on the manager node managing all the nodes present in the cluster to distribute and execute subtasks of the application. The work described in this thesis consists of a central management software along with a matrix multiplication based test application. The

resulting control software was used to distribute and execute the matrix multiplication work among nodes.

Different from the previous works on this topic, multiple execution, one-to-one backup and pool backup approaches have been examined to increase system reliability. Pool backup approach marks fewer nodes to be kept as sustainability nodes compared to the other approaches, however, still provides better reliability. As a result, pool backup was chosen as the preferred approach in this thesis.

On the other hand, system efficiency was increased by (1) optimizing file transfers by avoiding redundant transfers via MD5 based controls, (2) using the scoring algorithm designed as part of this thesis to prioritize the nodes providing a mechanism to assign priority scores to subtasks to be executed at nodes, (3) homogenizing use among all nodes to equalize weathering rates and delay the first breakdown.

Regarding the choice of communication framework, Berkeley Sockets, Message Passing Interface, Remote Procedure Call and Distributed Shared Memory approaches have been analyzed. Considering the embedded computers to be used can have varying configuration requirements and in the light of performance tests TCP/IP based Berkeley Sockets was preferred instead of Message Passing Interface, which is typically used in computer clusters.

As a result of using Berkeley Sockets, a new messaging structure was designed, implemented and integrated into the central management software. This new structure serializes the objects and files to be transferred, adds to the message structure, and on the receiving side extracts these from the message and reconstructs the objects and files.

OSCAR, ROCKS, WareWulf, xCAT and Debian Cluster Components were studied and compared to each other to serve as a reference to the developed software.

The central management software achieves partitioning the application into subtasks and executing these subtasks on nodes by three main components, namely, distributor, collector(s) and node(s).

Instead of orchestrating the whole process, the distributor only manages the collectors and nodes, simplifying the management of the process and providing component level insulation. Since the transfers between the collector and the nodes are independent of the distributor, there is no need for messaging and synchronization mechanisms between these units.

The collector simply processes the rule files that reside on the computers where the application subtasks are located and achieves the transfer of the subtasks to the nodes.

Finally, the nodes use the rules and associated files coming from the collector to execute their jobs.

All the requirements such as the files that are associated with the subtask to be executed at a node and the parameters are defined as a rule list by the user and stored in rule files. The rules have no specific information about the application to be distributed among the nodes. From the system's point of view, the application is simply a bunch of files, parameter sets and the way the subtasks are to be executed. This guarantees that the central management software is completely agnostic to the application to be executed.

The proposed system was tested with two desktop computers and four different types of embedded computers. The embedded computers are Raspberry PI Model B, Raspberry PI 2 Model B, BeagleBone Black Model C and HardKernel ODROID U3. The costs of these embedded computers are within 6-11% of the reference computer.

The tests were conducted in three stages. First, the application was run on the reference computer with all the subtasks parallelized. It was observed that the performance did not scale with the number of nodes due to serial execution limitations such as having a single processor and single storage unit.

Next, the same application was run on the embedded computers with all the subtasks parallelized and the performance was observed to scale with the number of nodes. This observation leads to the conclusion that the data transfer over network, which is the only serializing factor for this scenario, has negligible effect on the overall performance.

Then all the embedded computers were tested separately and their performances relative to the reference computer were measured as 4%, 21%, 11% and 48%, respectively.

Finally, a cost analysis was conducted where the cost / performance ratios of the embedded computers were compared to the reference computer to see if these embedded computers can be considered as an alternative to the reference computer. The resulting cost / performance ratios were found to be 58%, 289%, 156% and 314%, respectively. Based on these results, all embedded computers except for first version of Raspberry PI have better cost / performance ratios compared to the reference computer.



## 1. GİRİŞ

Günümüzde bilgisayarların hızını artırmak için, tüm devre içindeki yapıların küçültülmesi yönünde çalışılmaktadır. Silikon devre üretim sürecinin fiziksel sınırlamalarından dolayı, bu çalışmaların sınırlara dayandığı bilinmektedir. Bu nedenle hız artışının da sınırına yaklaşılmıştır. Ayrıca, işlemci hızının artırılması beraberinde güç tüketimini de artırmaktadır. Bu durumda işlem hızını artırmak için paralel mimarilerin önemi artmaktadır. [1]

Bugün özel ve genel amaçlı üretilen hemen hemen tüm işlemciler çok çekirdekli mimarilerle tasarlanmaktadır. Çok çekirdekli mimarilerin baskın tasarım tercihi olması, uygulamaların paralel çalışmaya uygun yöntemler ile geliştirilmesi yönünde bir akımın başlamasına vesile olmuştur.

İşlemci mimarilerinde çoklu çekirdek yapısına geçilmesiyle beraber aynı anda birden fazla işlemin birbirlerinden bağımsız çalışması mümkün olmuş ve bu sayede uygulamaların kendi içlerinde bağımsız çalışabilecek alt görevlerinin işlemcinin farklı çekirdeklerine dağıtılması yoluyla daha hızlı çalışmasının yolu açılmıştır.

Özellikle büyük ölçekli uygulamaların paralelleştirilmesi ve bu uygulamaların sunucu görevi gören çok çekirdekli bilgisayarlarda çalıştırılması, geliştirilen uygulamaların işlem bitirme zamanını önemli ölçüde azaltsa da hem üzerinde çalışılan verinin boyutunun önemli oranda artması hem de uygulamaların sahip olduğu bileşenlerin dışarıyla olan etkileşimlerinin sistemi yavaşlatma olasılığından dolayı hedeflenen işlem zamanının yakalanamamasına neden olmuştur.

Bunun yanında büyük hacimli işlemlerin belirli bir sürede bitirilmesine yönelik olarak tasarlanan sunucu bilgisayarları piyasadaki sayıları da göz önüne alındığında herkesin ulaşabileceği fiyatların çok ötesinde konumlandırılmış, buna karşılık beklenen başarımın yine de gerisinde kaldıkları gözlenmiştir. Ayrıca sunucu bilgisayarların uzun süreli çalışmalarından kaynaklanan ısı vb. kaynaklı olası sorunlar yapılan işin

tamamının belirli sürede askıda kalmasına neden olmakta, çözüme yönelik olarak yapılan periyodik bakımlar ek maliyet oluşturmaktadır.

Sonuç olarak büyük ölçekli işlemlerin belirli bir zamanda bitirilebilmesi mevcut çok çekirdekli sunucu tabanlı çözümler yerine **dağıtık mimariler** vasıtasıyla birden fazla bilgisayara ağ üzerinden yayılarak gerçekleştirilmektedir. Dağıtık mimariler, kendi içinde bir çok alt alana ayrılmış olup, uygulamaların çalıştırılma süreçlerini farklı noktalardan ele alıp gerekli iyileştirmeleri sağlamaktadırlar.

Son zamanlarda akıllı telefon ve tablet bilgisayarların yaygınlaşması ve pazarın büyümesinden dolayı atanmış sistemlere yönelik donanım / işlemci teknolojilerinde hızlı bir gelişim gözlenmiştir. Bunun paralelinde işlemci fiyatlarında da ciddi bir azalma gerçekleşmiştir. Bu gözlemler ışığında, büyük ölçekli uygulamaların, dağıtık mimarilerin alt alanlarından biri olan bilgisayar kümeleri mimarisi ile tasarlanmış bir atanmış sistemler bütünü vasıtasıyla belirgin bir başarımlar / fiyat üstünlüğüyle çözülebileceği fikri oluşmuş ve teze esin kaynağı olmuştur.

## 1.1 Tezin Amacı

Teze konu olan çalışma, kendini farklı veri kümeleriyle çok sayıda tekrar eden alt görevlere sahip büyük ölçekli örnek bir uygulamanın, doğası gereği paralelleştirmeye uygun olmasının üstünlüğünü de kullanarak, dağıtık sistem mimarisi vasıtasıyla ayrıştırılıp atanmış bilgisayarlarda çalıştırılmasını sağlamaktır.

Atanmış bilgisayarlar, belirli bir görevi gerçekleştirmek üzere, üzerinde sınırlı sayıda bileşene ve bireysel bilgisayarlara oranla düşük işlemci gücüne sahip, boyutça küçük bilgisayarlardır. Sahip oldukları bileşenlerin sınırlı sayıda olması ve düşük hızda çalışmaları, harcadıkları gücün az olmasını sağlamakta, bu sayede fansız tasarımlara sahip olabilmektedirler.

Atanmış bilgisayarların işlemci gücü her ne kadar bireysel bir bilgisayarın altında kalsa da sağladığı başarımlar ile çok ciddi bir fiyat üstünlüğü sağlamaktadır. Bu nedenlerden dolayı yeterli sayıda atanmış bilgisayarın birlikte çalıştırılmaları ile sorunun çözümünde hem hız hem de ekonomik üstünlük sağlayacağı sonucuna varılmıştır.

Seilecek daėıtık mimaride atanmış bilgisayarların kullanılması, hem görev ve veri tabanlı paralelleştirilmiş uygulamaların hem de atanmış bilgisayarların sahip olduėu işlemci mimarisine göre dizi ya da vektör paralelliliėini kullanan uygulamaların alıřmasına imkan vermektedir.

Ayrıca atanmış bilgisayarların birlikte ve baėımsız alıřtırılması sistemin bütününe ilişkin sürdürülebilirlik üstünlüėü de sağlamaktadır. Şöyle ki, atanmış bilgisayarların herhangi bir ya da birkaç tanesinde meydana gelebilecek fiziksel bir sorun otomatik olarak diėerleri tarafından yedeklenebilmektedir, böylece uygulama her türlü dış etkene raėmen başarılı bir şekilde alıřabilmektedir.

Kullanılacak atanmış bilgisayarların yetkinliėi sınırlı olduėundan ve işlemci gücünün tez kapsamında maksimum oranda alıřtırılmak istenen uygulamaya ayrılması isteminden, özellikle daėıtık mimarilerle beraber ortaya ıkılmış CORBA, DCOM gibi daėıtık nesne mimarileri, sisteme olan maliyetleri düşünülerek tercih edilmemişlerdir. Bunların yerine temelinde daėıtık sistem mimarisinin alıřma alanlarından biri olan bilgisayar kümeleri (computer cluster) mimarisi referans alınacak, bilgisayarlar arası haberleşme arabirimi olarak ise soket mimarisi bazlı mesaj aktarım mimarisi kullanılarak daėıtık bir yazılım uygulaması (yönetim yazılımı) tasarlanacaktır. Sonrasında örnek bir uygulamanın bu mimari vasıtası ile ayrıştırılıp atanmış bilgisayarlarda alıřması sağlanıp başarımları ölçülecek, elde edilen sonuçlar masaüstü ve sunucu bilgisayarlarda elde edilecek sonuçlarla karşılaştırılacaktır.

## **1.2 Literatür Arařtırması**

Bilgisayar kümeleri mimarisi, geçmişte çok uzun zaman süreceėi düşünülerek, gerçekleştirilemeyeceėi düşünülen uygulamaların gerçekleştirilmesini mümkün kılan bir yapı sunmasıyla beraber, çok çeřitli alanlarda kendisine kullanım sahası bulmuştur. Uygulama sahaları ne kadar farklı olursa olsun, paralelleşmeye yatkın uygulamalar birbirlerinden seri alıřılması gereken kısımların oranı ve türü noktasında ayrılmaktadırlar. Belirli alanlarda seri alıřılması gereken konular uygulama sahasının tipine göre farklı yaklaşımlarla özölmeye alıřılmıştır. Örnek olarak incelenen arařtırmalardan bazıları şunlardır;

### **1.2.1 Biyomedikal mühendisliğinde paralel ve dağıtık işleme teknikleri**

Biyomedikal alanında paralel dağıtık işleme tabanlı yapılan bu çalışmada [2], işlenecek olan resimler "sahip - köle" olgusunda sahip olarak tanımlanan bilgisayar tarafından bağımsız olarak kölelere dağıtılacaktır. Süreç, bilgisayar kümeleri mimarisine yakın bir modellemeye sahip olup, sahip olarak tanımlanan bilgisayarda çalıştırılacak bir yönetim yazılımı vasıtasıyla kölelerin varlığı kontrol edilecek ve müsait durumdakilere işlenecek olan resimler gönderilecektir.

Süreçte, köle bilgisayarlardan beklenen ana iş, tek bir resmin piksel piksel işlenmesi olduğundan, sürecin köle bilgisayarda başlayabilmesi için resmin köle bilgisayara aktarımının tamamlanması gerekmektedir. Bu durumda süreçte çözülmesi gereken en büyük risk resmin aktarımının doğru şekilde gerçekleşmesidir.

Çalışmanın detayında görüleceği üzere, aktarım resimlerin bloklara bölünüp köle bilgisayarlara aktarılması şeklinde gerçekleşmektedir. Seçilen yöntem standart bir aktarım şekli olup, herhangi bir yedekleme ve güvenli aktarım şablonu tercih edilmemiştir. En basit haliyle, sorun durumunda blokların gönderimi tekrarlanmaktadır. Dolayısıyla çalışmada tüm köle bilgisayarların sağlıklı çalıştığı önkoşulu ile süreç işletilmiştir.

Süreçte yapılan en önemli optimizasyon, aktarılacak olan resimlerin LZO sıkıştırma yöntemi [3] ile sıkıştırılıp sürecin hızlanıp hızlanmadığının gözlenmesidir. Zira buradaki mübadele, sıkıştırma ve geri açma işlemlerinde kaybedilen zamanın aktarım süresinde sağlanan iyileştirme ile karşılanıp karşılanamadığı noktasındadır.

### **1.2.2 Cluster Juggler - Bilgisayar kümesi tabanlı sanal gerçeklik**

Sanal Gerçeklik alanında yapılan bu çalışmada [4], paylaşımlı bellek modelinde çalışan bir sanal gerçeklik uygulaması olan "VR Juggler" uygulaması, bilgisayar kümeleri mimarisi vasıtasıyla hızlandırılmaya çalışılmıştır.

Sanal gerçeklik uygulamalarının temel çalışma gereksinimleri; bir giriş biriminden verinin okunması, işlenmesi ve belirli bir çerçeve/s hızında çıkışa aktarılması olarak özetlenebilir. Dolayısıyla çalışma kapsamında tasarlanan mimaride, düğümlerin

görüntü verisini alıp, yorumlaması ve sanal gerçekliğe özel bazı işlemlerden geçirmesi gerekmektedir.

Çalışmada üzerinde durulan ana tema, düğümlerin işlevsel bazda özelleştirilip ne oranda hız artışının sağlandığı noktasındadır. Bu nedenle, önce düğümlerin tek başlarına bütün görüntü işleme süreçleri sınanmış sonrasında düğümler işlevsel bazda ayrıştırılarak, örnek olarak bazı düğümler sadece yorumlama sürecini işlerken diğer bazı düğümler sanal gerçekliğe özgü süreci işlemişlerdir.

Temel olarak karşılaşılan en önemli sorun, resim çerçevelerinin düğümler arasındaki senkronizasyonunun sağlanması noktasında oluşması ve bu sorunun oluşturulan bilgisayar kümesi sisteminden yalıtılamamış olmasıdır. Diğer bir deyişle sanal gerçeklik uygulaması düğümler arası senkronizasyon işleminde devreye girmek zorunda kalmıştır.

### **1.2.3 Zemin-Yapı etkileşimi sorunlarının bilgisayar kümeleriyle paralel çözümü**

İnşaat Mühendisliği alanında yapılan bu çalışmada [5], Zemin-Yapı Etkileşimi sorunlarına dinamik çözüm üreten örtük Newmark integral algoritmasının paralelleştirilmesi ve bilgisayar kümeleri vasıtasıyla hızlı çözüm üretilmesi sağlanmıştır.

Çalışma kapsamında, geliştirilen uygulama paralelleştirildikten sonra düğümler mesaj aktarım arabirimi (MPI) vasıtasıyla haberleştirilmiş ve uygulama sahip ve köle düğümlerde çalıştırılmıştır. Sahip düğüm giriş verilerinin okunup köle düğümlere aktarılmasından sorumlu olup, köle düğümler uygulamaya ilişkin görevi çalıştırmaktadırlar. Burada göreve ilişkin ana iş, algoritmanın sahip olduğu veri modelinin belleğe yüklenip analizin yapılmasından ibarettir. Dolayısıyla, sahip düğüm modele ilişkin tüm veriyi tek seferde köle düğüme aktarmak zorundadır.

Yapılan çalışma algoritmanın başarımlar kriterleri ekseninde çalıştırılması olduğu için, tüm köle bilgisayarların sağlıklı çalıştığı önkoşulu ile süreç işletilmiştir.

### **1.2.4 RPiCluster - Raspberry Pi tabanlı Beowulf kümesi**

Matematiksel işlemlerin hızlandırılması amaçlı yapılan bu çalışmada [6], Raspberry PI donanımları kullanarak Beowulf Kümesi gerçekleştirilmiş ve başarımları

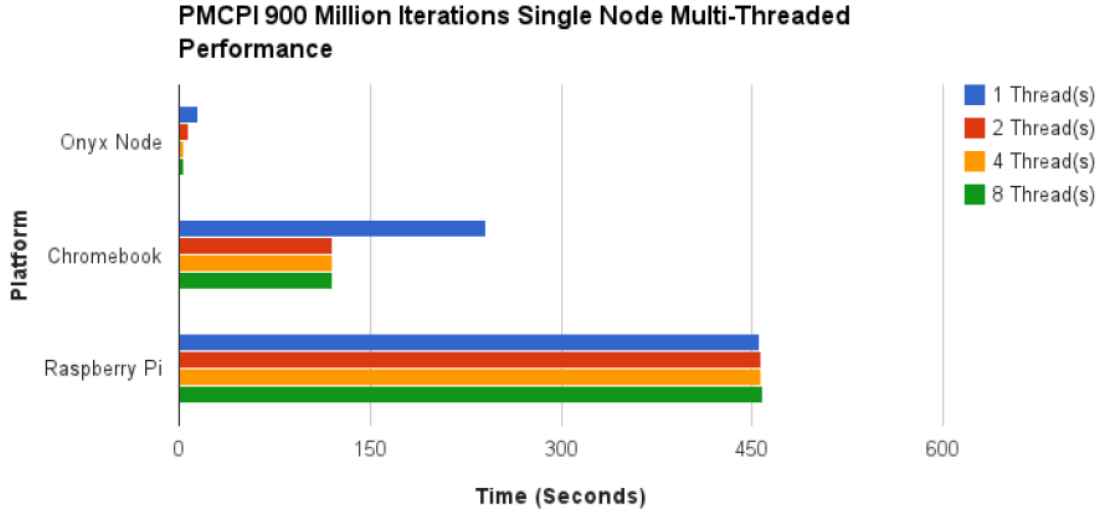
değerlendirilmiştir. Boise State Üniversitesi, Onyx Laboratuvarında gerçekleşen çalışmada, mevcut 32 düğümlü Onyx kümesine, Raspberry PI donanımları kullanılarak alternatif üretilmiş ve başarımlar / fiyat oranları karşılaştırılmıştır. Sınamada işlem olarak Monte Carlo algoritması kullanılarak **Pİ** sayısının üretilmesi hedeflenmiştir, dolayısıyla önce Pİ sayısının üretiminin paralelleştirilmesi konusu ele alınmıştır.

Sınamada kullanılan donanımların özellikleri Çizelge 1.1’de [6] belirtilmiştir.

**Çizelge 1.1:** Kullanılan donanım bileşenleri

|              | İşlemci             | Çekirdek | Hız     | Fiyat  |
|--------------|---------------------|----------|---------|--------|
| Onyx Düğümü  | Intel Xeon E3-1225  | 4        | 3.1 GHz | 1000\$ |
| Chromebook   | Samsung Exynos 5250 | 2        | 1.7 GHz | 250\$  |
| Raspberry PI | BCM2708 ARM11       | 1        | 700 MHz | 45\$   |

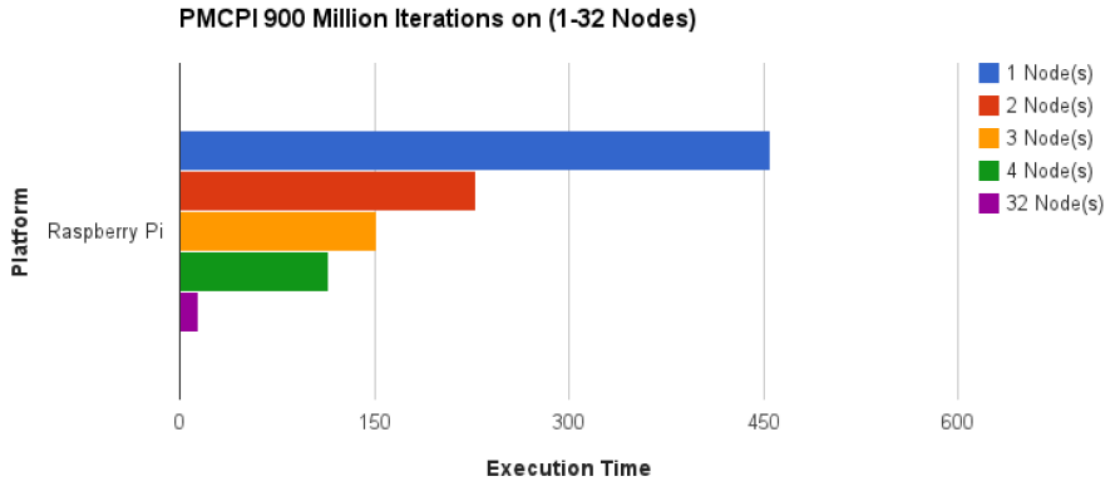
Sınama sonucu oluşan başarımlar karşılaştırması Şekil 1.1’de [6] gösterilmiştir.



**Şekil 1.1:** RPiCluster başarımlar grafiği.

Raspberry PI’lerin 1 ile 32 düğüm arasındaki başarımlar karşılaştırmasında ise Şekil 1.2’deki [6] gibi bir sonuç çıkmıştır. Buna göre düğüm sayısı ile başarımlar arasında doğru bir orantı vardır, yani sisteme eklenen her düğüm, sistemde yaklaşık %100 başarımlar artışı sağlamaktadır.

Onyx düğümü, sınamayı **3.85 s** de bitirirken Raspberry PI **456 s** de bitirmiştir. Buna göre, Onyx düğümü fiyat olarak Raspberry PI’nin yaklaşık **23** katı bir fiyata sahip iken başarımlar olarak **118** katı bir başarımlara sahiptir. Bu durumda Raspberry PI’nin deneme kapsamında uygun bir çözüm olmadığı ortaya çıkmıştır.



**Şekil 1.2:** RPiCluster çalışma zamanı grafiği.

### 1.2.5 Mikroişlemci mimarisinin dağıtık bellekli süper bilgisayarların hızlanmasına etkisi

Amdahl ve Gustafson'ın argumanlarının sınamasına yönelik olarak yapılan çalışmada [7] belirli bir sayıya kadar sisteme eklenen işlemcilerin gösterdiği başarımla, Amdahl'ın argumanında belirttiği yavaşlama ile açıklanabiliyorken, o noktadan itibaren ters bir etki göstermiştir. Profesyonel bir karşılaştırma uygulaması (CRAFT) [8] ile yapılan ölçümlerde her bir karşılaştırma döngüsünün sürdüğü zaman (hesaplama + haberleşme) işlemci sayılarına göre karşılaştırılmış ve Çizelge 1.2'deki [7] sonuç elde edilmiştir.

**Çizelge 1.2:** CRAFT karşılaştırma sonuçları.

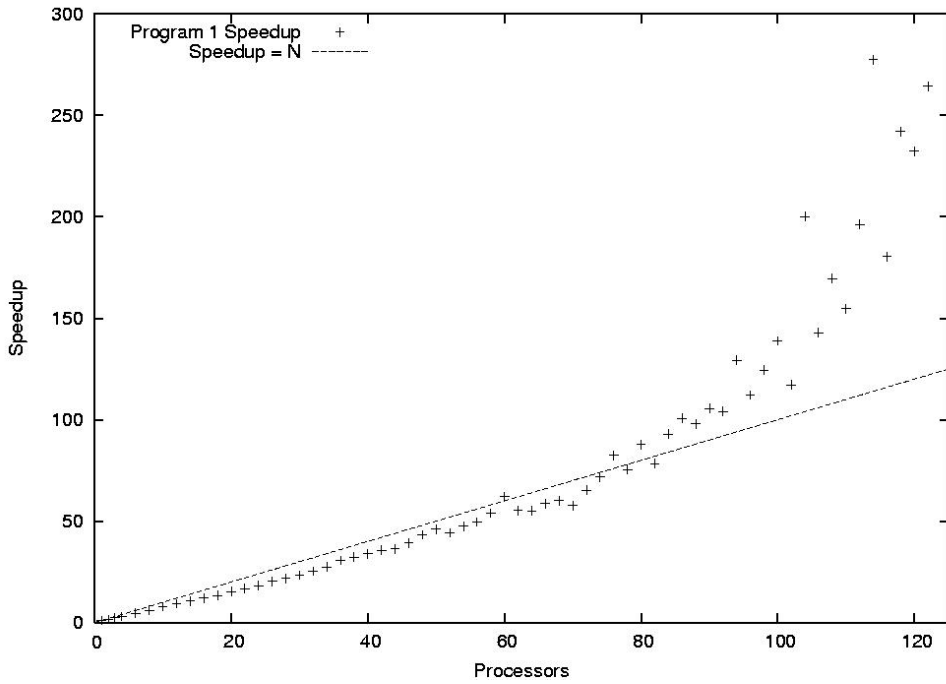
| İşlemci Sayısı | Yineleme Zamanı | Teorik Maksimum Hızlanma | Ölçülen Hızlanma |
|----------------|-----------------|--------------------------|------------------|
| 32             | 138.2           | 1                        | 1                |
| 64             | 69.7            | 2                        | 1.98             |
| 128            | 32.9            | 4                        | 4.2              |
| 256            | 16.2            | 8                        | 8.53             |

Karşılaştırma sonucuna göre, işlemci sayısını 32'den 64'e çıkarmak %1'lik bir yavaşlama yaratırken 128 ve 256 işlemciye çıkarılan durumlarda hızlanma sırasıyla 4,2 ve 8,53 kat olmaktadır. Çalışmada üzerinde durulan ispat, Amdahl'ın argumanında bahsettiği; karşılaşılan sorunun büyüklüğünün sistemdeki işlemci sayısı ile orantılı olarak artmasına karşın, seri çalışan kısmın etkisinin sistemdeki bileşenlerin cinsine ve hızına göre sorunun büyüyebileceği gibi küçülebileceği üzerinde olmuştur. Örnek

olarak uygulamanın giriş verisinin paralelleştirilip işlenmesi süreci, bilgisayar/ işlemci sayısı arttıkça verinin öncekine oranla daha büyük miktarının bellek / cep bellek üzerinde bir seferinde barındırılmasını sağlamaktadır. Dolayısıyla giriş verisiyle ilişkili işlemlerin hızı sistemdeki düğüm sayısı arttıkça, azalacağı yerde artmaktadır.

Çalışmada üç farklı uygulama parçası ile işlemcilerin cep belleklerinin sistemin toplam çalışma hızına olan etkisi araştırılmıştır.

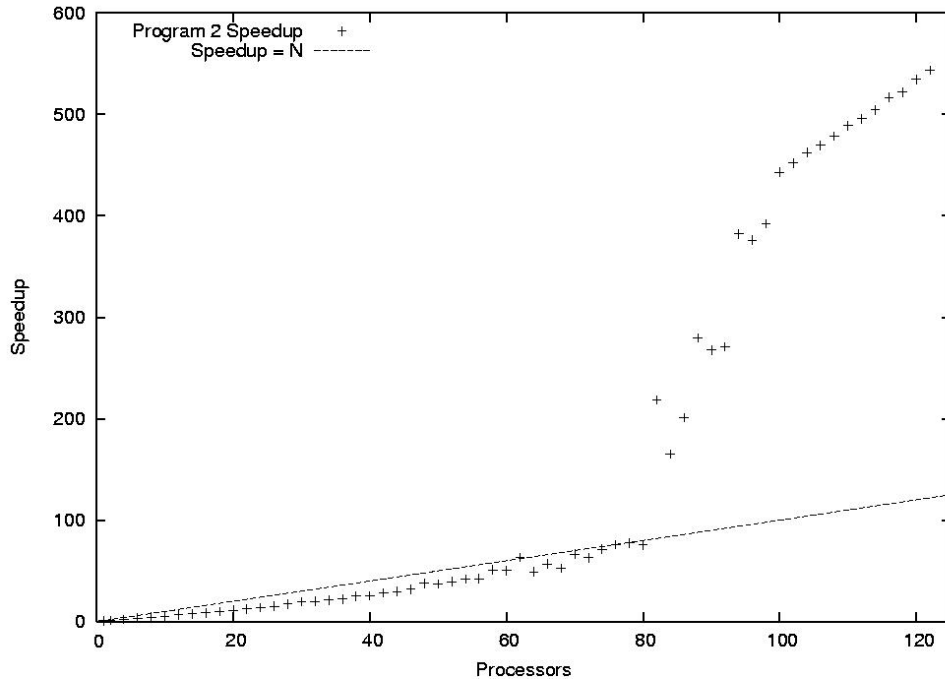
İlk uygulama, matematiksel bir işlemin geniş bir veri kümesi ile işlenmesinden ibarettir. Uygulamanın işlemcilere dağıtılması ile beraber, her bir işlemciye düşecek veri miktarı işlemci sayısı ile ters orantılı olarak azalmakta ve veri boyutunun işlemcinin cep belleğine sığıldığı noktada cep bellek ıskalması (*cache miss*) olmadığından belirgin bir başarımlı artışı sağlanmaktadır. Uygulamaya ilişkin hızlanma grafiği Şekil 1.3'deki [7] gibidir.



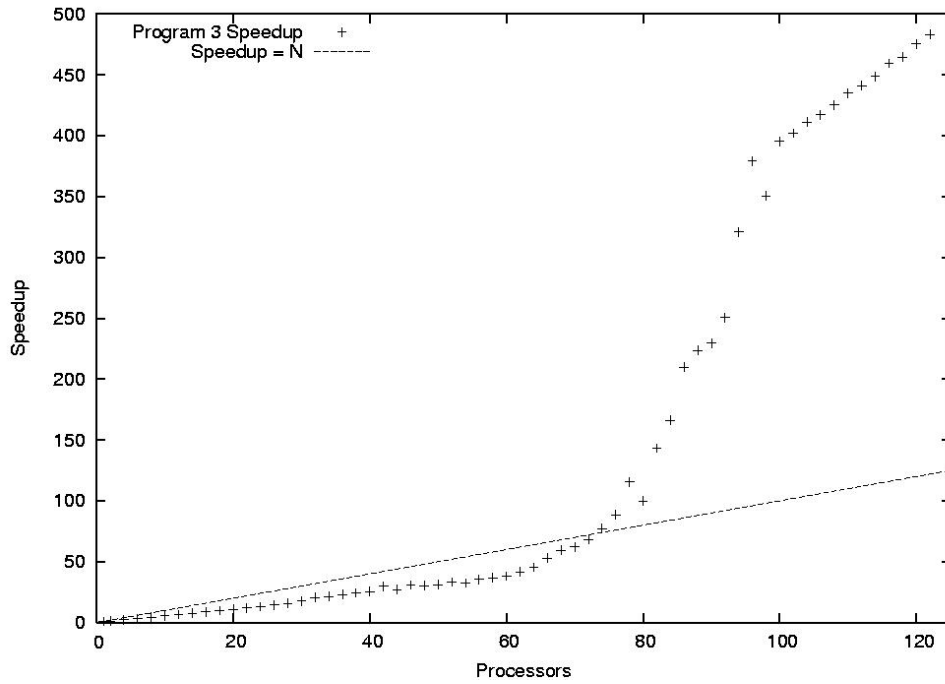
**Şekil 1.3:** Uygulama 1 hızlanma grafiği.

İkinci uygulamada verinin belleği kullanımı ardışıl değil, karışık yapılmaktadır. Bu şekilde cep bellek ıskalmasının oranı arttırılmaktadır. Uygulamaya ilişkin hızlanma grafiği Şekil 1.4'deki [7] gibidir.

Üçüncü uygulamada ise döngüde belli aralıklarla sonuçlar başka bir düğüme haberleşme ortamı üzerinden gönderilmektedir. Uygulamaya ilişkin hızlanma grafiği Şekil 1.5'deki [7] gibidir.



**Şekil 1.4:** Uygulama 2 hızlanma grafiği.



**Şekil 1.5:** Uygulama 3 hızlanma grafiği.

Çalışmada varılan sonuca göre, uygulama bir ve iki'nin karşılaştırmasından, işlemcilerin cep bellek kullanımının başarım üzerindeki katkısı, verinin tamamının cep belleğe girmeye başlamasından itibaren çok hızlı oranda artmaktadır. İkinci uygulama cep bellek ısıkalmasının fazla olmasından dolayı, ilk uygulamaya göre daha az bir

lineer davranış sergilemiş ve ilk uygulamadan farklı olarak hızlı artış ancak yaklaşık 80 işlemciden sonra olmuştur.

Üçüncü uygulamada varılan sonuca göre haberleşmenin etkisi işlemci sayısı ile orantılı artmaktadır, sistemin üzerinde genel olarak sabit bir etkisi vardır.

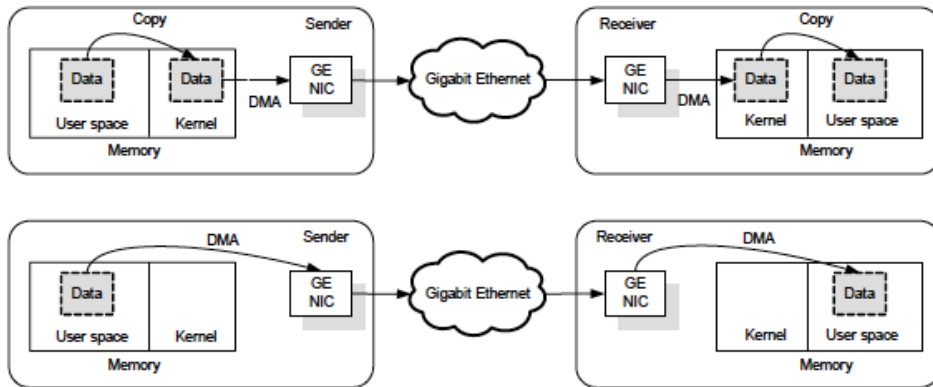
### 1.2.6 HPC sistemlerinde ağ başarımları çalışmaları

Bu çalışmada [9], bilgisayar kümeleri mimarisi ile kurulan sistemlerde, TCP, UDP ve MPI protokollerinin çeşitli senaryolar için veri aktarım hızları karşılaştırılmıştır. Çalışma iki farklı mimari üzerine kurulu bilgisayar kümesi üzerinde gerçekleştirilmiştir; Intel Xeon ve Alpha mimarileri. Her bir Intel Xeon düğümü 4 çekirdekli 3GHz işlemci ve 2 GB bellek içerirken, Alpha düğümü 4 çekirdekli 833 MHz işlemci ve 4 GB bellek içermektedir.

Her iki sistem için de öncelikle verinin uygulamadan alınması, işletim sistemi çekirdeğinde işlenip ortama salınması incelenmiş, jumbo çerçeve ve sıfır kopyalama gibi teknikler ile sistemde ciddi bir başarımları artışı görülmüştür.

Jumbo frame incelemesinde, standart TCP başlığının sisteme olan olumsuz etkisi %5.64 iken bu oran jumbo frame kullanıldığında %0.95'e düşmektedir.

Sıfır kopyalama tekniğinde ise aktarılacak verinin çekirdeğe fazladan kopyalanması engellenmiş, doğrudan ethernet kartının tampon alanına aktarımı sağlanmıştır. Tekniğe ilişkin grafik, Şekil 1.6' da [9] gösterilmiştir.



Şekil 1.6: Sıfır kopyalama tekniği.

Sonrasında yapılan protokol karşılaştırmaları, engelli ve engelsiz olarak iki farklı senaryoda gerçekleştirilmiştir. Karşılaştırma sonucu çıkan sonuç Çizelge 1.3’de [9] gösterilmiştir.

**Çizelge 1.3:** Sistemlerin 64 KB’lık mesaj aktarım hızları.

| Protocol | Test Mode                          | Throughput (Mbps) |       | Sender CPU load |       | Receiver CPU load |       |
|----------|------------------------------------|-------------------|-------|-----------------|-------|-------------------|-------|
|          |                                    | Intel             | Alpha | Intel           | Alpha | Intel             | Alpha |
| TCP      | Unidirectional Blocking            | 941.5             | 519.8 | 22%             | 21%   | 22%               | 29%   |
|          | Unidirectional Non-blocking        | 941.5             | 517.5 | 38%             | 38%   | 34%               | 37%   |
|          | Bidirectional Blocking (ping-pong) | 854.9             | 422.3 | 22%             | 21%   | 22%               | 21%   |
|          | Bidirectional Non-blocking         | 1827.8            | 721.7 | 47%             | 39%   | 45%               | 39%   |
| MPI      | Unidirectional Blocking            | 932.3             | 312.2 | 20%             | 20%   | 24%               | 27%   |
|          | Unidirectional Non-blocking        | 932.1             | 311.8 | 24%             | 26%   | 37%               | 32%   |
|          | Bidirectional Blocking (ping-pong) | 796.6             | 287.1 | 26%             | 26%   | 26%               | 23%   |
|          | Bidirectional Non-blocking         | 960.8             | 316.5 | 31%             | 34%   | 35%               | 37%   |
| UDP      | Bidirectional Multiplexing I/O     | 1867.2            | 775.6 | 34%             | 27%   | 32%               | 27%   |

Çalışma sonunda elde edilen sonuca göre;

- İşlemci hızları ağ başarımını en fazla etkileyen değişirge olmuştur.
- Sorun durumunda en güvensiz haberleşme protokolünün UDP olduğu görülmüştür. Özellikle Alpha sisteminde yeterli soket tampon alanı olmasına rağmen paket kayıpları olmuştur.
- UDP protokolünün aktarıma getirdiği ek yük diğerlerine oranla daha az olduğundan sistem karakteristiğini en iyi yansıtan protokol olmuş, herşeyin düzgün seyretmesi durumunda en yüksek başarımlar UDP protokolü ile sağlanmıştır.
- MPI protokolü TCP protokolünün üzerine inşa edildiğinden, TCP ve UDP’ye göre protokol ek yükü daha fazladır.
- Tek yönlü aktarımlarda engelli ve engelsiz aktarım arasında belirgin bir fark görülmemekte, TCP ile MPI arasında sadece yaklaşık %2’lik bir başarımlar farkı oluşmaktadır.

- Çift yönlü aktarımlarda ise durum farklılaşmakta, engelliye göre engelsiz aktarımdaki hız artışı, TCP protokolü için %53 olurken bu artış MPI için %17’de kalmıştır.

### 1.2.7 Beowulf kümesi üzerinde paralel SAR işlemcisinin geliştirilmesi

Sentetik Aralıklı Radar (SAR) teknolojisi üzerine yapılan çalışmada [10], SAR için geliştirilen G2 ardişlemcisine paralel çalışabilme özelliği kazandırılmaya çalışılmıştır. SAR; mikrodalga tekniğiyle bir radar dizisinden alınan ardışıl radar resimlerinin birleştirilerek yüksek çözünürlüklü resim edilmesini sağlayan radar cinsidir. G2 işlemcisi radara gelen ham veriyi işleyerek en optimum çözünürlükte resmi elde etmeye yönelik olarak tasarlanmıştır. İşlem temelde iki kısma ayrılmış olup bunlar;

- **Menzil sıkıştırması** : Radarın yol aldığı istikametın dikine bakan yönden gelen verinin işlenmesidir, atım (*pulse*) sıkıştırması, hareket sıkıştırması ve girişim baskılaması gibi görevleri içerir.
- **İstikamet (Azimut) sıkıştırması** : Radarın yol aldığı istikametten gelen verinin işlenmesidir, menzil bükülüm (*curvature*) düzeltmesi gibi görevler içerir.

Çalışmada, toplanan resimler satır bazlı olarak kümelenecek ve her küme diğer kümelerden bağımsız işlenecektir. Kümelerin büyüklüğü gelen sinyalin gürültü oranına ve RFI filtresinin çalışma limitlerine göre tahmini olarak belirlenmektedir.

Parallelleştirme işlemi sekiz düğümlü bir beowulf kümesi oluşturarak gerçekleştirilmiştir. Düğümlerin donanım özellikleri;

- **Merkez yönetim yazılımının çalıştığı düğüm** : 1 adet çift işlemcili bilgisayar, Intel Pentium II 350 MHz, 512 MB bellek
- **Alt görevlerin çalıştığı düğümler** : 7 adet tek işlemcili bilgisayar, Intel Pentium II 350 MHz, 256 MB bellek

Düğümlerde işlenen alt görevler; Menzil sıkıştırması, İstikamet sıkıştırması ve Köşe dönüş algoritması olarak belirlenmiştir. Düğümlerin birbirleri arasındaki haberleşmesi ise MPI protokolü vasıtasıyla sağlanmıştır.

**Çizelge 1.4: İşlemci karşılaştırmaları.**

| İşlemci                        | Zaman | Hızlanma |
|--------------------------------|-------|----------|
| G2 İşlemcisi                   | 9344  |          |
| Teorik Hızlanma                | 1168  | 8        |
| Teorik Hızlanma (%60 Verim)    | 1947  | 4.8      |
| G2 Paralel İşlemcisi (8 Düğüm) | 1507  | 6.2      |

Çalışma sonucunda ulaşılan başarımlar, Çizelge 1.4’ de [10] gösterilmiştir.

Buna göre sekiz düğümlü kümede oluşan başarımlar tek düğüme göre yaklaşık 6 kat olsa da düğümlerin sahip olduğu işlemci hızları ve bellek kapasiteleri tüm verinin gerçek zamanlı işlenmesine yetmemektedir.

### **1.2.8 Küçük, düşük maliyetli, disksiz Beowulf kümelerinin başarımlar model**

Düşük maliyetli atanmış bilgisayarlarla gerçekleştirilen bu çalışmada [11], disksiz çalışma kipinin standart bilgisayar kümeleri mimarisinden ayrıştığı noktalar ele alınmış, karşılaşılan sorunlar incelenip çözüm yolları araştırılmıştır.

Disksiz donanımlar üzerine kurulu küme mimarisini diğerlerinden ayıran en önemli fark, ağ başarımının sistem üzerindeki etkisinin çok yüksek olmasıdır. Donanımlar üzerlerinde işletim sistemi ve uygulamaları barındıramadığından, sistemdeki tüm yazılımsal bileşenler donanımlara ağ dosya sistemleri üzerinden yüklenmektedir. Disksiz çalışmanın olumlu ve olumsuz yanları aşağıda sıralanmıştır;

Olumlu yanları;

- Yedekleme daha kolaydır, çünkü tüm dosyalar tek lokasyonda bulunmaktadır.
- Güvenilirliğin de aynı şekilde tek noktada yapılması yeterlidir.
- Disk olmadığı için donanım maliyeti düşüktür.
- Kurulumu daha kolay, tek disk kopyası ile işlem yapılmaktadır.

Olumsuz yanları;

- Ağ çöktüğünde düğümler ulaşılamaz ve çalışamaz durumda olmaktadırlar.
- Ağ üzerindeki diske erişim, yerel diske oranla çok yavaştır.

- Donanımın üzerinde yetersiz bellek olması durumunda, uygulama sonlanacaktır.
- Deneysel olarak bazı işletim sistemi çekirdekleri yetersiz bellek durumunda takas işlemini ağ üzerinden gerçekleştirebilmektedirler. Bu durumda ise yoğun bir haberleşme yüzünden uygulamanın çalışma hızı çok yavaşlayacaktır.

Başarım karşılaştırmalarında kullanılmak üzere Çizelge 1.5'deki [11] donanımlar tercih edilmiştir;

**Çizelge 1.5:** Kullanılan donanımlar.

| Hardware Factor         | Hardware Performance Levels |                |                 |
|-------------------------|-----------------------------|----------------|-----------------|
| CPU Speed               | 233MHz                      | 433MHz         | 1800MHz         |
| Node Memory (RAM)       | 128MB                       | 256MB          | 512MB           |
| Network Speed / Routing | 10 Mbps Hub                 | 10 Mbps Switch | 100 Mbps Switch |

Karşılaştırma ölçümlerinde Çizelge 1.6'de [11] görülen sekiz farklı matematiksel denklemin ölçümleri sırasıyla 32, 128 ve 256 işlemcili sistemler ile sınıf A, B ve C de görülen veri büyüklükleri ile gerçekleştirilmiştir.

**Çizelge 1.6:** Ölçümü yapılan matematiksel denklem ve sınıfları.

| Benchmark Code                | Class A            | Class B            | Class C  |
|-------------------------------|--------------------|--------------------|----------|
| Embarrassingly Parallel (EP)  | $2^{28}$           | $2^{30}$           | $2^{32}$ |
| Multigrid (MG)                | $256^3$            | $256^3$            | $512^3$  |
| Conjugate Gradient (CG)       | 14,000             | 75,000             | 150,000  |
| 3-D FFT PDE (FT)              | $256^2 \times 128$ | $512 \times 256^2$ | $512^3$  |
| Integer Sort (IS)             | $2^{23}$           | $2^{25}$           | $2^{27}$ |
| LU Solver (LU)                | $64^3$             | $102^3$            | $162^3$  |
| Pentadiagonal Solver (SP)     | $64^3$             | $102^3$            | $162^3$  |
| Block Tridiagonal Solver (BT) | $64^3$             | $102^3$            | $162^3$  |

Yapılan ölçümlerde elde edilen sonuçlarla yapılan başarım modellemesi üç ana faktörle değerlendirilmiştir;

- D    mlerin bellek kapasitesi : Yetersiz bellek durumuna eriřilmediĐi m  ddet  e k  menin bařarımına bir etkisinin olmadıĐı g  r  lm  řt  r. Fakat yetersiz bellek durumunda d    m  n bařarımı takas iřlemi y  z  nden   ok y  ksek miktarda d  řmektedir.
- AĐın y  nlendirme ve kapasite bařarımı : AĐ bařarımı, sistemin genel bařarımını EP   l  m   (rasgele sayı   retimi) hari     ok y  ksek oranda etkilemiřtir. Burada aĐ bařarımı iki noktada incelenmesi gerekmektedir; y  nlendirme ve kapasite bařarımları.

İlk yapılan   l  mlerde y  nlendirme elemanı olarak "*Hub*" se  ilmiř,   zellikle y  ksek hızlı bilgisayarlarla yapılan   l  mlerde,   l  mler 24 saatlik bir zaman diliminde tamamlanamamıřtır. Sisteme eklenen yeni d    mler ise bařarımı arttırmak yerine azaltmıřtır.

İkinci yapılan   l  mlerde y  nlendirme elemanı olarak "*Switch*" se  ilmiř,   l  mler birkaç saatte tamamlanabilmiřtir.

Kapasiteye y  nelik olarak yapılan   l  mlerde, kapasitenin k  me bařarımını etkilediĐi g  r  lm  ř,   zellikle kapasitenin yeterli olmadıĐı durumlarda sisteme eklenen d    mler k  menin bařarımını y  kseltmek yerine d  ř  rm  řt  r.

- İřlemci bařarımı :   l  m sonu  larında 16 d    ml   1.8 GHz'lik k  me, 16 d    ml   233 MHz'lik k  meye oranla sadece %260 daha iyi bařarım g  stermiřtir. Buradan   ıkan sonu  , iřlemci bařarımının yapılan   alıřma kapsamında bařarımı etkileyen birinci fakt  r olmadıĐıdır. Hatta yavaş aĐ elemanları kullanılan bazı   l  mlerde d    mlerin iřlemci hızlarını arttırmanın, k  menin bařarımını azalttıĐı g  r  lm  řt  r.

  alıřmadan genel olarak   ıkarılan sonu  , disksiz donanımlardan kurulan beowulf k  mesinin bařarımı birinci derecede aĐ bařarımına baĐlı olduĐudur.

### 1.3 Tezin Katkısı

Bilgisayar K  meleri mimarisinde ana uygulamaya iliřkin alt g  revleri ger  ekleřtirecek d    mlerin hızları, s  rd  r  lebilirlikleri ve bu ama  la kendilerine atanan yardımcı g  revler, d    mlere aktarılabilecek alt g  revlerin aktarım řekilleri, mimarideki

farklılaşmayı sağlamaktadır. Uygulama sahasının önemine göre, kimi sahalarda güvenilirlik ön plana çıkarken kimilerinde ise hız daha ön planda yer almaktadır.

Tez kapsamında hedeflenen katkı, hem belli oranda güvenilirliği sağlayan bir düğüm altyapısı oluşturmak hem de düğümlere aktarılabilecek verinin en optimum şekilde aktarımını sağlamak olarak özetlenebilir. Bu nedenle tezin katkısı iki ana başlık altında incelenecektir.

### **1.3.1 Güvenilirlik**

Uygulama sahalarının güvenilirlik gereksinimleri ve önemlerine göre düğümlerin çalıştırılması, düğümlerden çıkan sonuçların doğruluğu, ısı vb. gibi sebeplerden dolayı bazı düğümlerin bozulması durumunda, bu düğümlerin en verimli şekilde yedeklenmesi gibi konular, güvenilirliğinin sağlanması noktasında ele alınıp incelenmiştir. [12]

#### **1.3.1.1 Çoklu çalıştırma**

Özellikle güvenilirlik gereksiniminin en üst düzey olduğu uygulama sahalarında, düğümlerden çıkan sonuçların doğruluğunun garantilenmesi en önemli değişirge olarak ortaya çıkmaktadır. Bu çeşit uygulamalarda her bir alt görevin gerçekleştirilmesi birden fazla tek sayıdaki düğüm ile çoğullanmakta ve düğümlerden çıkan sonuçlar karşılaştırılarak, uygulamanın kritiklik seviyesine göre işlem yenilenmekte ya da çoğunluğun doğruladığı sonuç doğru sonuç olarak kabul edilip alt görevler gerçekleşmiş kabul edilmektedir.

Çoklu çalıştırmaya yönelik olarak, bir düğümün bozulma olasılığına "B" dersek;

$0 < B < 1$  olasılığında ve her düğümün eşit bozulma olasılığında; alt görevin doğru sonuç verme olasılığı "G" aşağıdaki gibi hesaplanır. [13]

B : Bir düğümün bozulma olasılığı

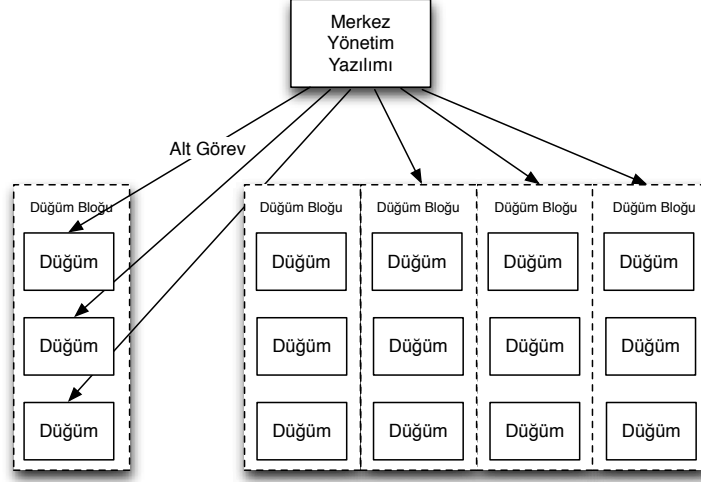
n : Bir düğümün desteklendiği yedek düğüm sayısı

N : Desteklenecek düğüm sayısı

G : Güvenilirlik

$$G = (1 - B^{n+1})^N \quad (1.1)$$

Çoklu çalıştırmaya ilişkin yapı Şekil 1.7'deki gibidir.



**Şekil 1.7:** Çoklu çalıştırma.

### 1.3.1.2 Birebir yedekleme

Bu çalışma kipinde, sistemde her düğüme karşılık bir düğüm yedekte tutulmakta, düğümde meydana gelecek olası bir sorunda mevcut düğümün görevini yedek düğüm devralmaktadır. Çoklu çalıştırmaya nazaran yedek düğüme aktarım sadece yedeklemenin olduğu zamanda yapılmakta, diğer zamanlarda düğüm ek enerji harcamamaktadır. Her ne kadar görevin sürdürülebilirliği belirgin bir şekilde iyileştirilmiş olsa da, bu çalışma kipinin en önemli zaafı, sistemde çok sayıda düğümün atıl olarak beklemesinden kaynaklanan başarımlar kaybı ve yedek düğümün de bozulması durumunda ilgili alt görevin gerçekleştirilememesinden kaynaklanan güvenilirlik açığıdır.

$0 < B < 1$  olasılığında ve her düğümün eşit bozulma olasılığında; uygulamanın toplamda doğru sonuç verme olasılığı "G" aşağıdaki gibi hesaplanır.

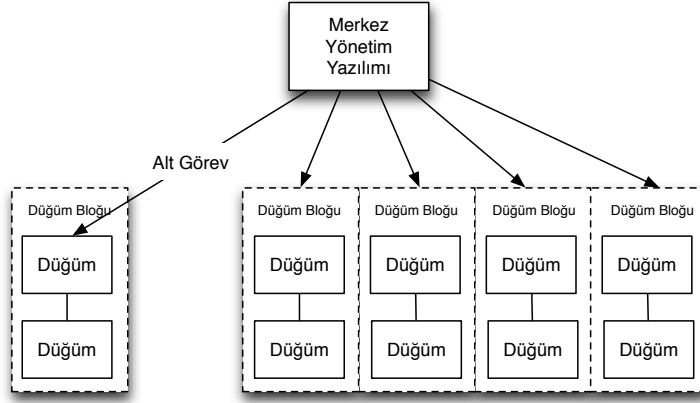
B : Bir düğümün bozulma olasılığı

N : Desteklenecek düğüm sayısı

G : Güvenilirlik

$$G = (1 - B^2)^N \quad (1.2)$$

Birebir yedeklemeye ilişkin yapı Şekil 1.8'deki gibidir.



Şekil 1.8: Birebir yedekleme.

### 1.3.1.3 Havuz yedeklemesi

Birebir yedekleme kipinin yarattığı düğüm israfını en aza indirmek ve yedek düğümün bozulması durumundaki güvenilirlik açığını gidermek için, her bir düğüme atalı bir yedek düğüm yerine, bozulan düğümler, belirli bir sayıda düğüm barındıran bir düğüm havuzundan çekilen düğümler ile yedeklenmektedir. Böylece hem boşta bekleyecek düğüm sayısı belirgin oranda azalmış olacak, hem de güvenilirlik açığı olasılığı düşürülmüş olacaktır.

$0 < B < 1$  olasılığında ve her düğümün eşit bozulma olasılığında; uygulamanın toplamda doğru sonuç verme olasılığı "G" aşağıdaki gibi hesaplanır.

B : Bir düğümün bozulma olasılığı

N : Desteklenecek düğüm sayısı

K : Yedek düğüm sayısı

G : Güvenilirlik

Yedek düğümün olmadığı durumda sistemin güvenilirliği;

$$G = (1 - B)^N \quad (1.3)$$

Tüm sistemin güvenilirliği;

$$G = G^N + \sum_{i=1}^Z \frac{N!}{i!(N-i)!} B^i G^{N-i} \sum_{j=i-1}^{K-1} \frac{K!}{(K-1-j)!(j+1)!} B^{K-1-j} G^{j+1} \quad (1.4)$$

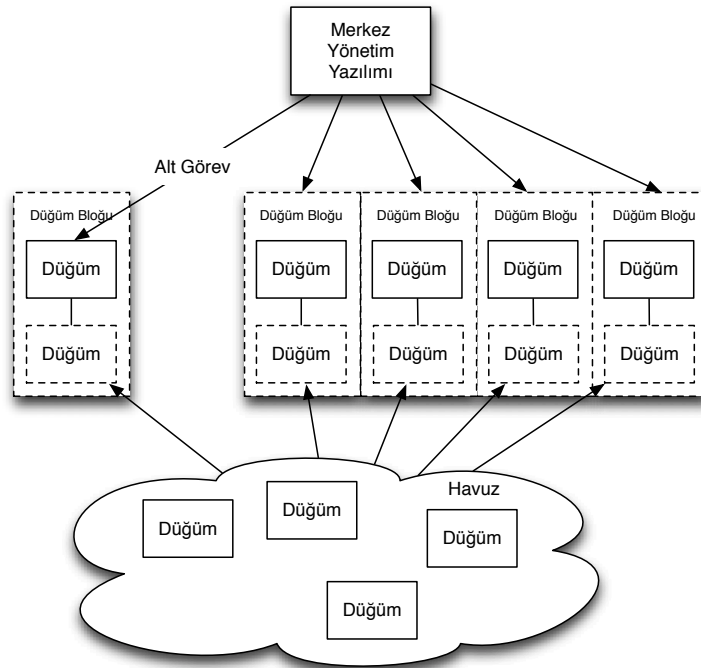
Bu bağlantıda,

$$N \leq K \implies Z = N \quad (1.5)$$

$$N > K \implies Z = K \quad (1.6)$$

şeklindedir.

Havuz yedeklemesine ilişkin yapı Şekil 1.9'deki gibidir.



**Şekil 1.9:** Havuz yedeklemesi.

#### 1.3.1.4 Yedekleme yönteminin seçimi

Havuz yedekleme yöntemiyle sadece bazı düğümlerin sürdürülebilirlik kümeleri olarak işaretlenip bekletilmesi, sistemin güvenilirliğinin sağlanması noktasında yeterli imkanı sağlamış olup, tez kapsamında sınırlı sayıda düğüm ile yeterli başarımın sağlanması açısından çoklu çalışma kipi tercih edilmemiştir.

### 1.3.2 Verim

Uygulama alt görevlerinin en yüksek başarımı sağlayacak şekilde sınırlı donanım bileşenlerine sahip düğümlerde çalıştırılması için uygulama alt görevlerinin düğümlere dağıtımında verime yönelik aşağıdaki kriterler göz önüne alınmıştır.

#### 1.3.2.1 Dosya aktarımı

Uygulama alt görevlerinin içeriğindeki ortak dosyaların düğümlere mükerrer aktarımını önlemek için MD5 [14] tabanlı bir yapı oluşturulacak, merkezi yapı ile düğüm arası dosya aktarımlarında, dosya içeriğinden önce karşılıklı olarak dosyalara ilişkin MD5 bilgileri aktarılıp, gereksiz dosya aktarımının önüne geçilecektir. Böylece ağda kaybedilen zaman en azda kalmış olacaktır. Aktarımdaki hızlanmayı ifade etmek istersek;

N : Aktarılacak dosya sayısı

M : MD5 katarının uzunluğu, sabit 16 Byte

K : Ortak dosya sayısı

U : Ortalama dosya uzunluğu

S : Hızlanma

$0 < S < 1$  aralığında;

$$S = 1 - \frac{KM + (N - K)U}{NU} \quad (1.7)$$

Genellikle ortalama dosya uzunluğu  $U \gg M$  kabul edilmiştir. Bu durumda M'i gözardı edersek, hızlanma

$$S = \frac{K}{N} \quad (1.8)$$

olacaktır.

### 1.3.2.2 Düğüm istatistiği

Merkez yönetim yazılımı düğümlere ilişkin kullanım istatistiği tutarak, düğümlerin homojen olarak kullanımını sağlayacak, böylece tüm düğümlerin aynı oranda yaşanması sağlanacaktır. Diğer yandan düğümlerin uzun süreli kullanımında oluşabilecek ısı vb. tipteki olumsuz koşulların düğümlere etkisi eşit olacağından, düğümlerin bozulma oranlarının eşit dağılacağı sonucuna varılmıştır. Şöyle ki, normal koşullar altında;

$T_i$  : Sistemin ilk bozulma zamanı

$T_b$  : Sistemde en uzun süreli çalışan düğümün çalışma zamanı

$$T_i \approx T_b \quad (1.9)$$

yakınsamasını yapabiliriz.

### 1.3.2.3 Önceliklendirme

Sistemde dolaşan mesajlar bir öncelik değeri ile ilklendirilip, dolaşımda oldukları süre boyunca öncelikleri arttırılacak, böylece geliştirilen yazılımdaki bileşenler hem mesaj hem de düğüm bazlı önceliklendirilmiş olacaktır. Bu sayede alt görevlerin öncelik sıralarına göre işlenmesi ya da melez atanmış bilgisayarlı kümelerde, önceliğe göre daha hızlı atanmış bilgisayara görevin atanması gibi seçenekler mümkün olabilecektir. Sistemin herhangi bir anında mesajların sahip olduğu öncelik değerini ifade etmek istersek;

$A\ddot{O}$  : Anlık öncelik değeri

$\ddot{I}\ddot{O}$  : İlk öncelik değeri

YK : Yineleme katsayısı

YS : Yineleme sayacı

$$A\ddot{O} = \ddot{I}\ddot{O} \times YK - YS \quad (1.10)$$

sonucuna ulaşmış oluruz.

#### 1.3.2.4 Yük dengeleyiciler

Uygulama alt görevlerinin çok sayıda olduğu durumlarda, merkez yönetim yazılımının düğümleri teker teker yönetmek yerine kümeler halinde yönetmesi ve bu kümelerdeki düğümlerin, yük dengeleyici olarak işaretlenen düğümler tarafından yönetilmesi, sistemin verimine olumlu etkisi olacaktır. Ancak tez kapsamında tercih edilen uygulamanın çapının çok büyük olmamasından dolayı, bu şekilde sağlanacak verim gözardı edilecektir.

#### 1.4 Tezin Kapsamı

Çalışma, tezin amacı doğrultusunda, dört ana bölümden oluşmaktadır;

- İlk bölümde, tezin amacına uygun olarak literatür araştırması yapılmış ve paralelleştirilen bir uygulamanın sınırlı kaynaklara sahip atanmış bilgisayarlar kullanılması noktasında dikkat edilmesi gereken noktalar incelenmiştir.

Yapılacak mimari seçim ile, sahip olunan toplam işlemci gücünün en verimli ve en güvenli şekilde nasıl kullanılması gerektiği araştırılmış, mevcut mimarilere katkı anlamında kullanılabilecek yöntemler incelenmiştir.

- İkinci bölümde dağıtık mimariler ve bunlara ilişkin haberleşme altyapıları ile yönetim mekanizmaları incelenmiştir. İncelemeler sonucunda tezin amacına en uygun mimarinin bilgisayar kümeleri mimarisi olduğu ortaya çıkarılmış, bu amaçla kullanılabilecek haberleşme altyapıları ile geliştirilecek yönetim yazılımına benzer yazılımlar incelenmiştir.

Literatür araştırmalarında ortaya çıkan sonuçlara göre atanmış sistemler için en uygun haberleşme altyapısının soket bazlı yapılar olduğu sonucuna varılmış, geliştirilecek yönetim yazılımına entegrasyonu kararı verilmiştir.

Diğer taraftan paralelleştirmenin getirileri ve sınırları noktasında Amdahl [15] ve Gustafson'un [16] çalışmaları incelenmiştir.

Sonrasında referans bir ölçüm uygulaması olarak matris çarpımının hesaplanması uygulaması tespit edilmiştir.

Son olarak tasarlanacak uygulamanın alt görevlerinin üzerinde çalışacağı atanmış bilgisayar donanımları araştırılmış, tercih edilen dört tanesinin işlemci, fiyat vb. gibi özellikleri listelenmiştir.

- Üçüncü bölümde geliştirilen merkez yönetim yazılımının geliştirilme süreci anlatılmış, sırasıyla ana iş akışı, ana bileşenler, haberleşme altyapısı, kuralların tanımlanması, kullanıcı arayüzü ve matris uygulamasının tasarımı konuları işlenmiştir.

Ana bileşenlerin işleyişi konusunda; farklı bilgisayarlarda çalışabilmesi için birbirlerinden bağımsız olması kararlaştırılmış, bu nedenle haberleşme arabirimlerinin de bağımsız olarak seçilebilecek bir yapıda olması sağlanmıştır. İş akışına uygun olarak tasarlanan mesaj tipleri ve bunların bileşenler arası aktarım ve işlenmesi konuları anlatılmıştır.

Haberleşme altyapısında, haberleşme soket bazlı olduğu için mesajların detaylı olarak serileştirilme ve ayrıştırılma işlemleri ile önceliklendirilme ve önceliğe göre işlenme adımları anlatılmıştır.

Uygulama alt görevlerinin yönetim yazılımı içerisinde kural yapısı kullanılarak tanımlanması ve bu yapı dahilinde düğümlerde çalıştırılma şekli anlatılmıştır.

Kullanıcı arayüzü olarak yapılan iki farklı tasarım ile bu arayüzler kullanılarak tetiklenebilecek eylemler anlatılmıştır.

Son olarak matris uygulamasının içeriği anlatılmıştır.

- Son bölümde geliştirilen yönetim yazılımının, temin edilen donanımlarla ölçülmesi konusu incelenmiştir.

İlk olarak kullanılacak platform ve donanımlar tanıtılmış, sonrasında düğümlerde çalıştırılacak uygulamaya ilişkin detaylı bilgi verilmiştir.

Ölçümlerde kullanılacak veri kümeleri listelenmiş, ölçüm şekilleri anlatılmıştır. Ölçümler 10 tekrarlı olarak yapılmış, bazı donanımlar için tek düğümlü, bazıları için ise tek, iki ve üç düğümlü olarak yapılmış, başarımlar sonuçları karşılaştırılmıştır.

Sonrasında masaüstü sistemi referans alınarak bir maliyet analizi yapılmış, sistemlerin güvenli çalıştırılma yüzdelere göre gereken donanımlar ve bunların toplam maliyetleri masaüstü sistem ile karşılaştırılmıştır.

Son olarak ölçüm sonuçları irdelenmiş, donanımların masaüstü sistemine oranla maliyet analizleri listelenip, çalışmanın başarılı ve başarısız olduğu noktalar değerlendirilmiştir.

## **2. DAĞITIK MİMARİ VE YÖNETİM YAZILIMLARI**

Uygulamaların, paralel çalıştırılma kavramından yola çıkarak, dağıtık mimariler vasıtasıyla farklı donanımlarda çalıştırılması ve bu sayede uygulamanın büyüklüğünün ya da süre kısıtlarının, bir sorun olmaktan çıkması, yapılan çalışmanın dağıtık mimariler çerçevesinde yapılması gerektiğini ortaya koymuştur. Tezin amacı doğrultusunda belirlenen sorunun çözümüne yönelik dağıtık mimari çeşitleri araştırılmış, seçilen yönteme ilişkin inceleme ayrıca ele alınmıştır.

### **2.1 Dağıtık Mimari Çeşitleri**

Uygulamaların ayırık istemcilerde ayrıştırılarak çalıştırılması düşüncesi beraberinde iki nokta arasındaki haberleşmenin gerekliliği ve bu haberleşmenin nasıl olması gerektiği konusunu gündeme getirmektedir. Tasarlanacak olan uygulamanın temelde hangi haberleşme teknolojisi üzerine inşa edilmesi gerektiği de önemli bir noktadır. Bu nedenle dağıtık sistemlerin günümüze kadar gelişimi ve tez kapsamında hangi noktadan itibaren referans alınıp ilerlenmesi gerektiği araştırılmıştır.

#### **2.1.1 Ana bilgisayarlar**

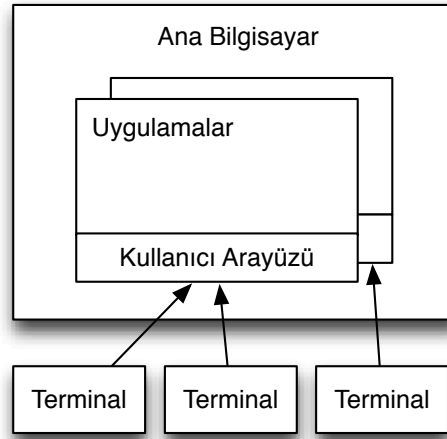
Çok güçlü bilgisayarlar olarak da tanımlayabileceğimiz ana bilgisayarlar [17], kendisine bir ağ üzerinden bağlanan çok sayıda kullanıcıya paralel hizmet vermesiyle, haberleşme anlamında dağıtık sistemlerin ilk örneğini oluştururlar. Çok sayıda işlemci barındırmaları ve bu işlemcileri paralel kullanabilmeyi destekleyen işletim sistemleri vasıtasıyla birden fazla kullanıcının uzak terminaller üzerinden uygulamaları çalıştırmaları mümkün olmuştur.

Ana bilgisayarlarda çalışan yazılımlar, her uygulama parçasını ve buna bağlı bileşenleri kendi içerisinde barındırmalarından dolayı tek parçalı bir yapıya sahiptirler. Uzak terminaller, çalıştırdıkları uygulamayı tümüyle ana bilgisayarlar üstünde

çalıştırmakta, dolayısıyla aslında sadece kullanıcı ile ana bilgisayar arasında bir ağ köprüsü görevi üstlenmektedirler.

Ana bilgisayarlar, fiziksel boyutlarının büyük olması, içerdiği bileşenlerin çokluğu ve pahalılığı, periyodik bakımları ve kendilerine özel ortamlarda bulunması zorunlulukları da düşünüldüğünde oldukça maliyetli sistemlerdir. Herhangi bir sorun yaşanması durumunda tüm sistemin ve sisteme bağlı bütün kullanıcıların iş yapamaz durumda kalmasına neden olmaktadır.

Ana bilgisayarların üzerlerinde çalıştırdıkları işletim sistemleri çoklu kullanıcıya aynı anda hizmet verme kabiliyetine sahip özel yazılımlardır. Bu da uzaktan bağlanan kullanıcıların tek bir işletim sistemine özel olarak tasarlanmış uygulamaları çalıştırma koşuluna, dolayısıyla da uygulamaların o sistemin kısıtları ve özellik havuzu (sanal makine, derleyici desteği vb.) ile sınırlandırılmalarına yol açmıştır. Ana bilgisayarların çalışma yöntemi Şekil 2.1’da gösterilmiştir.



**Şekil 2.1:** Ana bilgisayar çalışma yöntemi.

### 2.1.2 İstemci / Sunucu mimarisi

Ana bilgisayarların da içinde bulunduğu tek parça mimarilerde bütün sistem tek cihazda çalıştırılmakta ve bunun sonucu olarak bütün risk aynı cihaza yüklenmekte ve paylaştırılmamaktadır.

Uygulamaların bir kısmının terminallerde çalıştırılabilir olması fikrinden hareketle istemci / sunucu mimarisi [18] geliştirilmiştir. Bu mimaride bütün yük tek bir bilgisayardan (sunucu) alınarak istemci ile paylaştırılmaktadır. Böylece hem sunucu

bilgisayar, ana bilgisayar sistemlerinde olduđu kadar çok maliyetli donanımlara sahip olmak zorunda kalmamakta hem de risk belirli ölçüde iki bilgisayar arasında dağıtılmaktadır.

İstemciler üzerlerinde kendi işletim sistemi ve yazılım ortamlarını barındırdıklarından kendi başlarına uygulamaları çalıştırabilmektedirler; uygulamaların ihtiyaç duyması halinde de sunucu ile iletişim kurmaktadır. Böylece uygulama bazında iş bölümü gerçekleşmekte, istemci gerekli gördüğü uygulama bileşenini kendisinde ya da sunucuda çalıştırabilmektedir.

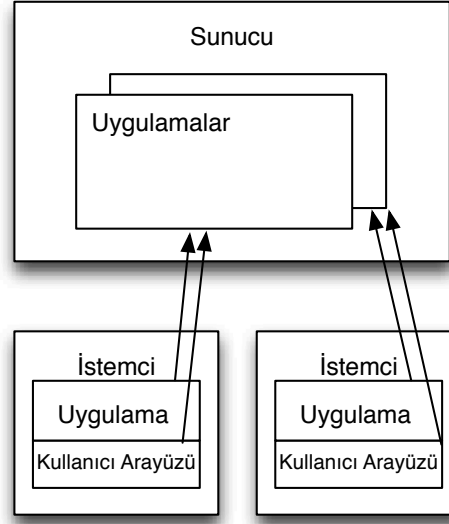
Günümüz istemci / sunucu mimarilerinde temel olarak seçilen yöntem; veri tabanı gibi her kullanıcının ortak kullanımında olan ve çok sayıda giriş / çıkış işlemini belirli bir hızın üstünde gerçekleştirmesi gereken bileşenler sunucuda, kullanıcı arayüzü ile ilişkili bileşenler istemcilerde, uygulamanın akışını yönlendiren kural takımları ihtiyaca göre sunucu ya da istemcide yer almaktadır.

İstemci / sunucu mimarisinde dikkat edilmesi gereken nokta sunucuya birden fazla istemcinin aynı anda erişmesi durumunda veri tutarlılığının sağlanmasıdır. Diğer bir deyişle sunucu, istemcilerden gelen istekleri belirli bir senkronizasyon mekanizması dahilinde işlemekte, istemcilere her zaman en güncel veriyi sunabilmektedir.

İstemci / sunucu mimarisinde karşılaşılan en önemli sorun, özellikle sunucuda yapılan güncellemelerin bütün istemcilerde bir güncelleme gerektirdiği durumdur. Örnek olarak veri tabanı erişim yönteminde yapılacak ufak bir değişiklik, bütün istemcilerde bir güncelleme gerektirmektedir. Bu tarz sorunları aşmak için seçilen yöntem, uygulamaların kendi içinde katmanlı bir yapıda tasarlanmasıdır. Her katmanın haberleşebildiği diğer katmanlar tanımlanmakta dolayısıyla katmanlardan herhangi birinde yapılan bir değişiklik uygulamanın bütünü yerine sadece ilgili katmanlarda güncelleme yapmayı gerektirmektedir. İstemci / Sunucu mimarisinin çalışma yöntemi Şekil 2.2’da gösterilmiştir.

### **2.1.3 Dağıtık nesne mimarisi**

Dağıtık nesne mimarisinde [19]; uygulamaların sahip olduđu alt bileşenler, istemci / sunucu arasında paylaştırılacak şekilde tasarlanmak yerine tüm işleyiş nesne tabanlı olarak birden fazla bilgisayar üzerinde bir arada çalışacak şekilde tasarlanır. Nesneler



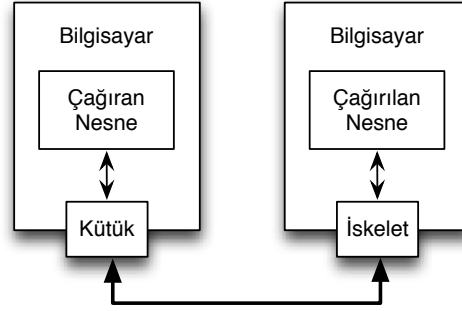
**Şekil 2.2:** İstemci / Sunucu mimarisi.

arası haberleşme bir ağ katmanı üzerinde saydam olarak mesajlaşma vasıtasıyla yapılır. Dağıtık nesne mimarisini istemci / sunucu mimarisinden ayıran en önemli özellik; sistem, kullanıcıya tek bir ana bilgisayarmış gibi saydam gözükür. Yani kullanıcı çalıştırdığı uygulamanın ağ üzerinde farklı bilgisayarları kullandığının farkında olmaz.

Dağıtık nesne mimarisinin diğer bir önemli üstünlüğü ise; mimari, sağladığı işlevselliği üzerinde çalıştığı işletim sistemi, donanım, programlama dili gibi bileşenlerden yalıtıldığı için sistemdeki nesneler birbirleriyle tanımlı olan arabirimler üzerinden karşı sistemin fiziksel bileşenlerine bağlı olmaksızın mesajlaşabilmektedirler. Bu da dağıtık mimarinin çok geniş ölçekte uygulama sahasına yayılmasına imkan vermektedir. Sistemin güncelleme gereksinimleri, arabirim değişiklikleri olmadığı müddetçe sistemin diğer kısımlarını etkilememekte, böylece sistemin sürdürülebilirlik maliyetini en aza indirmektedir.

Dağıtık nesne mimarisi için geliştirilen uygulamalar, nesne tabanlı tasarlandığı için, işlevselliklerini nesnelere entegre edilen servisler üzerinden tanımlı arabirimler vasıtasıyla sağlamaktadırlar. Temel işleyiş servisler üzerinden olduğu için, istemci / sunucu gibi bir ayırım servis üzerinden değerlendirilmekte, bir istemci bilgisayar, sunucu işlevselliğine sahip bir servise sahip olmasıyla sunucu gibi çalışabilmektedir. Dağıtık nesne mimarisinin çalışma yöntemi Şekil 2.3’da gösterilmiştir.

Dağıtık nesne mimarisini oluşturan nesnelerin birbirleriyle olan haberleşmelerinde kullanılan yapılardan bazıları şunlardır;



**Şekil 2.3:** Dağıtık nesne mimarisi.

#### 2.1.3.1 Uzak yordam çağrıları

Uzak yordam çağrısı (*Remote Procedure Call (RPC)*), belirli bir işlevin bir program tarafından saydam olarak başka bir adres uzayı ya da bilgisayarda çalıştırılmasını sağlayan süreçler arası iletişim yöntemidir. Bu yöntemde programcı, eriştiği bilgisayara erişim yöntemiyle ilişkili ek bir efor sarfetmesine gerek kalmaz.

Java programlama dilinde, nesne tabanlı olarak "Uzak Metot Çağırımı" (*Remote Method Invocation (RMI)*) adıyla yeniden yapılandırılmıştır.

#### 2.1.3.2 CORBA

*Common Object Request Broker Architecture (CORBA)* [20], platform ve programlama dilinden bağımsız olarak dağıtık nesnelerin birarada çalışmasını ve bilgi alışverişini sağlayan ara katman çözümüdür. Nesnelere haberleşmeleri için bir arabirim sağladığından dolayı aslında bir yazılım yolu olarak da adlandırılabilir.

#### 2.1.3.3 DCOM

*Distributed Component Object Model (DCOM)* [21], Microsoft firmasının çıkardığı dağıtık yazılım bileşenleri arasında haberleşmeyi sağlayan ve sadece Microsoft firmasının ürünleriyle sınırlı altyapıdır. Temelde CORBA'nın alternatifi olsa da çok arabirimli nesneleri desteklemesi ve RPC tabanlı olması gibi farklarla CORBA'dan ayrılır.

## 2.1.4 Bilgisayar kümeleri mimarisi

Bilgisayar kümeleri (*Computer Clusters*) mimarisi [22], birbirlerine gevşek ya da sıkı olarak bağlı bilgisayarların bir arada çalışarak bir görevi yerine getirdikleri mimaridir. Merkezi bir yönetim yazılımı vasıtasıyla her bir bilgisayara aynı görev atanıp gerçekleştirilmesi sağlanmaktadır.

Mimaride, zorunlu olmamakla beraber bilgisayarların birbirleriyle hem yazılımsal hem de donanımsal olarak aynı yapılandırmada olması tercih edilmekte, böylece gerçekleştirilecek görevin türdeş kalması sağlanmakta, bilgisayar bazlı farklı yapılandırma gereksinimi ortadan kalkmaktadır.

Mimarinin temel amacı, uygulamaların alt görevlerinin birden fazla bilgisayara dağıtılarak bir süper bilgisayarda çalıştırılıyormuş gibi başarımlı elde edilmesini sağlamaktır. Burada dikkat edilmesi gereken ana husus, uygulamaların alt görevlerinin birbirlerine bağımlılıklarının olmaması ve birbirleriyle birlikte çalışabiliyor olmasıdır. O yüzden bilgisayar kümeleri, standart bilgisayar uygulamalarının işletilmesi için uygun ortamı sunmazlar.

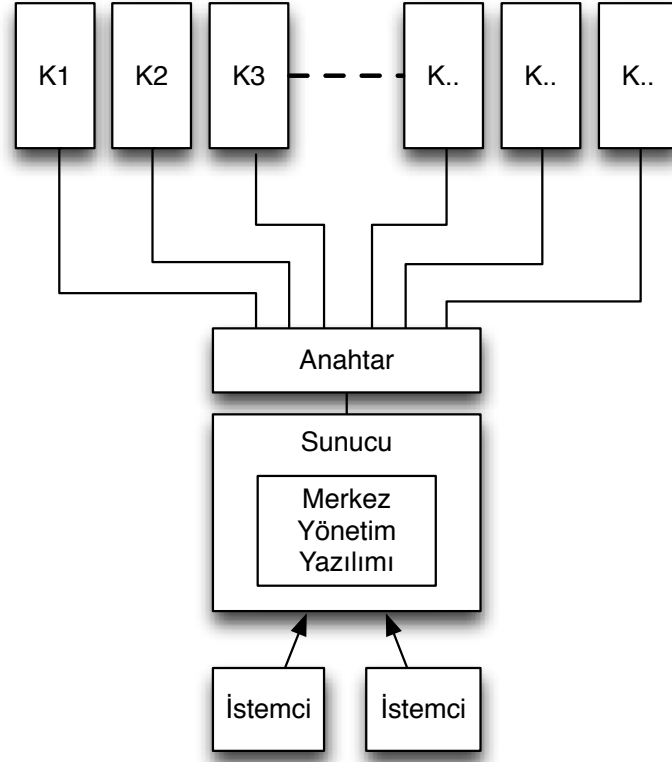
Bilgisayarların birbirlerine olan bağlantısı genellikle yüksek hızlı yerel ağlar kullanılarak sağlanmakta, böylece verinin bilgisayarlara aktarılması işleminin başarımlı kriterlerine etkisinin en azda kalması sağlanmaktadır.

Mimarideki genel yaklaşım, diğer dağıtık mimarilerden farklı olarak, bilgisayar kümelerinin üzerinde çalışan bir merkezi yönetim yapısına sahip olması ve bu yapı vasıtasıyla bilgisayar kümelerinin dışarıyla yalıtımının sağlanmasıdır. Böylece sistem dışarıda tek bir süper bilgisayarmış gibi gözükmektedir. Bilgisayar kümeleri mimarisinin çalışma yöntemi Şekil 2.4'da gösterilmiştir.

Sistemin başarımlı ve güvenilirlik yapısı yine merkez yönetim yazılımı sayesinde kümelerden bir ya da birkaçına farklı görevler yükleyerek iyileştirilebilmektedir. Bunlardan birkaçı aşağıda sıralanmıştır;

### 2.1.4.1 Yüksek devamlılık kümeleri

Sistemin sürekliliğini sağlamak üzere bazı kümeler yedekte boşta tutulmakta, aktif kümelerde meydana gelen sorunlarda bu kümeler yapılan görevi devralmaktadırlar.



**Şekil 2.4:** Bilgisayar kümeleri mimarisi.

Sistemdeki küme sayısı ile doğru orantıda, belli sayıda küme, sürdürülebilirliği sağlamak adına rezerve edilmektedir.

#### **2.1.4.2 Yük dengeleyici kümeler**

Merkez yönetim yazılımı gelen isteklere hızlı cevap veremediği, özellikle çok sayıda alt görevin olduğu durumlarda, bazı kümeler "yük dengeleyici kümeler" olarak tanımlanıp, gelen isteklerin dengeli bir şekilde diğer kümelere dağıtımından sorumlu olurlar.

#### **2.1.4.3 Yüksek başarılı kümeler**

Özellikle büyük ölçekli veri uygulamalarında kendisine atanan görevin bölünüp paralelleştirilmesi ve başarımın artırılmasına yönelik tanımlanmış kümelere.

#### **2.1.4.4 Beowulf kümesi**

Beowulf [23], paralel hesaplamalarda kullanılmak üzere bilgisayar kümeleri mimarisini referans alan çok bilgisayarlı bir mimari örneğidir. Genellikle bir sunucu bilgisayarı ile çok sayıda istemci bilgisayarının birbirine lokal bir ağda ethernet ya da başka bir ağ arabirimi vasıtasıyla bağlandığı bir sistemden oluşur. Mimarinin ortaya çıkış amacı, sistemin kolay ulaşılabilen ve maliyeti düşük donanım / yazılım bileşenleriyle oluşturulmasına imkan vermesidir. Kümeyi oluşturan bilgisayarlar için herhangi bir özel donanım bileşeni gerekmemekte, standart linux, freebsd gibi ücretsiz işletim sistemleri kullanılabilmektedir.

Haberleşme için genellikle Mesaj Aktarım modelini kullanan haberleşme teknolojileri kullanılmaktadır.

Beowulf kümesini diğer küme mimarilerinden ayıran bir özellik de istemci rolündeki bilgisayarların oldukça yalın olmaları ve sisteme olan katkılarının dışarıdan sadece ek işlemci ve bellek desteği olarak gözükmeleridir.

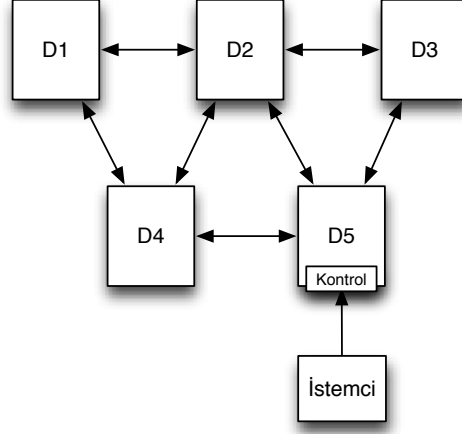
#### **2.1.5 Grid mimarisi**

Grid mimarisi [24] genel olarak bilgisayar kümeleri mimarisinde olduğu gibi çok sayıda gevşek bağlı bilgisayarın birarada çalışarak bir görev yerine getirdikleri bir mimaridir. Fakat bilgisayar kümeleri mimarisinden farklı olarak, bilgisayarlar çok farklı lokasyonlarda ve farklı yazılım / donanım bileşenlerine sahip olabilmektedirler. Çok sayıda bilgisayar bu sayede grid mimarisine hızlı bir şekilde entegre edilebilir.

Grid mimarisine bir anlamda kümeleri coğrafi olarak değişik yerlere yayılmış süper bir sanal bilgisayar olarak bakabiliriz. Her bir bilgisayarın bağımsız yapılandırılma olması, üzerlerinde çalışacak görevlerin de diğer bilgisayarlardan olabildiğince bağımsız çalışmasını gerektirmektedir. Diğer yandan özellikle başarımların ölçümlerinin yapıldığı uygulamalarda grid mimarisi, bilgisayar kaynaklarının içeriklerinin kontrol edilememesi yüzünden sağlıklı sonuçlar vermemektedir.

Bilgisayarların coğrafi olarak birbirinden uzakta olabiliyor olması bir görevin tamamlanmasındaki başarımların kriterleri içerisinde görevle ilişkili verilerin aktarım zamanının da alınmasını zorunlu kılmaktadır.

Mimarinin günümüzde popülerleşmesinin en önemli nedeni; insanların gönüllülük esasıyla kendi bilgisayarlarının, sosyal sorumluluk ya da akademik amaçlı projelerde kullanılmasını sağlayan bir yapı sunmasıdır. Grid mimarisinin çalışma yöntemi Şekil 2.5’da gösterilmiştir.



**Şekil 2.5:** Grid mimarisi.

## 2.2 Dağıtık Mimarilerin Değerlendirilmesi

Yapılan araştırmadan yola çıkarak, tasarlanacak mimarinin dağıtık tabanlı olması ve dışarıya karşı kendini yalıtması, fakat aynı zamanda sisteme fazla yük getirmeyen bir yapıda olması gerektiği ortaya çıkmıştır. İncelenen mimarilerde;

- Ana bilgisayar mimarisinin, sistemin maliyeti düşünüldüğünde amaca uygun olmadığı sonucuna varılmıştır.
- İstemci / Sunucu mimarisinin donanım ve yazılım gereksinimleri yüksek olmamakla birlikte bu mimarinin dışarıya karşı bir yalıtım sağlamadığı ve her istemcinin sunucuyla olan bağlantısının ayrı ayrı tanımlanması gerektiği göz önünde tutulduğunda, mimarinin amaca uygun olmadığı sonucuna varılmıştır.
- Dağıtık nesne mimarisi, dışarıya karşı bir yalıtım ve platform (işletim sistemi, programlama dili vb.) esnekliği sağlasa da, belli bir ara katmanın üzerinde çalışma zorunluluğu getirmektedir. İlgili ara katmandan kaynaklanan ek işlem yükünün, özellikle atanmış bilgisayarların başarımları / fiyat oranı en uygun olacak şekilde seçildikleri düşünüldüğünde, bu sistemde gerçekleştirilecek göreve atanacak

işlemci gücünün ciddi bir kısmını kullanacağı sonucuna varılmıştır. Ara katmanın getirdiği yük dolayısıyla atanmış bilgisayarların sağlayacağı işlemci gücü yetersiz kalacağı için, mimarinin amaca uygun olmadığı sonucuna varılmıştır.

- Grid mimarisi, özellikle donanım kontrolünün kolay olmaması, güvenilirlik zaafı ve ağ kaynaklı başarımların kayıplarının çokça olacağı öngörülerek, mimarinin amaca uygun olmadığı sonucuna varılmıştır.

Yapılan araştırma sonucunda amaca en uygun çözümün bilgisayar kümeleri mimarisi olduğu sonucuna varılmıştır, şöyle ki;

- İş gerçekleştirecek atanmış bilgisayarlardan uygulamanın dağıtım yükünü almakta,
- Dışarıya bir yalıtım sağlamakta,
- Atanmış bilgisayarlara ek bir yazılım yükü getirmemekte,
- Gerçekleştirilecek göreve ilişkin aktarım yükünün en azda kalmasını sağlamakta,
- Sistemin sürdürülebilirliğini en az maliyetle yapabilmektedir.

Sayılan üstünlüklerine karşın sistemin en önemli eksikliği platform esnekliği sağlamamasıdır. Çalıştırılacak uygulamanın atanmış bilgisayarın sahip olduğu işlemci mimarisi ve işletim sistemi ile uyumlu olması gerekmektedir. Ancak bilgisayar sistemlerinde genel olarak platform bağımsızlığı ile başarımlar arasında karşılıklı dışlama durumu hakimdir, yani platform bağımsızlığı beraberinde belirgin bir başarımlar kaybını da getirmektedir. Tezin amacındaki öncelik başarımlar artırımı olduğundan, platform bağımsızlığı bu nedenle değerlendirme dışında kalmıştır.

Merkez yönetim yazılımı, bilgisayar kümeleri mimarisinin ana bileşenlerinden olup, paralelleştirilen alt görevlerin, bağlı bilgisayarlara aktarımı, çalıştırılması, sürdürülebilirliği ve güvenilirliğinin sağlanması gibi görevleri gerçekleştirir. Böylece bilgisayarlar atanan göreve ilişkin yürütmeyi başarımlar en optimumda olacak şekilde gerçekleştirirler. Yazılımın çalışacağı ana bilgisayar hem kullanıcı arayüzünün gereksinimleri hem de çok sayıda istemcinin yönlendirdiği iş isteklerini hızlı cevaplayabilmesi için sunucu özellikli bir bilgisayar olacaktır. Sistemde görevleri yürütecek atanmış bilgisayarlar, tasarımda **düğüm** olarak adlandırılmışlardır.

Tez kapsamında seçilen donanımların düşük maliyetli oluşu, birbirleri arasındaki bağlantıların ethernet arabirimi vasıtasıyla sağlanması ve unix tabanlı işletim sistemlerini çalıştırabilmesi, tasarımın; bilgisayar kümeleri mimarisinin özelleşmiş bir örneği olan **Beowulf Kümesi** mimarisine yakın olduğunu göstermektedir.

## 2.3 Haberleşme Altyapısı

Bilgisayar kümeleri mimarisinde genellikle düğümler arasındaki mesafenin kısa olması ve çoğunlukla sistem kurulumu için lokal ağların tercih edilmesi, kullanılacak haberleşme teknolojisi konusunda başarıya odaklanabilmeyi mümkün kılmıştır. Mesaj Aktarımı ve Uzak Yordam Çağrılarını, işlemler arası haberleşme teknolojilerinde en yoğun olarak kullanılan iki yöntemdir. Dağıtık Paylaşımlı Bellek, daha az tercih edilmesine karşın sağladığı soyutlama vasıtasıyla geliştiriciye standart paylaşımlı bellek arabirimi sağlamaktadır.

Başarım kriterini de göz önüne alarak bahsedilen haberleşme modelleri incelenmiştir. [25]

### 2.3.1 Mesaj aktarımı

Mesaj aktarımı, işlemler ya da bilgisayar düğümleri arası mesaj gönderme işini en basit gönder / al ilkeleri vasıtasıyla sağlar. Aktarılan mesaja ait verinin alıcı ve vericiden soyutlama yeteneği düşük seviyededir, diğer bir deyişle, gönderilen veriye ilişkin seçilen protokol, verinin serileştirilmesi ve geri toplanması, senkronizasyon gibi konular geliştirici tarafından ayrıca ele alınması gereken konulardır.

Mesaj Aktarımının sisteme sağladığı en önemli üstünlük, sisteme olan maliyetinin ve gerektirdiği işlemci gücünün az olması ek bir altyapı gerektirmemesi ve hemen hemen tüm işletim sistemi platformlarında bulunmasıdır.

Mesaj aktarımı aşağıdaki özellikler çerçevesinde haberleşmede istenen esneklik ve çeşitliliği sağlamaktadır. [26] [27]

- **Engelli / Engelsiz Çalışma** : Engelli çalışma kipinde, gönder ilkelini çağıran işlem, mesajın gönderilmesi işlemi gerçekleşene kadar engellenir, aynı şekilde Al ilkelini çağıran işlem mesaj alımı gerçekleşene kadar engellenir. Engelsiz çalışma kipinde

ise, çağırılan ilkeller işlemi engellemez, işletim sisteminin kesme desteği vasıtasıyla mesajın gönderilip ya da alındığından haberdar olunur.

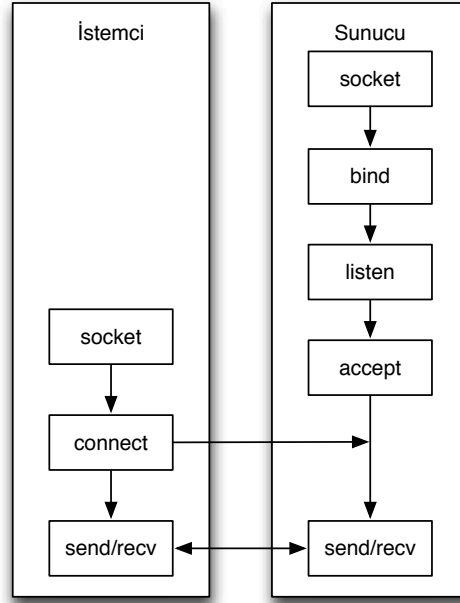
- **Tamponlu / Tamponsuz Mesajlar** : Tamponlu çalışma kipinde, gönderilen ya da alınan mesajların mesajın işlendiği ana kadar bir tampon alanda tutulması, böylece gönder / al ilkellerinin senkron kaymasında ortaya çıkabilecek olası veri kayıplarının önüne geçilmiş olunur.
- **Güvenli / Güvensiz Gönderim** : Güvenli gönderim çalışma kipinde gönderici her gönderme işleminden sonra alıcıdan bir alındı mesajı bekler. Uygulamadaki beklentiye göre de ya mesaj gönderimi tekrarlanır ya da hata mesajı ile uygulama tetiklenir.
- **Doğrudan / Dolaylı Gönderim** : Doğrudan gönderim kipinde, gönderilen mesaj doğrudan alıcı işlemin kendisine, dolaylı gönderimde ise alıcı tarafından tanımlı bir porta gönderilir.

### 2.3.1.1 Berkeley soketleri

Berkeley Soketleri (*Berkeley Sockets* (BSD Sockets)) [28], işlemler arası haberleşmede (IPC) kullanılan ve çift yönlü haberleşme imkanı sunan, içinde Internet (TCP/IP) ve Unix Alan Soketleri olmak üzere iki farklı türünün bulunduğu, uygulama programlama arabirimi ve buna bağlı kütüphaneden oluşur. Özellikle istemci / sunucu mimarilerinin gelişimiyle, istemci ve sunucu arasındaki haberleşmede ana altyapıyı oluşturmuş, diğer dağıtık mimarilerde de kendisine kullanım sahası bulmuştur. Berkeley soketlerinin çalışma akışı Şekil 2.6'de gösterilmiştir.

Berkeley Soketleri iki düğüm arasında haberleşmeyi sahip olduğu aşağıda sıralanan işlevler ile sağlamaktadır.

- **socket** : Bir sayıyla ifade edilen ve yapıyı ilk kullanım konumuna getiren çağrıdır.
- **bind** : Oluşturulan soketin bir adres yapısıyla eşleştirildiği çağrıdır. Adres, Internet Soketlerinde IP adresi ve portuna, Unix Alan Soketlerinde ise sistemdeki bir dosya yoluna karşılık gelmektedir.
- **listen** : Soketi, karşıdan gelecek bağlantı isteklerini dinleme kipine getiren çağrıdır.



Şekil 2.6: Berkeley soketleri.

- **accept** : Karşıdan gelen bağlantı isteğini kabul edip, yeni bir soket vasıtasıyla bağlantı yolu oluşturan çağrıdır.
- **connect** : Karşıya bağlantı isteği gönderen çağrıdır.
- **send** : Karşıya veriyi gönderen çağrıdır.
- **recv** : Karşıdan veri alan çağrıdır.
- **close** : Soket bağlantısını kapatmaya yarayan ve kullandığı sistem kaynaklarını sisteme iade eden çağrıdır.

Berkeley Soketleri engelli ve engelsiz olarak her iki modda da çalışmakta, kurulan bağlantı tipine göre *Stream* soket tipi ile güvenli, *Datagram* soket tipi ile güvensiz haberleşme imkanı sağlamaktadır.

### 2.3.1.2 Mesaj aktarım arabirimi

Mesaj Aktarım Arabirimi (*Message Passing Interface* (MPI)), mesaj aktarımının farklı platformlarda taşınabilirliğinin sağlanması açısından hem sözdizimi hem de semantik açısından bir standartlaşmaya gidilmesiyle oluşan bir mesaj aktarım şemasıdır. [29] MPI'nın tanımlanmasına kadar olan süreçte, PICL [30], PVM [31] gibi çok sayıda

şema mevcuttu ve geniş alanda bir kullanım sahaları yoktu. Bu nedenle bir şemayı kullanıp geliştirilen uygulamalar, diğer bir şema kullanılarak geliştirilen uygulamalarla haberleşemiyordu. Benzer şekilde günümüzde yine çokça kullanılan Berkeley ve Windows Soket Mimarileri sınırlı işlevselliğe sahip taşınabilir mesaj aktarım arabirimi sundular.

MPI tanımının sağladığı en önemli üstünlük, geliştiricilere verim, taşınabilirlik ve işlevsellik gibi özellikleri taviz vermeden sağlamasıdır. Özellikle geliştiricilere taşınabilir uygulamalarını çeşitli üreticiler tarafından farklı donanım ve yazılım bileşenlerine sahip bilgisayarlarda çalıştırabilme imkanını ve işlem gruplarının dinamik yönetimi gibi kullanışlı ek özellikleri paralel uygulamaların kullanımına sunmaktadır.

MPI özünde, paralel uygulama geliştirmeyi kolaylaştırıcı işlevlere ve yapılara sahip, standartlaşmış tanımlama ve kurallar bütünüdür.

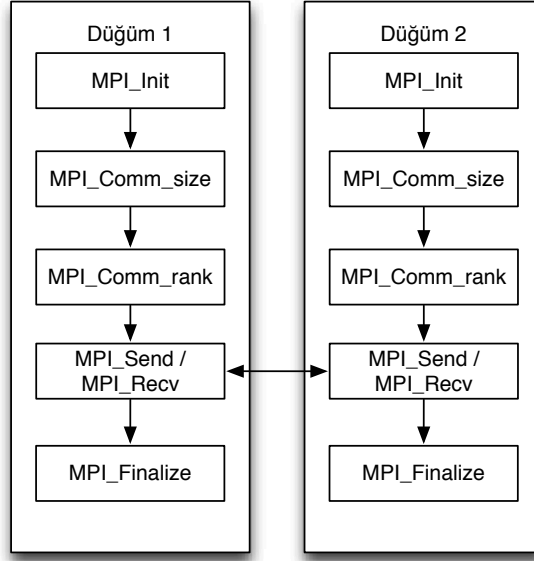
Genel olarak soket mimarisinin iyi bir alternatifi olarak ortaya çıkan MPI, aşağıdaki özelliklerle soket mimarisinden ayrılmaktadır;

- Soket mimarisinde IP-Port veya dosya adı ile temsil edilen kimlik bilgisi MPI’da **rank** adı verilen sayılarla temsil edilmektedir.
- Soket mimarisi sadece noktadan noktaya bağlantıyı desteklerken, MPI noktadan noktaya bağlantının yanısıra tek taraflı bağlantı ve çoklu bağlantı (kollektif) ’yı da desteklemektedir.
- Soket mimarisinin aktardığı veri blok halinde ardışıl sıralanmış ham veri iken, MPI tanımlı mesajlar olarak aktarmaktadır. O yüzden mesajın serileştirilmesi işlemi soket mimarisinde geliştiricinin sorumluluğundadır.
- MPI, soket mimarisine göre daha hızlı bir mesajlaşma imkanı sunmaktadır.

Mesaj aktarım arabiriminin çalışma akışı Şekil 2.7’de gösterilmiştir.

MPI, iki düğüm arasında haberleşmeyi aşağıda sıralanan işlevler ile sağlamaktadır.

- **MPI\_Init** : MPI kütüphanesinin ayağa kaldırılmasını sağlayan çağrıdır.
- **MPI\_Comm\_size** : Kaç düğümün ya da işlemin olduğunu döndüren çağrıdır.



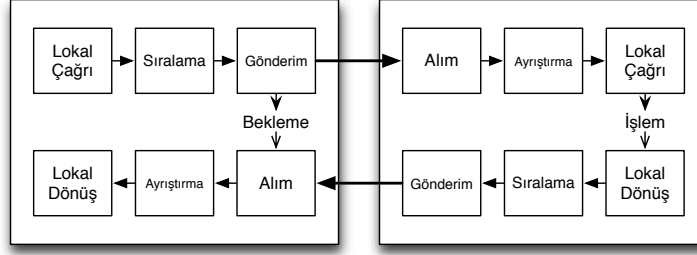
**Şekil 2.7:** Mesaj aktarım arabirimi.

- **MPI\_Comm\_rank** : Düğümün kimlik bilgisini döndüren çağrıdır.
- **MPI\_Send** : Karşıya veriyi gönderen çağrıdır.
- **MPI\_Recv** : Karşıdan veri alan çağrıdır.
- **MPI\_Finalize** : Kütüphanenin kullandığı sistem kaynaklarını sisteme iade eden çağrıdır.

### 2.3.2 Uzak yordam çağrısı

Uzak Yordam Çağrısı, geliştiriciyi verinin karşıya aktarımındaki yönetim yükünden kurtarmakta, geliştiriciye sadece lokal bir işlev çağrısı gibi bir semantik sunarak kullanım kolaylığı sağlamakta ve alt taraftaki haberleşme altyapısının detaylarından soyutlamaktadır. [32]

Temel kullanımı, uzak yordam çağrısını kullanacak işlem, değiştirgeleri geçirerek süreci işletmekte ve bu arada çalışması engellenmektedir. Alt tarafta değiştirgelerin serileştirilmesi sağlanıp haberleşme kanalıyla karşıya aktarılmakta ve karşı tarafta aynı işlev vasıtasıyla değiştirgeler açılıp süreç işletilmekte, sonuçlar aynı şekilde geri karşıya iletilmekte ve engel çözülmektedir. Uzak yordam çağrısının çalışma akışı Şekil 2.8’de gösterilmiştir.



**Şekil 2.8:** Uzak yordam çağrısı.

Uzak Yordam Çağrısından beklenen saydamlığı sağlamak için aşağıdaki özelliklerin her uzak yordam çağrısını bulunduran sistemde olması gerekmektedir. [33] [32] [26]

- **Bağlama** : İşlemin çağrı yaptığı işlev ile karşı taraftaki işlevin lokasyonu arasında bağlantıyı oluşturmak zorundadır.
- **Saydamlık** : Geliştirici, yapılan çağrının uzakta bir düğümde çalıştırıldığını bilmemelidir. Bu nedenle sistem, oluşabilecek hataların tesbit edilip düzeltilmesi, değiştiregelerin aktarımı ve kural dışı durumların işlenmesi görevlerini geliştiriciye yansıtmadan gerçekleştirmek zorundadır.
- **Koşut Zamanlılık** : İşlevlerin birden fazla işlem tarafından aynı anda çağrılması durumunu haberleşmeyi etkilemeden başarıyla gerçekleştirmek zorundadır.
- **Çok Türellik** : Farklı platformlar arasındaki işlev çağrılarındaki veri yapıları ve sunumlarını algılayıp gerekli dönüşümü gerçekleştirmek zorundadır.

### 2.3.3 Dağıtık paylaşımlı bellek

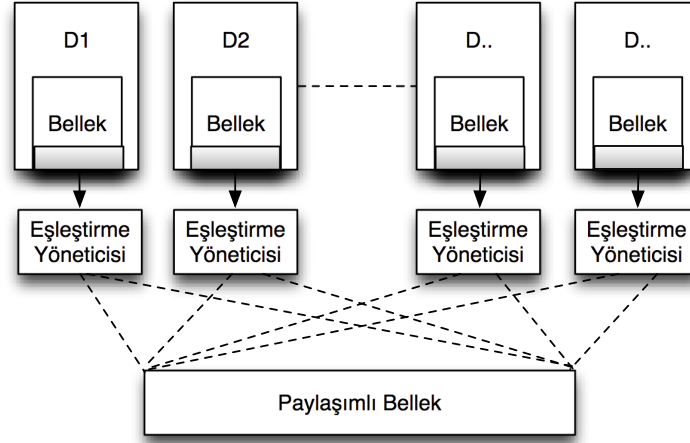
Dağıtık Paylaşımlı Bellek (*Distributed Shared Memory* (DSM)) diğer haberleşme yapılarına göre ek bir soyutlama sağlayıp geliştiriciyi bütün mesajlaşma detaylarından kurtarmakta, buna karşın işletim sisteminin karmaşıklığını artırmaktadır. Dağıtık Paylaşımlı Bellek yapısını bir ağ üzerinde farklı bilgisayarlara dağıtılmış ama dışardan bakıldığında tek bir merkezi bellek gibi davranan bir paylaşımlı bellek gibi görebiliriz. [34] [35]

Belleğe erişim sanal adres yapısı üzerinden olduğu için işlemlerin bellekle olan haberleşmeleri doğrudan adreslenebilir alanlardan okuma / yazma işlemleriyle gerçekleştirilmektedir. Yapının sağladığı yüksek soyutlama sayesinde geliştiriciler

alt taraftaki haberleşmenin yapısından bağımsız bir şekilde karmaşık veri yapıları kullanarak uygulamalarını gerçekleştirebilmektedirler.

Dağıtık Paylaşımlı Bellek modelinde haberleşme tarafındaki mesaj aktarım işlemleri işletim sistemi bünyesinde ele alınmakta, bellek yetersizliği yada tutarlılığının sağlanması gibi karşılaşılabilecek sorunlar işletim sistemi katmanında çözülmektedir.

Belleğin belirli bir bölgesine erişim, mesaj aktarım modeline benzer şekilde oku / yaz ilkeleri vasıtasıyla veriyi paylaşımlı bir değişkenden oku ya da değişkene yaz şeklinde gerçekleşmektedir. Dağıtık paylaşımlı belleğin çalışma akışı Şekil 2.9'de gösterilmiştir.



**Şekil 2.9:** Dağıtık paylaşımlı bellek.

Bir Dağıtım Paylaşımlı Bellek modelinden beklenen özellikler aşağıdaki gibidir; [34] [36] [37]

- **Parça Yapılılık** : Fiziksel lineer bir bellek alanının adreslenme aralıkları en küçük tanımlanabilir veri temsili olan **bit** den başlar ve çeşitli nesne tabanlı yapısal formlara kadar çeşitlilik gösterir. [36] Adreslenme aralığı, paylaşılan verinin parça boyutlarıyla doğrudan ilişkilidir. Küçük seçilmesi veriye erişimde daha kolaylık sağlasa da donanım maliyeti yüksektir. Özellikle fiziksel belleklerin farklı lokasyonlarda bulunması ve adres aralıklarının boyutunun küçük seçilmesi, ağda ek bir trafik yaratılmasına neden olacaktır.

Mevcut duruma bir çözüm, dağıtık bellek yönetiminin sayfa bazlı olarak yapılmasıdır. Bu çözüm, bir sanal bellek modelinin geliştirilmesini ve sayfalama

işleminin ağ üzerinden gerçekleştirilmesini gerektirmektedir. Sonuçta ortaya çıkan çözüm standart paylaşımlı bellek modeline oldukça benzemektedir.

- **Tutarlılık** : Düğümler arası aktarılan verinin tutarlılığı, en kolay şekliyle verinin, erişim isteğinde bulunan tüm bilgisayarlara aktarılmasıyla sağlanmaktadır, bu da en yalın haliyle sanal bellek modellerinde görülen "*trashing*" e benzer bir durumun oluşmasına yani verinin sürekli yer değiştirmesinden kaynaklanan başarım sorunlarına neden olmaktadır.

*Trashing* sorununun çözümü ise verinin birden çok kopyasının bilgisayarlarda bulunmasına imkan verilmesiyle gerçekleşmekte, fakat bu sefer de verinin tüm bilgisayarlardaki kopyalarının tutarlılığının sağlanması sorun olarak karşımıza çıkmaktadır. Dağıtık olmayan çok işlemcili sistemlerde sorun cep bellek tutarlılığı protokolleri [38] vasıtasıyla çözülsün de dağıtık sistemlerde çok fazla ağ trafiği yaratmakta bu konuda bir seçim yapılması söz konusu olmaktadır;

Karşılıklı dışlama ilkesi doğrultusunda ya veri tutarlılığı **katı** modeller seçilip başarım kaybı kabullenilecek ya da veri tutarlılığı **zayıf** modeller seçilip başarım artışı sağlanacaktır.

- **Senkronizasyon** : Paylaşılan veriye erişim, senkronizasyon vasıtasıyla korumalı bir şekilde yapılmak zorundadır. Senkronizasyon genellikle işletim sistemi katmanında bir senkronizasyon yöneticisi vasıtasıyla ya da uygulama tarafında geliştiricinin kontrolünde yapılmaktadır.
- **Çok Türelilik** : Farklı platformları aynı anda destekleyecek şekilde dağıtık bellek paylaşımının modellenmesi beraberinde verinin dönüştürülmesi işlemini gerektirmektedir. **Mermaid** yaklaşımındaki gibi tek sayfada sadece tek bir veri tipine izin vermek, örnek çözüm olarak gözüksün de bu ve benzer çözümler sistemin genelinde ciddi başarım sorununa neden olmaktadır.
- **Ölçeklendirilme** : Dağıtık bellek paylaşımı modeli, doğası gereği mevcut paylaşımlı bellek modeline göre fiziksel sınırlara takılmadan çok daha iyi ölçekte başarım sunabilmektedir.

### 2.3.4 Altyapının belirlenmesi

Bilgisayar kümeleri mimarisinde hedeflenen amaç; çalıştırılacak uygulamanın yüksek ölçeklikte bir düğümler yığnında yüksek başarımlı kriterlerinden ödün vermeden çalıştırılması olduğu için inceleme bu iki kriter doğrultusunda yapılmıştır.

- **Uzak Yordam Çağrısı** : Sistemde her ne kadar amaçlanan yalıtımı sağlasa da hem ek bir yazılım altyapısı gereksinimi getirmesi hem de bilgisayar kümeleri mimarisinde yalıtım, kümelerden bağımsız bir yönetim yazılımı vasıtasıyla sağlandığı için, sistemde gereksiz bir işlemci gücü harcayacağı düşünüldükten tercih edilmemiştir.
- **Dağıtık Paylaşımlı Bellek** : Haberleşme konusunda ek bir efor ve bilgi gereksinimine ihtiyaç duyulmaması ve geliştirmenin tek bir bilgisayar özelinde yapılmış gibi bir imkan sağlamasıyla diğer altyapıların önüne geçmektedir. Ancak hem işletim sistemi desteği ve ek işlemci gücü gerektirmesi hem de geliştirilecek uygulamanın içeriğine göre senkronizasyon ve önkütük tutarsızlığı gibi sorunların çözümünün ağda ciddi bir başarımlı kaybı yaşatma riskinden dolayı tercih edilmemiştir.

Yapılan inceleme sonucunda, haberleşme altyapısı kapsamında en doğru seçimin, mesaj aktarım tabanlı bir altyapının olduğu ortaya çıkmıştır. Mesaj Aktarım modelleri arasında seçim yapma konusunda belirleyici etken sınırlı atanmış bilgisayar modellerine ilişkin donanım ve yazılım bileşenlerinin farklılıkları ve her sistemde bir mesaj aktarım modelinin desteğinin olup olmadığı olmuştur. Desteğin olmadığı durumlarda da kurulum ve sürdürülebilirlik konusunda ek sorun yaratma riskleri düşünülmüştür.

Bahsedilen noktalar ışığında Mesaj Aktarım Arabirimi ile Soket mimarisi karşılaştırılmış, Mesaj Aktarım Arabirimi bilgisayar kümeleri mimarisinde sıkça kullanılsa da;

- Standarta uygun kütüphanenin her işletim sisteminde varlığının kontrolü, aktif edilmesi ya da kurulumu,

- Standart derleyici / alıřtırma ortamları dıřında kendine zg derle / baėla / alıřtır gereksinimi,
- Tez kapsamında drt farklı trde atanmıř bilgisayar ve bařarımlarının karřılařtırılacaėı masast ve dizst sisteminde desteėinin saėlanması gibi gereksinimlerden tr tercih edilmemiřtir.

Sonuç itibarıyla Berkeley Soketleri, tez kapsamında haberleřme altyapısında kullanılmak zere seilmiř, geliřtirilecek uygulama kapsamında dėmler arası kullanılacak mesajlařma tanımlarının ve serileřtirme gibi iřlemlerin soket bazlı olarak tasarlanıp kullanılması gerektiėi sonucuna varılmıřtır.

Berkeley Soket ktphanesinin iřlemler arası haberleřme (IPC) iin saėladıėı internet ve unix alan soketleri tez kapsamında desteklenecek, uygulama bileřenleri arası haberleřme tercihe gre her iki soket tipinden biri olabilecektir.

Tasarımın haberleřme altyapısı dahilinde kurulacak olan soket bazlı tasarımda, sonlandırma gibi senkronizasyon gereksinimleri iin diėer bir IPC yaklařımı olan hat da desteklenecektir.

#### **2.3.4.1 Internet soketleri**

Soket haberleřmesinin, iki bilgisayar arasında IP protokol vasıtasıyla IPC gereklemesini saėlayan trdr. Gvenli haberleřme iin TCP protokol vasıtasıyla "*stream*" tipi, gvensiz haberleřme iin ise UDP protokol vasıtasıyla "*datagram*" tipi haberleřme stilini destekler. Adresleme yapısı; IP adresi ve Port bileřenlerinden oluřmaktadır. IP adresi bilgisayarın dıř dnyaya grnen adresini, port ise bilgisayar ii uygulamalara tahsis edilen adresi temsil eder. Yerel IP adresi (Geri dnř adresi olarak da adlandırılır) vasıtasıyla bilgisayar ii IPC gereklemesinde de kullanılmaktadır.

#### **2.3.4.2 Unix alan soketleri**

Soket haberleřmesinin, bilgisayar ii IPC gereklemesini Unix dosya sistemi vasıtasıyla saėlayan trdr. Unix dosya izin yapısını kullanabildiėinden iřlemler

arası haberleşmede işleme erişim yetkisi gibi internet soketlerinde olmayan ek bir güvenilirlik desteği sağlamaktadır.

Internet soketlerinde yer alan ve bilgisayar içi haberleşme için gereksiz, TCP akış kontrolü, onay paketleri, kapsulleme gibi ek yük getirici işlemler Unix alan soketlerinde yer almaz. Adresleme yapısı; geçici oluşturulan bir dosyanın adres tanımı olarak oluşturulmasından ibarettir.

#### 2.3.4.3 Hat

Hat, işlemler arası haberleşmede tek yönlü haberleşme imkanı sağlayan bir haberleşme yöntemidir. Temel olarak ilk giren ilk çıkar (FIFO) mantığında bir kuyruk yapısında çalışır. Hattın bir ucuna sadece veri yazılabiliyorken diğer ucundan sadece veri okunabilmektedir. Genel bir uygulama sahası; işletim sistemlerinin giriş / çıkış sistemlerine entegre edilmesi ve işlemlerin sıralı bir şekilde çalışmalarına imkan veren **İş Hattı** yapısını sağlamasıdır.

#### 2.4 Mesajlaşma Altyapısı

Haberleşme altyapısı olarak Berkeley Soketlerinin seçilmesinden dolayı, düğümler arası haberleşmede kullanılacak mesajlaşma yapısının tanımlanması ve tüm düğümlere tanıtılması gerekmektedir.

Aktarılmak istenen nesnelere ya da dosyalara ilişkin verilerin serileştirilerek mesaj yapısına eklenmesi ve karşı tarafta aynı verinin mesajdan ayrıştırılarak ilgili nesnelerin ya da dosyaların oluşturulması mesajlaşma altyapısının sorumluluğu dahilindedir.

Mesaj yapısı temelde üç bileşenden oluşmaktadır;

- **İmza** : Her mesaj, aktarım esnasında karşılaşılabilecek senkronizasyon sorunlarına karşın, belirli bir karakter dizisi ile başlatılmakta, karşı tarafta imzası doğrulanmayan her mesaj yok sayılmaktadır.
- **Başlık** : Mesajın oluşturulduğu ve aktarıldığı sisteme ilişkin tür, adres, öncelik gibi bileşenlerin tanımlandığı kısımdır.
- **Blok Dizisi** : Her mesaj, aktarılmak üzere bilgiyi bloklar halinde tutmakta ve karşıya aktarmaktadır. Her mesajda mutlaka bir sonlanma bloku bulunmak zorundadır.

## 2.5 Bilgisayar Kümeleri Mimarisi Yönetim Yazılımları

Büyük ölçekli uygulamaların bilgisayar kümeleri mimarisi vasıtasıyla yüksek başarımlı olarak çalıştırılabilir olması, çok sayıda oluşumu mimarinin geliştirilmesi ve çoğullanması konusunda teşvik etmiştir. Akademik veya ticari amaçlı olarak bir çok üniversite, topluluk ve firma, donanımsal ya da yazılımsal olarak mimarinin tamamına ya da bir kısmına yönelik çalışma içerisine girmişler, kendi çözümleriyle ortaya çıkmışlardır.

Yaptığımız çalışmanın, beowulf mimarisine yakın bir çalışma olması, mimarinin donanımdan ziyade yazılım tarafına odaklanmamızı sağlamıştır. Çalışmanın sınanması amacıyla gerçekleştirilen merkez yönetim yazılımına referans olması açısından piyasada bulunan birkaç merkez yönetim dağıtımı incelenmiştir. Dağıtımlar sağladıkları araçlar vasıtasıyla aynı anda alt görevleri başlatıp, sonlandırabilmekte ve düğümlerin dinamik kontrolünü sağlayabilmektedirler. Diğer taraftan genel amaçlı yazılım oldukları için uygulama bazlı yapılandırmalar ve kurallar uygulama tarafında ele alınmak zorundadır.

Çalışmadaki odak alan yüksek başarımlı küme modeli olduğundan, incelenen yazılımlar bu doğrultuda özelleşmiş araçlara sahip merkez yönetim dağıtımlarıdır.

### 2.5.1 OSCAR

OSCAR (Open Source Cluster Application Resources) [39], yüksek başarımlı bilgisayar kümeleri oluşturmak üzere geliştirilen ve linux işletim sistemi türevleri üzerinde çalışan açık kaynak kodlu bir merkez yönetim dağıtımı projesidir. Projenin temel amacı, herkesin bedava ulaşabileceği ve kolayca yapılandırabileceği bir yazılım vasıtasıyla büyük ölçekli uygulamaların çalışmasını sağlamak, linux işletim sistemine yüksek başarımlı kümeleri desteğini kazandırmaktır. OSCAR'ın sahip olduğu araçlar;

- İş yönetim sistemi : OpenPBS / Torque+Maui
- Paralel kütüphaneler : MPICH, LAM/MPI ve PVM
- Küme izleme sistemi : Clumon ve Ganglia
- Otomatik kurulum sistemi desteği: Var

- Küme yönetim araçları : C3
- Paralel dosya sistemi : pvfs

### 2.5.2 Rocks

Rocks [40], San Diego süper hesaplama merkezinde geliştirilmiş olup, en çok kullanılan merkez yönetim dağıtımı özelliğini taşımaktadır. İşletim sistemi desteği olarak sadece RedHat linux türevlerini desteklemekte, düğümlerde bu işletim sistemine yönelik çok sayıda paketin kurulumunu gerektirmektedir. Rocks'ın en olumlu yönü, küme yapılandırmasının veri tabanı, web portalı gibi ortamlarda saklanabiliyor olması ve Java desteği sunmasıdır. Rocks'ın sahip olduğu araçlar;

- İş yönetim sistemi : Condor, SGE or OpenPBS/Torque+Maui
- Paralel kütüphaneler : MPICH, MPICH-MPD ve PVM (sadece ana bilgisayarda)
- Küme izleme sistemi : Ganglia
- Otomatik kurulum sistemi : Var
- Küme yönetim araçları : Rocks
- Paralel dosya sistemi : pvfs

### 2.5.3 Warewulf

Warewulf [41] nispeten yeni bir merkez yönetim dağıtımı olup, Rocks ve OSCAR yazılımlarının iyi yönlerini desteklemeyi amaçlamıştır. Geliştirilmesindeki ana düşünce, düğümlerdeki işletim sistemini olabildiğince küçük tutmak, aynı zamanda daha ileri seviye iş yönetim sistemleri sağlamaktır. Warewulf'u diğer merkez yönetim yazılımlarından ayıran temel özellikler, uygulamaların farklı ağlarda çalışmasını sağlaması ve disksiz çalışmayı desteklemesidir. Warewulf'un sahip olduğu araçlar;

- İş yönetim sistemi : SGE
- Paralel kütüphaneler : LAM/MPI and PVM
- Küme izleme sistemi : Ganglia
- Otomatik kurulum sistemi : Yok

#### **2.5.4 xCAT**

xCAT [42], IBM tarafından geliştirilmiş bir merkez yönetim dağıtımı olup, özellikle IBM sunucularıyla çalışmak üzere geliştirilmiş ve optimize edilmiştir. İşletim sistemi desteği olarak sadece RedHat linux türevlerini desteklemektedir. xCAT'in sahip olduğu araçlar;

- İş yönetim sistemi : OpenPBS/Torque+Maui
- Paralel kütüphaneler : MPICH, LAM/MPI and PVM
- Küme izleme sistemi : Ganglia
- Otomatik kurulum sistemi : Yok
- Paralel dosya sistemi : gpfs

#### **2.5.5 Debian küme bileşenleri**

Debian Küme Bileşenleri [43], Ruder Boskovic Enstitüsünde geliştirilen ve Debian türevi linux işletim sistemlerini destekleyen merkez yönetim dağıtımıdır. Debian Küme Bileşenlerinin sahip olduğu araçlar;

- İş yönetim sistemi : Torque
- Paralel kütüphaneler : MPICH ve LAM/MPI
- Küme izleme sistemi : Ganglia
- Otomatik kurulum sistemi : Var

#### **2.6 Paralleleştirme**

Bilgisayar kümeleri mimarisi, bağımsızlaştırılmış alt görevlerin çokça bulunduğu uygulamalarda belirgin bir başarımlı artışı sağlamasıyla, resim işleme, matematiksel hesaplama, genetik dizilim arama ve müşteri bazlı işlem yapma uygulamaları gibi birçok uygulamada ana tercih edilen mimari olmuştur.

Diğer taraftan yapısı paralelleştirmeye uygun fakat, giriş / çıkış gibi çeşitli serileştirilme gereksinimlerinden dolayı tam bağımsız çalışma imkanına sahip olmayan uygulamalar da belirli başarımlar ödünleri verilerek bilgisayar kümeleri mimarisinin üstünlüklerinden faydalandırılabilir. Burada başarıma özgü en önemli kriter seri çalışan kısmın toplam işleyişe olan oranıdır. Bu konuda yapılmış iki çalışma mevcuttur;

### 2.6.1 Amdahl'ın kanunu

Amdahl'ın 1967'de AFIPS konferansında belirttiği argüman [15]; paralelleştirilebilen bir uygulamanın çalışma hızı sisteme ne kadar ek işlemci ya da düğüm eklenirse eklensin, sistemin sahip olduğu seri çalışmak zorunda olan kısım yüzünden limitlidir. Örnek vermek gerekirse %95'i paralel çalışabilen bir uygulamanın ulaşabileceği en yüksek hız artışı, sisteme ne kadar ek işlemci ya da düğüm eklenirse eklensin 20x ile sınırlıdır. Amdahl'ın argümanını ifade etmek gerekirse,

S : Hız artışı

N : Düğüm ya da işlemci sayısı

P : Uygulamanın paralelleştirilebilir kısmı  $0 < P < 1$  olmak üzere

$$S(N) = \frac{1}{(1 - P) + \frac{P}{N}} \quad (2.1)$$

### 2.6.2 Gustafson'un kanunu

Gustafson iki noktada Amdahl'dan farklı argüman ortaya koymuştur [16]. Bunlardan ilki,

Doğada her uygulamanın paralelleştirilmeleri ile beraber hız artışı sağlayamayacağı onun yerine üretilen sonucun çoğullanabileceği noktasındadır. Buna örnek olarak hava tahmin uygulaması verilebilir, daha çok işlemci gücü tahminin daha hızlı yapılması yerine daha doğru sonuçlar üretilmesini sağlar.

Diğer bir argüman ise, paralelleştirme ile beraber uygulamanın seri çalışmak zorunda olan kısmın oranının uygulamanın tümünde sabit kalamayacağı noktasındadır. Diğer bir deyişle, sisteme eklenen her işlem gücünün, uygulamanın paralel çalışan kısmına

etkisi seri çalışan kısmına oranla daha fazla olacaktır. Gustafson'un argümanını ifade etmek gerekirse,

S : Hız artışı

N : Düğüm ya da işlemci sayısı

P : Uygulamanın paralelleştirilebilir kısmı  $0 < P < 1$  olmak üzere

$$S(N) = 1 + P(N - 1) \quad (2.2)$$

## 2.7 Sınama Uygulamasının Belirlenmesi

Paralelleştirilecek uygulamanın belirlenmesi, seri çalışan kısmın en aza indirgenmesi amaçlı olarak iki nokta düşünülerek değerlendirilmiştir;

- **Sadece işlemci bağımlı** : Uygulamanın ayrıştırılmasıyla elde edilecek alt görevlerin mümkün olduğu kadar sadece işlemci bağımlı görevler olması, GPU ya da giriş / çıkış gibi işlemleri gerektirmemesi gerekmektedir.
- **Bağımsız çalışma** : Alt görevler, mümkün olduğu kadar diğer alt görevlerle etkileşimde olmamalı, bağımsız çalışmaları gerekmektedir. Böylece ağ haberleşmesinde kaybedilecek zamanın sınama sonuçlarına etkisi en azda kalacaktır.

Her iki nokta da değerlendirildiğinde, içerisinde matematiksel bir sorunun farklı değiştirge kümeleriyle çalıştırılmasının gerektiği hayali bir uygulamanın tez kapsamında en uygun sınama uygulaması olacağı düşünülmüştür. İşlemciyi kısa sürede en yüksek başarımda çalıştırmak için görevlerin basit haliyle yüksek sayıda aritmetik işlem gerektiren bir uygulama olması gerekmektedir.

Belirtilen koşulları da desteklemesi açısından, tez kapsamında hayali bir matris çarpım uygulaması düşünülmüş, farklı değiştirge kümeleri ile tekrarlanan hesaplama işlemi, bilgisayar düğümlerinde bağımsız olarak yapılması kararlaştırılmıştır.

## 2.8 Donanımların Belirlenmesi

Akıllı telefon ve tablet bilgisayar endüstrisinin hızla gelişmesiyle beraber, atanmış donanım ve işlemci sektöründe ürün çeşitliliği artmış, artışla beraber, fiyatlarda ciddi bir azalma meydana gelmiştir. Bu sektöre yönelik geliştirilen az maliyetli işlemciler, üçüncü parti bilgisayar pazarında özellikle uygulama bazlı (video oynatıcı gibi) düşük maliyetli bilgisayar çözümlerinin geliştirilmesine olanak sağlamıştır.

Uygulama geliştiricilerine yönelik olarak tasarlanan ve geliştirme kiti olarak pazarlanan atanmış bilgisayar çözümleri, çok fazla sayıda geliştiricinin bağımsız olarak farklı platformlar ve alan için yazılım / donanım ürünü geliştirmesine olanak sağlamıştır. Robotik, medya, haberleşme sektörlerinde ya da akademik amaçlı olarak çok sayıda kullanım sahasına rastlanmaktadır.

Tez kapsamında ulaşılmak istenen amaç, paralelleştirilen uygulamanın başarımları / fiyat oranı en uygun çözüm vasıtasıyla gerçekleştirilmesi olduğu için, piyasada çeşitli alternatifleri bulunan geliştirme kitlerinden bazıları incelenecek, yazılacak uygulamanın entegrasyonu sağlanacak ve başarımları / fiyat oranları karşılaştırılacaktır.

Piyasada bulunan ve özellikleri EK-A1’de değinilen dört geliştirme kiti incelenmiş ve başarımları sonuçları karşılaştırılmıştır.

- **Raspberry PI Model B [44]** : Piyasada en fazla sayıda geliştirme imkanı ve modeli bulunan geliştirme kitidir. İşlemci SoC sinin içerisinde bulunan grafik işlemci ünitesi, kitin özellikle medya bazlı ürünlerde çokça tercih edilmesini sağlamıştır. Üretici firmanın sağladığı sınaama sonuçlarına göre pratikteki başarımları 300 Mhz Pentium 2 işlemci ile eşdeğerdir.
- **Raspberry PI 2 Model B [44]** : Raspberry PI Model B nin bir üst modeli olan bu geliştirme kiti, üretici firmanın sağladığı sınaama sonuçlarında selefine göre yaklaşık altı kat başarımları artışı sağlamaktadır.
- **BeagleBone Black Model C [45]** : Özellikle genişleme yetenekleriyle öne çıkan BeagleBone Black, geliştiricilere Raspberry PI’nin ürün yelpazesinin ötesinde ürün geliştirme imkanı vermektedir.

- **HardKernel ODROID U3 [46]** : Diğer incelenen geliştirme kitlerinin yanında işlemci gücüyle öne çıkmaktadır.

### 3. MERKEZ YÖNETİM YAZILIMI

Merkez Yönetim Yazılımı, bilgisayar kümeleri mimarisinde, kümelere iş aktarım kontrolünün tek merkezden olmasını sağlayan, kümeler üzerindeki haberleşme, dengeleme, yedekleme gibi işleri yöneten ve düğümlerin yalın bir şekilde kendisine atanan görevi yerine getirmesini sağlayan yazılımdır. Tez kapsamında bilgisayar kümeleri mimarisinin gereksinimlerine uygun bir merkez yönetim yazılımı geliştirilmiştir.

Sınama uygulamasının paralelleştirilmiş alt görevleri, belirli bir kural dahilinde merkez yönetim yazılımına tanıtılması gerekmektedir. Yazılım, tanımlanacak bir kural vasıtasıyla uygulamaya ilişkin alt görevleri düğümlere aktaracaktır.

Merkez yönetim yazılımının, farklı bilgisayarlarda bulunabilecek, paralelleştirilmiş birden fazla uygulamayı aynı anda işleyip, kontrolündeki düğümlere aktarması gerekmektedir. Bunun için yazılım, içerisinde ayrık ve bağımsız çalıştırılabilen alt bileşenlere sahip olacak ve bu bileşenler vasıtasıyla uygulamalar işletilmek üzere düğümlere aktarılacaktır.

Sistem genelindeki saydamlığı sağlamak için, yazılım; paralelleştirilen uygulamaya ilişkin içerik ya da çalıştırılma işlemleri ile ilişkili herhangi bir süreçte yer almayacak, bu konudaki tüm sorumluluk, uygulama geliştiricisinin sisteme sağladığı kurallara ait olacaktır.

Başarıma ilişkin sınama ve karşılaştırmalar için yazılım ölçümle ilişkili alt görevler ile düğümlerin başarımlar ve sürdürülebilirliklerine ilişkin raporlama kabiliyetine sahip olacaktır.

Yazılım, kontrolünü sağladığı düğümlerin yedeklenmesi ve görevlerin dengeli dağıtılmasından sorumlu olacak, düğümlerin kullanım istatistiklerini tutacaktır.

### 3.1 İş Akışı

Sürecin belli bir başarımı sağlaması için geliştirilecek yazılım; tek noktadan tam merkezi yönetim yerine, bağımsız çalışabilen alt bileşenleri vasıtasıyla, bileşenlerini farklı düğüm ya da bilgisayarlarda konumlandırarak, yoğun ağ haberleşmesi ve senkronizasyon gibi olumsuz etkilerden uzak işleyişini gerçekleyecektir.

Merkez Yönetim Yazılımı, uygulamanın ayrıştırılarak düğümlerde çalıştırılmasına ilişkin tüm süreci dağıtıcı, toplayıcı(lar) ve düğüm(ler) den oluşan üç ana bileşenle gerçekleştirmektedir.

Dağıtıcı tüm süreci yönetmek yerine sadece toplayıcı ve düğümleri yönetmekte, böylece hem tüm sürecin yönetimi kolaylaşmakta hem de yazılım içerisinde bileşen bazlı yalıtım sağlanmaktadır. Toplayıcı ve düğümler arası aktarım dağıtıcıdan bağımsızlaştığı için haberleşme ve senkronizasyon gibi ek işler ortadan kalkmıştır.

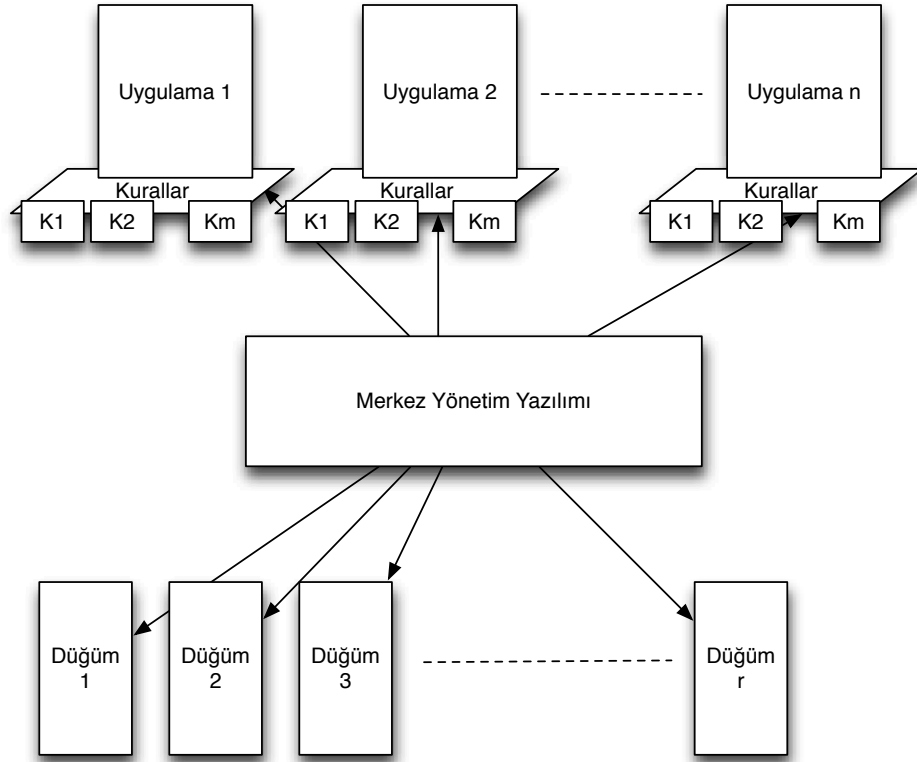
Toplayıcı, uygulamanın bulunduğu bilgisayarda çalışıp, uygulamaya ilişkin paralelleşmiş alt görevlerin tanımlandığı kural dosyalarını çözmekte ve alt görevlerin düğümlere aktarımını sağlamaktadır.

Düğümmler ise toplayıcılardan gelen kural ve kuralla ilgili dosyaların çalıştırılmasını sağlamaktadırlar.

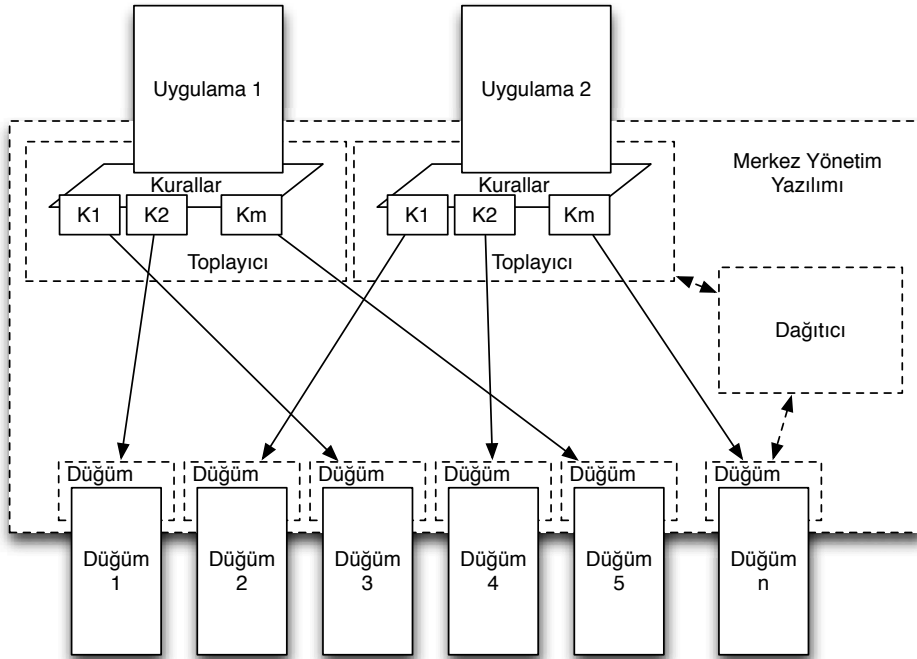
Bileşenlerin ilk çalıştırılmaları, bilgisayarların ve düğümlerin açılmasıyla otomatik tetiklenmekte ve ilklendirilmeleri bağımsız olarak gerçekleşmektedir. Merkez yönetim yazılımının işleyişi Şekil 3.1’de detaylı görünümü Şekil 3.2’da gösterilmiştir.

Uygulama alt görevlerinin düğümlere aktarılıp çalıştırılmasına ilişkin süreç aşağıdaki aşamalarla sağlanmaktadır.

1. Dağıtıcı bulunduğu haberleşme ortamına "**Uyandır**" mesajı gönderir ve bunu belirli aralıklarla yapmaya devam eder. Böylece sisteme yeni toplayıcı ve düğümlerin dinamik olarak giriş ve çıkışı sağlanmış olur.
2. Uyandır mesajını alan toplayıcı ve düğümler dağıtıcıya geri "**Hazır**" mesajı gönderirler ve mesaj bekleme kipinde beklemeye başlarlar.
3. Düğümlerden gelen "**Hazır**" mesajını alan dağıtıcı, düğümleri "**boşta**" olarak işaretleyip düğüm listesine ekler. Daha önce hazır mesajı aldığı ama son uyandır



**Şekil 3.1:** Merkez yönetim yazılımının işleyişi.



**Şekil 3.2:** Merkez yönetim yazılımının detay görünümü.

mesajında cevap alamadığı düğümleri ise listeden siler. Belirlenen güvenilirlik değerine göre yedekte tutulacak düğüm sayısını hesaplar. Bu noktadan sonra mesaj bekleme kipinde toplayıcıdan düğüm istek mesajı bekler.

4. Kullanıcılar tarafından toplayıcıların sağladığı kullanıcı arayüzü vasıtasıyla uygulamanın çalıştırılması süreci başlatılır. Bu aşamada kullanıcı uygulamaya ilişkin kurallar ve kurallarla ilişkili dosyaların, dosya sistemindeki yolunu toplayıcıya tanıtırlar ve kuralların işletilmesi süreci tetiklenir.
5. Toplayıcı önce dağıtıcıya **"Zaman"** mesajı göndererek, dağıtıcının süreçten haberdar olması sağlanır. Dağıtıcı zaman mesajını alınca, sistemin o anki zaman bilgisini, daha sonra sürecin toplam zamanını hesaplamada kullanmak üzere kaydeder.
6. Toplayıcı bu noktadan sonra kuralı işleme sürecini başlatır. Önce dağıtıcıya **"Düğüm"** mesajı göndererek boş düğüm isteğinde bulunur.
7. Düğüm mesajını alan dağıtıcı, kullanım sayılarına göre sıralanmış düğüm listesinin başından itibaren boş düğüm arar, bulamaması durumunda toplayıcıya "0" adresli **"Düğüm"** mesajı gönderir. Bulması durumunda ise önce düğümü **"ön meşgul"** olarak etiketler ve düğümün adresini **"Düğüm"** mesajı ile toplayıcıya gönderir. Aynı zamanda düğümün gelecek **"Meşgul"** mesajını tanımlı bir süre boyunca beklemek için bir sayaç kurar.
8. Düğüm mesajını alan toplayıcı, düğümün adresinin "0" olması durumunda, durumu "boşta düğüm yok" olarak kaydedip mesaj bekleme kipine geri döner. Geçerli bir düğüm adresinin olması durumunda da toplayıcı işletilecek kurala ilişkin kural dosyasını çözümleyip düğüme **"Kural"** mesajı ile birlikte gönderir.
9. Kural mesajını alan düğüm önce dağıtıcıya **"Meşgul"** mesajı göndererek dağıtıcının düğüm tarafında sürecin başladığından haberdar olmasını sağlar. Sonrasında düğüm, kural bilgisini çözümleyerek, uygulama bileşeninin çalışması için gerekli olan dosya gereksinimlerini çıkarır ve bu dosyaları dosya sisteminde arar. Bulunamayan dosyalara ilişkin içerik bilgisini **"gönderilmeyecek"** olarak işaretler. Son olarak **"gönderilecek"** olarak işaretlenmiş dosyaların MD5 [14] bilgilerini kuralda varsa kuraldan çekerek yoksa kendisi hesaplayarak toplayıcıya **"MD5"** mesajı ile gönderir.
10. Meşgul mesajını alan dağıtıcı, ilgili düğüme ilişkin kullanım değerini bir arttırır, düğümü **"meşgul"** olarak işaretler, sayacı iptal eder ve düğüm listesini kullanım

sayılarına göre tekrar sıralar. Dağıtıcı, tanımlı süre boyunca "**Meşgul**" mesajını alamazsa, ilgili düğümü listeden siler, yedek olarak işaretlenmiş düğüm sayısını bir azaltır ve yedinci adım tekrarlanır.

11. MD5 mesajını alan toplayıcı, kurala ilişkin dosya listesinin MD5 bilgileri ile gelen mesajdaki MD5 bilgilerini karşılaştırır ve eşleşme olanları "**gönderilmeyecek**" olarak işaretler. Sonrasında "**gönderilecek**" olarak işaretlenen dosyaları dağıtıcıya "**Dosya**" mesajı ile gönderir ve kural listesinde varsa diğer kural için süreci yeniden başlatır.
12. Dosya mesajını alan düğüm, mesajla gelen dosyaları kuralla ilişkili dizinlere yerleştirir ve işletim sisteminin kabuğu vasıtasıyla kural dosyasında yer alan çalıştırıcıları sırayla çalıştırır ve sonlanmalarını bekler. İşlemlerin bitmesiyle beraber dağıtıcıya "**Hazır**" mesajı gönderir.
13. Hazır mesajını alan dağıtıcı, düğümü "**boşta**" olarak işaretler ve yedek sayısını günceller.

Merkez yönetim yazılımının iş akışı Şekil 3.3'de gösterilmiştir.

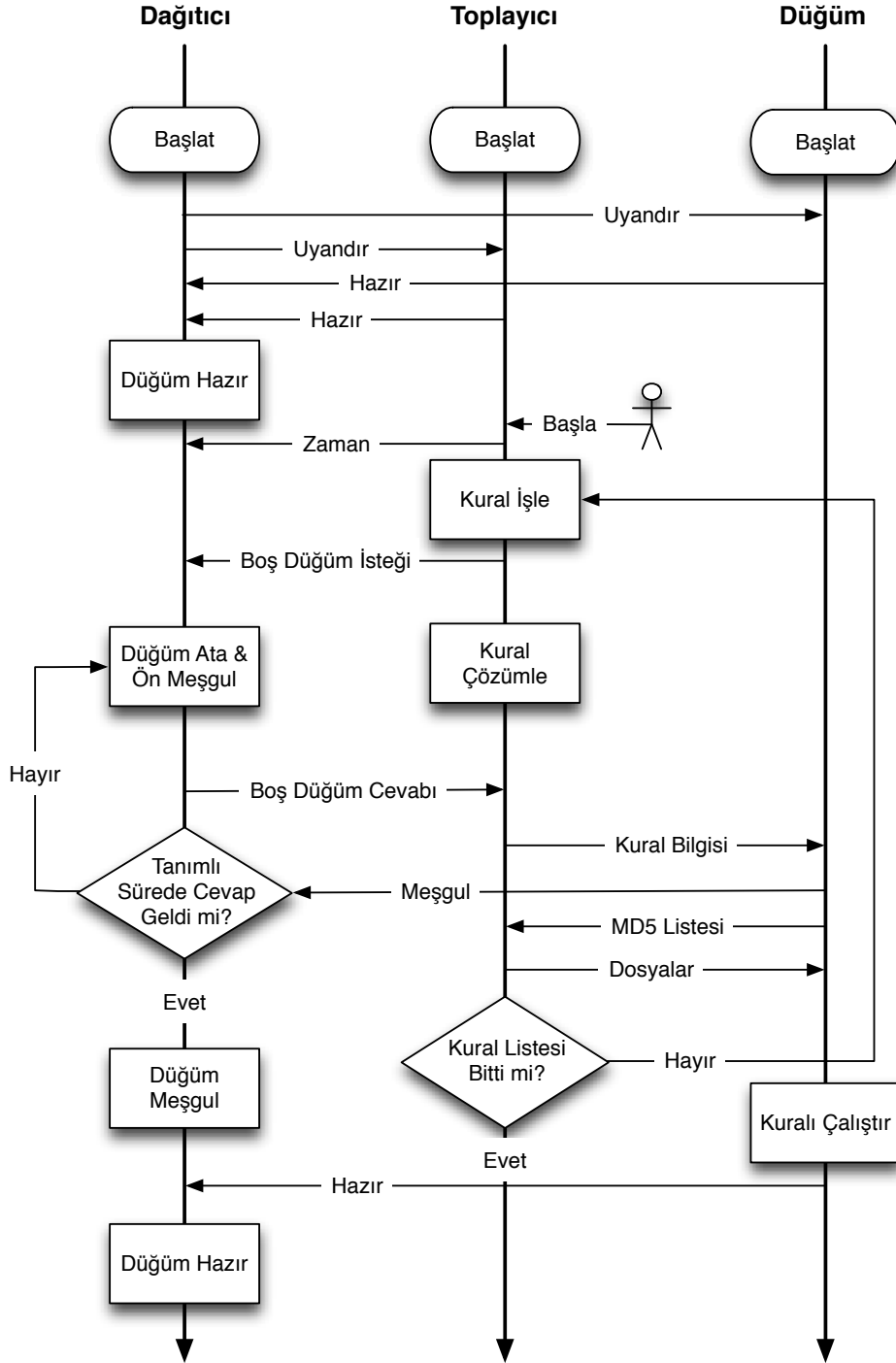
## 3.2 Ana Bileşenler

Merkez Yönetim Yazılımının ana bileşenlerinin detaylı incelemesi aşağıdaki başlıklarda sıralanmıştır;

### 3.2.1 Dağıtıcı

Merkez Yönetim Yazılımının sadece bir tane bulunan ana bileşeni olup, görevleri;

- Sistemin başlatılıp ilk konuma getirilmesi,
- Diğer bileşenlerin başlatılması için, bileşenlere "**Uyandır**" mesajının gönderilmesi ve listelerinin tutulması,
- Toplayıcılardan gelen boş düğüm isteklerinin tanımlı öncelik sırasına göre karşılanması, düğümlere ilişkin kullanım istatistiklerinin tutulması, yedek olarak bekletilecek düğümlerin kontrolü ve düğümlerin dengeli dağıtımı olarak sıralanabilir.



Şekil 3.3: İş akışı.

Dağıtıcının ilk konuma gelmesi ve diğer bileşenleri uyandırması, normal koşullarda merkez yönetim yazılımının çalıştığı bilgisayarın açılmasıyla beraber otomatik tetiklenmesi gerekmektedir, fakat tez kapsamında bu durum bir kullanıcı arayüzü vasıtasıyla akışların izlenebilmesi için elle tetikleme yoluyla gerçekleştirilecektir.

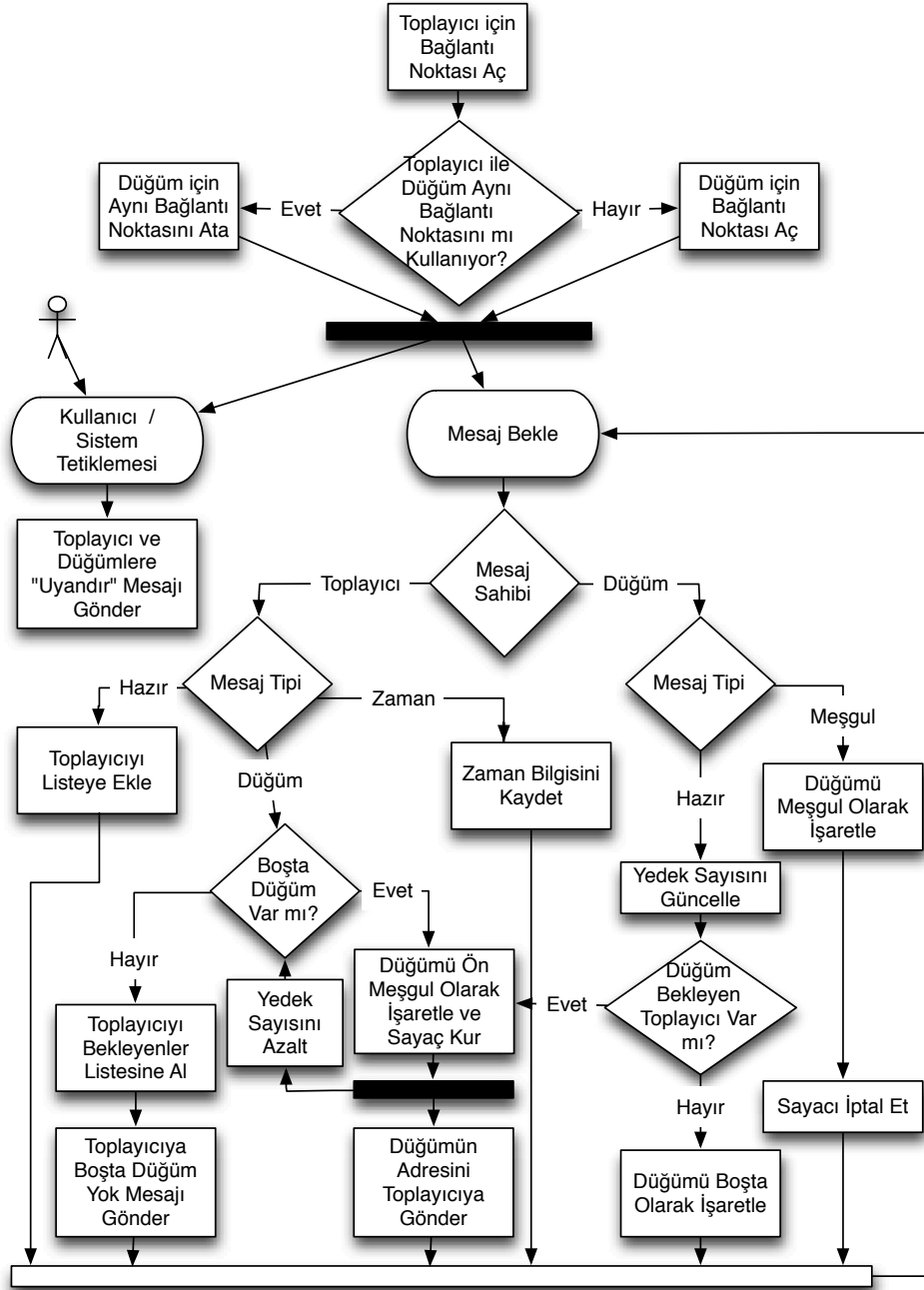
Kullanıcı arayüzü vasıtasıyla hem dağıtıcının ilk duruma getirilmesi hem de diğer bileşenlerin uyandırılması ayrı ayrı tetikleme mekanizmasıyla gerçekleştirilecektir.

Yine benzer şekilde dağıtıcının diğer bileşenlerle olan iletişim arabirimi, kullanıcı arayüzü vasıtasıyla mevcut üç haberleşme arabirimi; internet soketleri, unix alan soketleri ve hat (sınama amaçlı) arasından seçilecektir.

Dağıtıcının iş akışı Şekil 3.4’de gösterilmiştir.

Dağıtıcının çalışmasına ilişkin iş akışı dört temel aşamada değerlendirilebilir;

- **İklendirme** : Dağıtıcı, iklendirme işleminde ilk olarak, diğer bileşenlerle olan haberleşme arabirimini belirler, ve ortamdaki gelen mesajların işleneceği geriçağırım işlevini haberleşme arabirimine geçirir. Eğer ortak kullanılacak arabirim olması durumunda, yeni arabirim nesnesi yaratılmak yerine mevcut olan diğer bileşenlere haberleşme arabirimi olarak atanır. Sonrasında dağıtıcı mesaj bekleme durumuna geçiş yapar.
- **Uyandırma** : Kullanıcı arayüzü vasıtasıyla, dağıtıcı haberleşme arabirimini kullanarak diğer bileşenlere **Uyandırma** mesajı gönderir. Gönderme işlemi, haberleşme arabirimi olarak internet soketlerinin seçilmesi durumunda, tez kapsamında çalışma ortamı olarak lokal ağ seçildiği için, dağıtıcı kendi alt ağındaki tüm IP adreslerine uyandırma mesajı gönderme şeklinde gerçekleşir. Haberleşme arabiriminin Unix alan soketleri seçilmesi durumunda ise, dağıtıcı soket dosyalarının bulunduğu dizinde ".sock" dosyalarını arar, bu dosyalardan adres bilgilerini çıkarır ve elde edilen adreslere uyandırma mesajı gönderir.
- **Meşgul Bekleme** : Toplayıcıdan gelen düğüm isteğinde, atanan boş düğümün bozulma durumunu kontrol amaçlı bir sayaç kurulur ve düğümden tanımlı bir süre boyunca "**Meşgul**" mesajı beklenir. Mesaj gelmemesi durumunda düğüm listeden çıkartılır, yedek düğüm sayısı bir azaltılır ve düğüm atama işlemi tekrarlanır.



Şekil 3.4: Dağıtıcı akış diyagramı.

- **Mesaj Bekleme** : Dağıtıcı, mesaj bekleme kipinde, diğer bileşenlerden mesaj beklemeye başlar ve gelen mesajın sahibi ve tipine göre aşağıdaki adımları uygular; Mesaj sahibinin **Toplayıcı** olduğu durum;
  - **Hazır** : Mesaj, dağıtıcının uyandırma mesajının ardından toplayıcı tarafından sistemdeki varlığına ilişkin olarak gönderilir. Dağıtıcı mesajı aldığı anda, toplayıcının adresini toplayıcı listesine ekler.

- **Zaman** : Toplayıcı, uygulamaya ilişkin kuralları işlemeye başlamadan önce, sürecin başladığını zaman mesajı ile dağıtıcıya bildirir. Dağıtıcı sürecin tamamlanma sürecini hesaplamak için mesajı alır almaz, sistemin zaman bilgisini kaydeder ve sürecin düğümler tarafında bitmesini bekler.
- **Düğüm** : Toplayıcı, uygulamaya ilişkin kuralları işletmek için dağıtıcıdan düğüm mesajıyla boşa düğüm talep eder. Dağıtıcı, boşa düğüm olması durumunda, düğümü "**ön meşgul**" olarak işaretler ve düğümün adresini toplayıcıya yeni bir düğüm mesajı ile gönderir. Aynı zamanda düğümden gelecek "**Meşgul**" mesajını tanımlı bir süre boyunca beklemek için bir sayaç kurar. Boşa düğüm olmaması durumunda ise düğüm adresini "0" olarak atayıp toplayıcıya yeni bir düğüm mesajı gönderir.

Mesaj sahibinin **Düğüm** olduğu durum;

- **Hazır** : Mesaj, düğüm tarafından dağıtıcının uyandırma mesajının ardından sistemdeki varlığına ilişkin olarak ya da üzerindeki işin bittiğini haberdar etmek için gönderilir. Dağıtıcı mesajı aldığı anda, önce düğümün düğüm listesindeki varlığını kontrol eder, yoksa listeye ekler ve yedek olarak işaretlenecek düğüm sayısını günceller, varsa düğümün kullanım sürecine ilişkin zaman bilgisini kaydeder. Sonrasında toplayıcı bekleme listesinde bekleyen bir toplayıcı olup olmadığını kontrol eder. Olması durumunda düğümü "**ön meşgul**" olarak işaretler ve düğümün adresini toplayıcıya yeni bir düğüm mesajı ile gönderir. Aynı zamanda düğümden gelecek "**Meşgul**" mesajını tanımlı bir süre boyunca beklemek için bir sayaç kurar. Olmaması durumunda düğümü "**boşa**" olarak işaretler.
- **Meşgul** : Dağıtıcı mesajı aldığı anda, düğümün adresini düğüm listesinde arar, düğümü "**meşgul**" olarak işaretler, düğüme ilişkin meşgul bekleme sayacını iptal eder, düğümün kullanım değerini bir arttırır, düğüme ilişkin çalışmaya başlama zaman bilgisini kaydeder ve listeyi düğümlerin kullanım değerlerine göre azdan çok doğru yeniden sıralar.

### 3.2.2 Toplayıcı

Tanımlı kurallar çerçevesinde düğümde yürütülecek göreve ilişkin gerekli dosyalar ve değiştirge listelerini toplamak ve bunların düğümlere gönderimini sağlamaktan sorumludur. Sistemde birden fazla sayıda bulunabilir. Toplayıcı zorunlu olmamakla beraber, uygulamanın olduğu dosya sistemine erişim sağlaması gerektiğinden, bağımsız bir merkez yönetim yazılımı bileşeni olarak, uygulamanın ayrıştırıldığı bilgisayarda ayık olarak çalıştırılması hız açısından önemlidir.

Uygulamanın, düğümlere aktarılıp çalıştırılması süreci, toplayıcının tetiklenmesi vasıtasıyla başlatılmaktadır. Tetikleme işlemi, toplayıcının çalıştığı bilgisayarda bir kullanıcı arayüzü vasıtasıyla sağlanabildiği gibi dağıtıcı tarafından mesaj yoluyla da tetiklenebilmektedir. Uygulamanın paralel çalışmaya yönelik ayrıştırma işleminin kontrolü uygulamanın bulunduğu bilgisayarda olduğu için, toplayıcının tetiklenmesi aynı bilgisayarda elle yapılması tercih edilmiştir. Geliştirilen bileşendeki uygulaması da bu şekilde olacaktır.

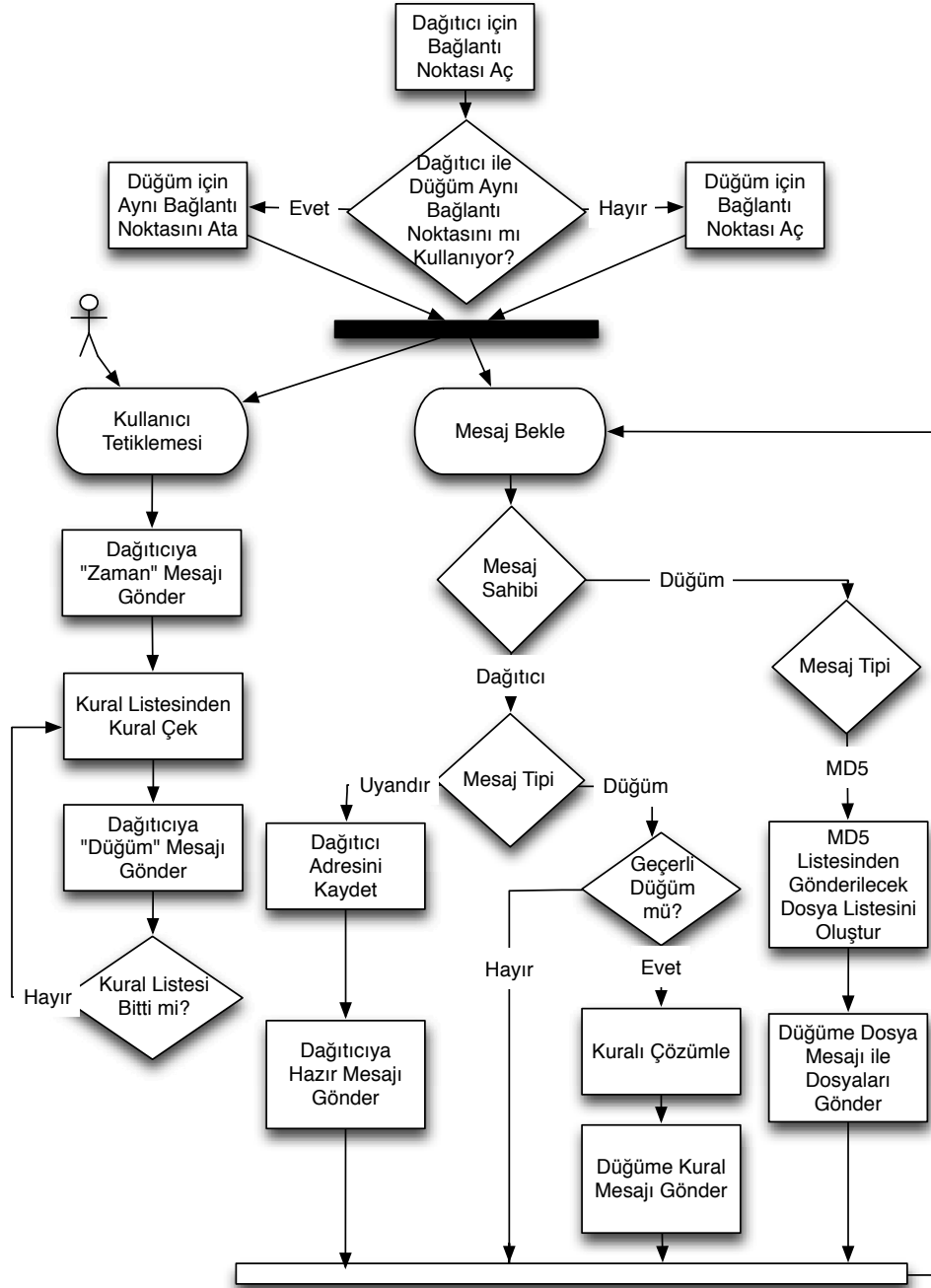
Her toplayıcı bağımsız olarak birden fazla kuralı sırayla işleyip süreci düğümlere yönlendirme kabiliyetine sahiptir. Kuralların birbirlerinden ayrıştırılması işlemi, her kuralın kendisine atanmış bir dizin ve dizinde yer alan kural dosyası vasıtası ile sağlanmıştır. Dolayısıyla toplayıcının süreci başlatması için önce işletilecek kuralın yer aldığı dizinin toplayıcıya tanıtılması gerekmektedir.

Toplayıcı, dizinde yer alan "Rule.json" adındaki kural dosyasını çözümleyerek süreci işletmektedir.

Toplayıcının iş akışı Şekil 3.5’de gösterilmiştir.

Toplayıcının çalışmasına ilişkin iş akışı dört temel aşamada değerlendirilebilir;

- **İklendirme** : Toplayıcı, iklendirme işleminde ilk olarak, diğer bileşenlerle olan haberleşme arabirimini belirler ve ortamdan gelen mesajların işleneceği geriçağırım işlevini haberleşme arabirimine geçirir. Eğer ortak kullanılacak arabirim olması durumunda, yeni arabirim nesnesi yaratılmak yerine mevcut olanı diğer bileşenlere haberleşme arabirimi olarak atanır. Sonrasında toplayıcı mesaj bekleme durumuna geçiş yapar.



Şekil 3.5: Toplayıcı akış diyagramı.

- **Uyandırılma** : Toplayıcının sisteme dahil, dağıtıcıdan gelen uyandırma mesajına hazırım cevabı dönmesiyle gerçekleşmektedir. Bu noktadan sonra toplayıcı, kullanıcıya sürecin başlatılması (tetiklenmesi) iznini vermektedir.
- **Başlatma** : Toplayıcının, uygulamanın çalıştırılması sürecini başlatması kullanıcı arayüzü vasıtasıyla kullanıcı tarafından başlatılmaktadır. Bu aşamada ilk işlem, toplayıcı, dağıtıcıya zaman mesajı göndererek sürecin başladığını haber vermekte, sonrasında tanımlı kural listesindeki kuralları tek tek sırayla işlemektedir. Her

kuralda işletilen ilk işlem dağıtıcıya düğüm mesajı gönderilerek, boş düğüm isteğinde bulunulmasıdır.

- **Mesaj Bekleme** : Toplayıcı, mesaj bekleme kipinde, diğer bileşenlerden mesaj beklemeye başlar ve gelen mesajın sahibi ve tipine göre aşağıdaki adımları uygular; Mesaj sahibinin **Dağıtıcı** olduğu durum;

- **Uyandır** : Mesaj ile toplayıcı, dağıtıcının adresini kaydeder ve dağıtıcıya uygulama süreçlerini başlatmaya yönelik hazırım mesajı döner. Kullanıcı arayüzü vasıtasıyla kullanıcılara süreci başlatma yetkisi verir.
- **Düğüm** : Dağıtıcı, toplayıcıdan gelen boş düğüm isteğine olumlu ya da olumsuz düğüm mesajı döner. Toplayıcı gelen düğüm mesajındaki adres değerinin "0" olması durumunda boş düğüm yok kaydını oluşturur ve hiçbir şey yapmadan mesaj beklemeye devam eder. Gelen mesajda geçerli bir düğüm adresi bilgisi olması durumunda işlenmek üzere beklenen kuralı çözümler ve çözümlenmiş kuralların tutulduğu adres bazlı listeye ekler. Ardından kural bilgisini ilgili düğüme gönderir.

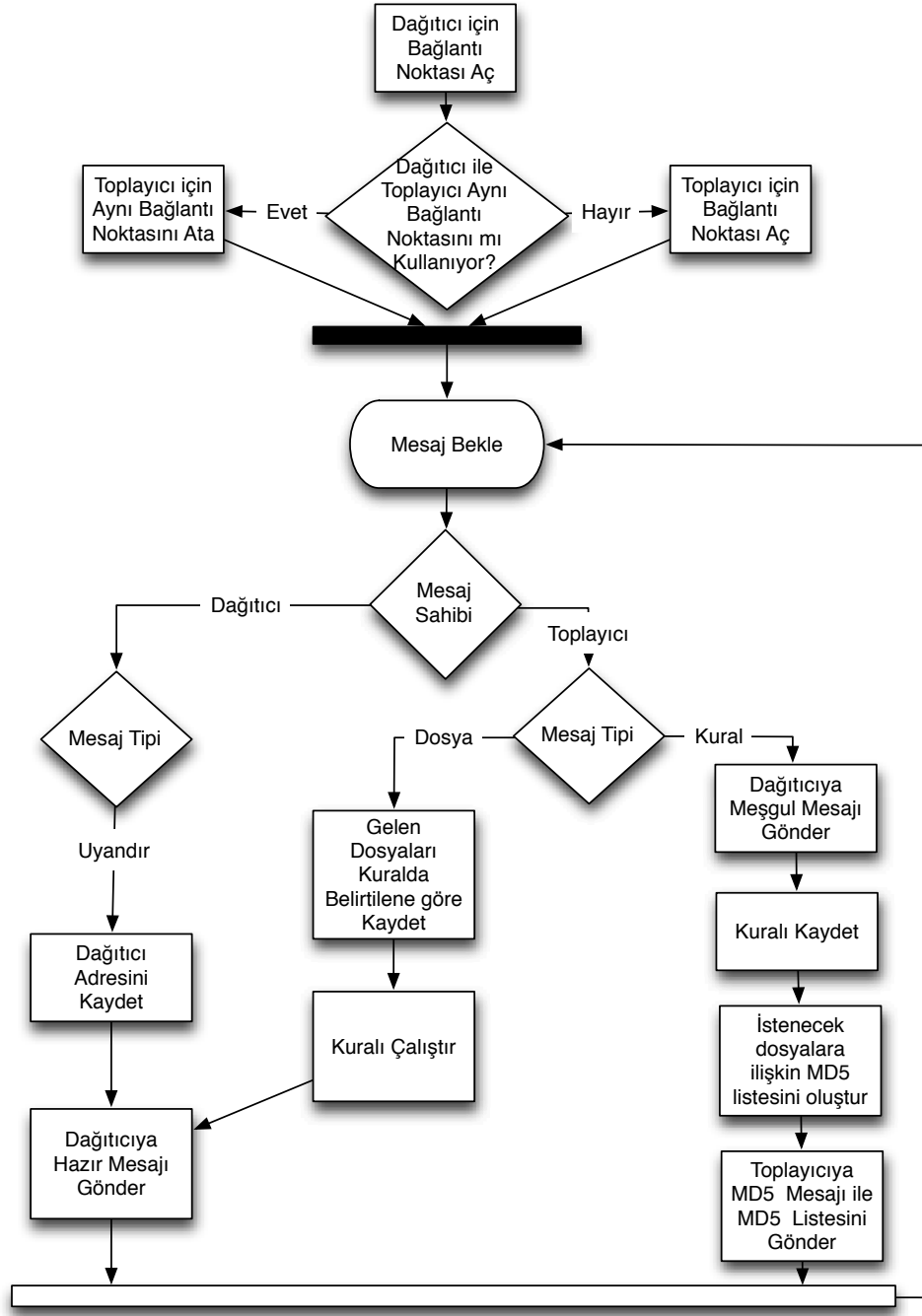
Mesaj sahibinin **Düğüm** olduğu durum;

- **MD5** : Düğümler, kuralı çözümleyerek kendilerinde olan dosyalara ilişkin MD5 listesini toplayıcıya gönderirler. Toplayıcı, aldığı listedeki her bir MD5 değerini ilgili düğüme ilişkin oluşturduğu içerik listesindeki MD5 değerleri ile karşılaştırır ve olumlu olan her sonuç için o dosyayı "**gönderilmeyecek**" olarak etiketler. Sonrasında içerik listesinde yer alan bütün "**gönderilecek**" etiketli dosyaları düğüme gönderir.

### 3.2.3 Düğüm

Toplayıcıdan gelen iş isteklerini, yine toplayıcıdan gelen kurallar çerçevesinde yürütmekten sorumludur. Sistemde birden fazla sayıda bulunabilir. Bağımsız bir merkez yönetim yazılımı bileşeni olarak atanmış bilgisayarlarda ayrıık çalışmaktadır. Kuralların yürütülmesi işlemi, işletim sisteminin sağladığı kabuk (shell) üzerinden çalıştırılabilir dosyaların sırayla çalıştırılması ile gerçekleştirilmektedir.

Toplayıcının iş akışı Şekil 3.6'de gösterilmiştir.



Şekil 3.6: Düğüm akış diyagramı.

Düğümün çalışmasına ilişkin iş akışı üç temel aşamada değerlendirilebilir;

- **İklendirme** : Düğüm, iklendirme işleminde ilk olarak, diğer bileşenlerle olan haberleşme arabirimini belirler ve ortamdan gelen mesajların işleneceği geriçağırım işlevini haberleşme arabirimine geçirir. Eğer ortak kullanılacak arabirim olması durumunda, yeni arabirim nesnesi yaratılmak yerine mevcut olanı diğer bileşenlere

haberleşme arabirimi olarak atanır. Sonrasında düğüm mesaj bekleme durumuna geçiş yapar.

- **Uyandırılma** : Düğümün sisteme dahli, dağıtıcıdan gelen uyandırma mesajına hazırım cevabı dönmesiyle gerçekleşmektedir. Bu noktadan sonra düğüm, mesaj bekleme kipine girip toplayıcıdan kural dosyası beklemektedir.
- **Mesaj Bekleme** : Düğüm, mesaj bekleme kipinde, diğer bileşenlerden mesaj beklemeye başlar ve gelen mesajın sahibi ve tipine göre aşağıdaki adımları uygular; Mesaj sahibinin **Dağıtıcı** olduğu durum;

- **Uyandır** : Mesaj ile düğüm, dağıtıcının adresini kaydeder ve dağıtıcıya kendisine gelen kuralları çalıştırmaya yönelik hazırım mesajı döner.

Mesaj sahibinin **Toplayıcı** olduğu durum;

- **Kural** : Düğüm, kural mesajını alır almaz, önce dağıtıcıya "**meşgul**" mesajı atarak, başka toplayıcıdan istek gelmesini engeller. Kuralı çözümler ve kuralın çalışması için gereken dosya listesini kendi dosya sisteminde arar, bulunan dosyalara ilişkin MD5 değerlerini bir listeye ekler ve bu listeyi MD5 mesajıyla toplayıcıya gönderir.
- **Dosya** : Mesajla beraber gelen dosyalar, dosya sistemine kaydedildikten sonra, kural dosyasında belirtilen çalıştırıcılar vasıtasıyla kural işletilir. Sürecin bitmesiyle beraber dağıtıcıya "**hazır**" mesajı atılarak toplayıcılardan yeni görev beklenir.

### 3.3 Haberleşme Altyapısının Tasarlanması

Haberleşme altyapısının tasarımındaki temel öncelik, merkez yönetim yazılımının ana bileşenlerinin birbirleriyle haberleşmesini saydam olarak sağlaması ve haberleşme arabirimleri arasındaki değişimin üstteki yapıda bir değişiklik ve farkındalık gerektirmemesidir. Bileşenler, kullanıcının belirlediği haberleşme arabirimini ve mesajların işleneceği geriçağırım işlevlerini altyapıya sağlayacaklar, bütün mesajlaşma, önceliklendirme, içerik yönetimi ve çözümleme işlemleri altyapının sorumluluğunda

bileşenlere kapalı olarak gerçekleşecektir. Böylece bileşenlerin haberleşme ile ilişkili çağrılarını sadece mesaj gönder ve bekle / çözümle işlemlerinden ibaret olacaktır.

Haberleşme altyapısında kullanılmak üzere üç teknoloji; internet soketleri, unix alan soketleri ve hat tez kapsamında desteklenecektir. Hat, soket mimarisine nazaran tek yönlü olması ve çoklu istemci desteği sunmadığı için sadece debug amaçlı olarak haberleşmede ve haberleşmenin kontrolüne yönelik bildirim tasarımında kullanılacaktır.

### **3.3.1 Zamanlayıcı**

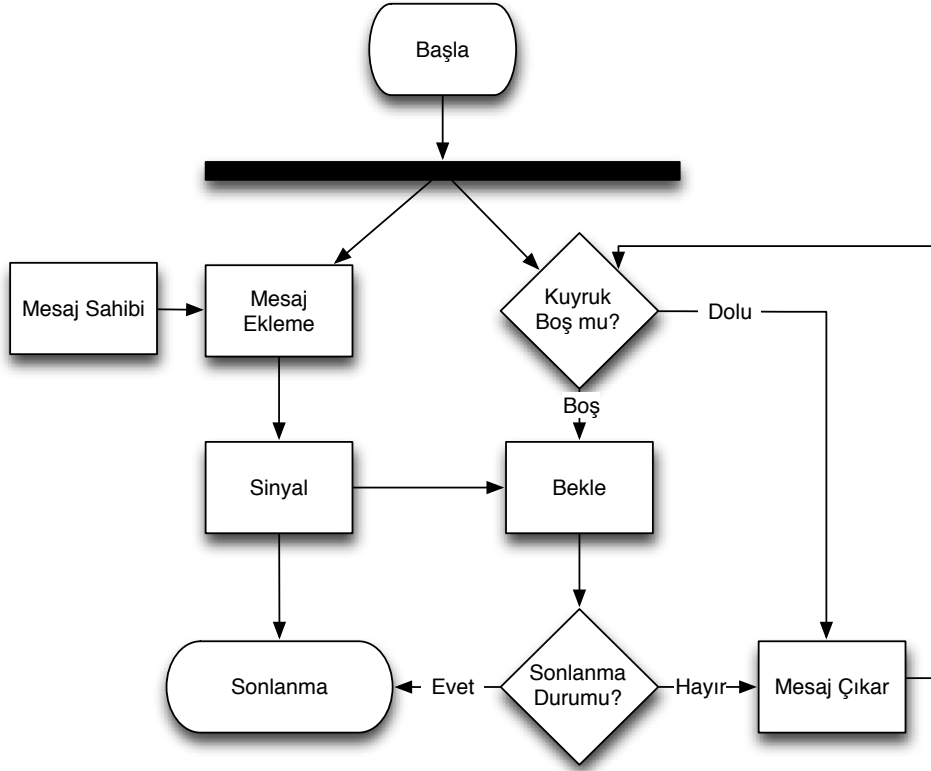
Sistemde oluşturulan bütün mesajların, bir öncelik değeri ile ilklendirilmesi ve böylece sahip oldukları öncelik değerine göre belli bir sırada işletilmeleri düşünülmüştür. Her mesajın belirli bir öncelik değeriyle bir kuyruğa alınması ve öncelik değerinin belirlediği sırada çıkmasına yönelik olarak zamanlayıcı sınıfı tasarlanmıştır. Zamanlayıcı kuyruğuna mesaj girişi dış çağrılarla olmakta, buna karşılık mesajların işlenip kuyruktan çıkması ise zamanlayıcının kendi kanalında olmaktadır. Zamanlayıcı, işletim sisteminin desteklediği senkronizasyon teknikleri vasıtasıyla kuyrukta mesaj olduğu sürece çalışmakta, mesaj kalmadığı durumda uyku durumunda tetiklenmeyi beklemektedir.

Zamanlayıcının çalışmasına ilişkin iş akışı Şekil 3.7’de gösterilmiştir.

Zamanlayıcının çalışmasına ilişkin iş akışı iki temel aşamada değerlendirilebilir;

Mesajın çıkarılıp yönlendirilmesi;

1. Haberleşme arabirimi, mesaj aktarımının önceliklendirilmesi için bir zamanlayıcı nesnesi oluşturur.
2. Zamanlayıcı, oluşur oluşmaz bir kanal açar ve bu kanalda bir mesaj kuyruğu oluşturur.
3. Kuyrukta mesajın olması durumunda, tüm mesaj kuyruğu taranır, bütün mesajların mesaj öncelikleri arttırılır ve en yüksek öncelikli mesaj kuyruktan çekilir.
4. Kuyrukta mesaj yoksa bekleme kipine geçilir ve sistemden uyandırılmak üzere sinyal beklenir.



**Şekil 3.7:** Zamanlayıcı akış diyagramı.

5. Sinyal gelmesi durumunda önce sonlanma durumu oluşmuş mu kontrol edilir.
6. Sonlanma durumu varsa zamanlayıcı kanalı kapatılır ve kaynaklar sisteme iade edilir.
7. Sonlanma durumu yoksa üç numaralı adıma dallanılır.

Mesajın eklenmesi;

1. Mesaj sahibi ana bileşen, haberleşme arabirimi vasıtasıyla zamanlayıcıya mesajı ekler.
2. Zamanlayıcı mesajı kuyruğa ekler ve kuyruğa sinyal gönderip kuyruğu bekleme kipinden çıkarır.

### 3.3.2 Mesaj

Ana bileşenler arası mesaj aktarımının soket bazlı tasarlanmasından dolayı, mesaj nesnelerinin gönderen tarafta serileştirilmesi ve alıcı tarafta ayrıştırılması işlemi proje kapsamında ayrıca ele alınmış ve gerekli yapılar oluşturulmuştur.

Mesaj nesnesi, aktarılabacak mesaja ilişkin tüm verileri belirli bir başlık ve blok yapısı ile karşıya aktarmakta ve iki taraflı sağlanacak senkronizasyon ile aktarılan verinin düzgün ayrıştırılması sağlanmaktadır. Her mesaj, tipi ne olursa olsun bloklardan oluşmakta ve bloklar halinde aktarım sağlanmaktadır.

Mesaj aktarım yapısı Şekil 3.8’de gösterilmiştir.



**Şekil 3.8:** Mesaj aktarım yapısı.

Mesaj aktarımı; önce bir imza değerinin karşıya gönderilmesiyle başlamaktadır. İmza, aktarımın senkronizasyonunu sağlamak üzere iki tarafında mutabık kaldığı iki sekizlik rastgele bir değerdir. İmza alıcı tarafta belirlenen değerle örtüşmüyorsa mesaj işleme alınmamaktadır.

Sonrasında mesajın başlık bilgisi gönderilmektedir. Mesajın başlık yapısında aşağıdaki bilgiler mevcuttur;

- Mesajın sahibi olan bileşen tipi
- Mesajın gönderildiği bileşenin tipi
- Mesajın öncelik değeri
- Mesajın tipi
- Boş alan, hizalama için
- Mesaj sahibinin adresi
- Değişken alan

Mesaj başlık yapısı Şekil 3.9’de gösterilmiştir.

Bir sonraki aşamada mesajın içeriği bloklar halinde gönderilmektedir. Her mesaj, katar tipine göre farklı sayıda bloğa sahip olabilir. Bloklar kendi içlerinde de alt alanlara ayrılabilirler fakat bunların sayısı üç ile sınırlandırılmıştır. Bloğun başlık yapısında aşağıdaki bilgiler mevcuttur;

| Kaynak Tipi   | Hedef Tipi | Öncelik | Mesaj Tipi | Padding |
|---------------|------------|---------|------------|---------|
| Kaynak Adresi |            |         |            |         |
| Değişken      |            |         |            |         |

**Şekil 3.9:** Mesaj başlık yapısı.

- Blok tipi
- Boş alan, hizalama için
- Birinci alanın boyu
- İkinci alanın boyu
- Üçüncü alanın boyu

Mesaj bloğunun başlık yapısı Şekil 3.10’de gösterilmiştir.

| Blok Tipi | Padding |
|-----------|---------|
| Boyut 1   |         |
| Boyut 2   |         |
| Boyut 3   |         |

**Şekil 3.10:** Mesaj blok başlığı yapısı.

Bir mesajda bulunabilecek toplam altı çeşit blok vardır;

1. **Dosya Bilgisi Bloğu** : Dosyanın dosya sistemindeki referans yeri ve MD5 bilgilerini kapsar.

2. **Dosya İçerik Bloğu** : Dosyanın dosya sistemindeki referans yeri, MD5 değeri ve dosyanın içeriğini kapsar.
3. **MD5 Bloğu** : Dosyanın MD5 bilgisini kapsar.
4. **Değiştirge Bloğu** : Değiştirgenin tipi ve içeriğini kapsar.
5. **Çalıştırıcı Bloğu** : Çalıştırıcı makrosunu kapsar.
6. **Sonlanma Bloğu** : Mesaj aktarımının bitirilmesine yönelik gönderilen bloktur. Mesajı alan bileşen sonlanma bloğu ile o ana kadar gelen verileri kullanıp bir mesaj oluşturur.

Mesaj aktarımında kullanılabilecek üç farklı katar tipi vardır;

1. **Kural katarı** : Düğümde çalıştırılacak göreve ilişkin kuralların yer aldığı dosyanın gönderilmesi / alınması işlemidir. Kural dosyası, dosya içerik bloğu vasıtasıyla karşıya aktarılır.
2. **İçerik Katarı** : Kural dosyasında tanımlı dosyalardan, "**gönderilecek**" olarak işaretlenenlerin, dosya içerik blokları vasıtasıyla gönderilmesi / alınması işlemidir.
3. **MD5 Katarı** : Kural dosyasında tanımlı dosyalardan, "**gönderilecek**" olarak işaretlenenlerin MD5 bilgilerinin, MD5 blokları vasıtasıyla gönderilmesi / alınması işlemidir.

### 3.3.2.1 Mesaj tipleri

Bileşenler arası haberleşmede kullanılmak üzere sekiz tane mesaj tipi tanımlanmıştır;

1. **Uyandır** : Mesaj, dağıtıcılar tarafından sistemdeki toplayıcı ve düğümlerin varlığını sorgulamak üzere gönderilen mesajdır. Sadece mesaj imzası, başlığı ve sonlandırma bloğu içermekte olup herhangi bir mesaj katarı içermez.
2. **Hazır** : Mesaj, toplayıcılar ve düğümler tarafından uyandırıldığını ya da düğümler tarafından yeni görev almaya hazır olduğunu belirtmek üzere dağıtıcıya gönderilen mesajdır. Sadece mesaj imzası, başlığı ve sonlandırma bloğu içermekte olup herhangi bir mesaj katarı içermez.

3. **Zaman** : Mesaj, toplayıcılar tarafından kural işleme sürecinin başladığını dağıtıcıya bildirmek üzere gönderilen mesajdır. Sadece mesaj imzası, başlığı ve sonlandırma bloğu içermekte olup herhangi bir mesaj katarı içermez.
4. **Düğüm** : Mesaj, toplayıcılar tarafından dağıtıcıya boş düğüm isteğinde bulunmak ya da dağıtıcı tarafından toplayıcılara boş düğüm adres bilgisini bildirmek üzere gönderilen mesajdır. Sadece mesaj imzası, başlığı ve sonlandırma bloğu içermekte olup herhangi bir mesaj katarı içermez.
5. **Kural** : Mesaj, toplayıcılar tarafından kural dosyasının düğümlere aktarılması için gönderilen mesajdır. Mesaj imzası, başlığı, kural katarı ve sonlandırma bloğu içerir.
6. **MD5** : Mesaj, düğümler tarafından bulundurdıkları dosyaların MD5 değerlerinin toplayıcıya aktarılması için gönderilen mesajdır. Mesaj imzası, başlığı, MD5 katarı ve sonlandırma bloğu içerir.
7. **Dosya** : Mesaj, toplayıcılar tarafından düğümlerde bulunmayan dosyaların düğümlere aktarılması için gönderilen mesajdır. Mesaj imzası, başlığı, içerik katarı ve sonlandırma bloğu içerir.

### 3.3.3 Arabirim

Desteklenecek haberleşme arabirimlerinin ortak yapıları düşünülerek, ana bir sınıftan türetilmeleri ve ortak işleyişlerin bu sınıf vasıtasıyla sağlanması düşünülmüştür. Görevleri;

- Giden ve gelen mesajların tutulup önceliklerine göre işlenmesini sağlayan bir mesaj kuyruğunun (zamanlayıcı) oluşturulması,
- Zamanlayıcıdan çekilen mesajların işlenmesini sağlayacak işlevin zamanlayıcıya değiştirge olarak geçirilmesi,
- Mesajın gönderim işlemini yapacak işlevin zamanlayıcıya değiştirge olarak geçirilmesi,
- Haberleşmenin sonlandırılması,
- Haberleşme arabiriminin mesaj alım ve gönderim kanallarının oluşturulması,

- Bildirim socketinin oluşturulması,
- Zamanlayıcıya mesaj eklenmesi,
- Haberleşme arabiriminin saklanıp istendiğinde dönülmesi olarak özetlenebilir.

### **3.3.4 Net**

Ana bileşenlerin bilgisayarlar arası haberleşmesinde kullanılacak haberleşme yüzü olup Arabirim sınıfından türetilmiştir. Net arabirimi TCP / IP bazlı internet socket yapısını kullanmaktadır. Görevleri;

- TCP tabanlı bir socketin oluşturulması,
- Socketin bir adres + port yapısıyla eşleştirilmesi,
- Socketin gelen çağrılarını karşılamak üzere dinlenmesi,
- Haberleşme arabiriminin mesaj alım ve gönderim kanallarının oluşturulması,
- Mesaj alım kanalında haberleşme ve bildirim socketlerinin dinlenmesi,
- Mesaj alımında yeni bir socketle yeni bir kanal oluşturulup mesajın ağdan çekilmesi,
- Sonlanma bildirimi gelmesi durumunda iletişimin sonlandırılması olarak özetlenebilir.

### **3.3.5 UnixSocket**

Ana bileşenlerin bilgisayar içi haberleşmesinde kullanılacak haberleşme yüzü olup Arabirim sınıfından türetilmiştir. Görevleri;

- Unix Alan Socketi tabanlı bir socketin oluşturulması,
- Socketin geçici bir dosya ile eşleştirilmesi,
- Socketin gelen çağrılarını karşılamak üzere dinlenmesi,
- Haberleşme arabiriminin mesaj alım ve gönderim kanallarının oluşturulması,
- Mesaj alım kanalında haberleşme ve bildirim socketlerinin dinlenmesi,

- Mesaj alımında yeni bir soketle yeni bir kanal oluşturulup mesajın ağdan çekilmesi,
- Sonlanma bildirimi gelmesi durumunda iletişimin sonlandırılması olarak özetlenebilir.

### 3.3.6 Hat

Ana bileşenlerin bilgisayar içi haberleşmesinde debug amaçlı olarak kullanılacak haberleşme yüzü olup Arabirim sınıfından türetilmiştir. Çoklu istemci desteği sağlamadığından uygulama genelinde kullanılmayacaktır. Görevleri;

- **pipe** sistem çağrısı ile bir dosya tanımlayıcısının oluşturulması,
- Haberleşme arabiriminin mesaj alım ve gönderim kanallarının oluşturulması,
- Mesaj alım kanalında haberleşme ve bildirim soketlerinin dinlenmesi,
- Mesajın ağdan çekilmesi,
- Sonlanma bildirimi gelmesi durumunda iletişimin sonlandırılması olarak özetlenebilir.

### 3.3.7 Bağlayıcı

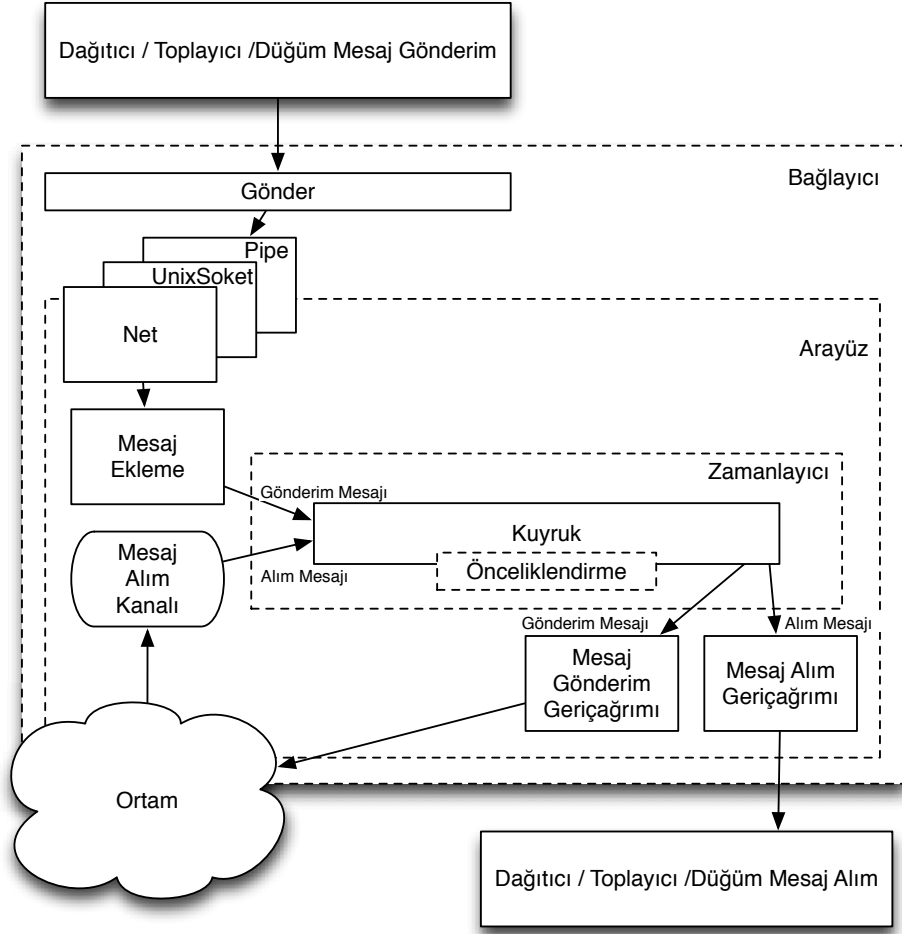
Ana bileşenlerin haberleşme arabirimleri ile tek noktadan bağlantısını sağlayan sınıftır. Bileşenlerin, haberleşme ile ilişkili işlerden soyutlanmasını sağlar. Görevi kendisine değiştirge olarak girilen arabirim tipine göre ilgili arabirim nesnelerini oluşturmak ve arabirim sınıfına ait işlevselliği, oluşturulan nesnelere özgü olarak sağlamaktır.

### 3.3.8 Mesaj akışı

Mesaj Akışı, ana bileşenlere özgü oluşturulan bağlayıcılar vasıtasıyla karşıya taşınmakta, yine benzer şekilde karşı tarafta bağlayıcıların mesajı ortamdan çekmesi ve ana bileşenlerin ilgili geriçağırım işlevlerini tetiklemesiyle işlenmektedirler.

Mesaj akışı Şekil 3.11’de gösterilmiştir.

Mesaj akışını alım ve gönderim olmak üzere iki aşamada inceleyebiliriz;



Şekil 3.11: Mesaj akışı.

### 3.3.8.1 Mesaj Gönderimi

Mesaj gönderimine ilişkin adımlar aşağıda sıralanmıştır;

- Ana bileşenin kendi kanalında kurala ilişkin izin bilgisi ile mesaj nesnesi oluşturulur.
- Mesajın alıcı ve gönderici bileşen bilgileri doldurulur.
- Mesaj tipi bilgisi doldurulur ve mesaj tipine göre öncelik değeri ve mesaj tipine özel diğer gerekli bilgiler doldurulur.
- Mesaj, bileşene atalı bağlayıcının gönderim birimi olarak atanmış haberleşme arabiriminin mesaj gönderim işlevine düşer.
- Haberleşme arabiriminin mesaj gönderim işlevi, mesajı sahip olduğu önceliğe göre işlenmek üzere arabirime bağlı zamanlayıcının kuyruğuna ekler.

- Zamanlayıcı kendi kanalında mesajı öncelik değerine göre kuyruktan çeker ve mesajı bağlı olduğu arabirimin gönderim geriçağırım işlevine yönlendirir.
- Arabirimin mesaj gönderim kanalına düşen mesaj burdan ortama salınır.

### 3.3.8.2 Mesaj Alımı

Mesaj alımına ilişkin adımlar aşağıda sıralanmıştır;

- Ana bileşene bağlı haberleşme arabiriminin mesaj alım kanalında mesaj ortamdan çekilir.
- Haberleşme arabiriminin mesaj alım işlevi, mesajı sahip olduğu önceliğe göre işlenmek üzere arabirime bağlı zamanlayıcının kuyruğuna ekler.
- Zamanlayıcı kendi kanalında mesajı öncelik değerine göre kuyruktan çeker ve mesajı bağlı olduğu bileşenin mesaj alım geriçağırım işlevine yönlendirir.
- Bileşenin mesaj alım işlevinde mesaj ayrıştırılır ve işlenir.

## 3.4 Kuralların Tanımlanması

Düğümelerde çalıştırılacak uygulamaya ilişkin dosyalar ve değiştirge kümeleri gibi tüm gereksinimler uygulama sahibi tarafından kural listesi halinde tanımlanmakta ve kural dosyalarında barındırılmaktadır. Kurallar dağıtımı yapılacak uygulamanın içeriğine dair bir bilgi barındırmazlar, uygulamaya bakışları sadece dosya ve değiştirge listeleri ve bunların çalıştırılma şekillerinden ibarettir. Dolayısıyla merkez yönetim yazılımının uygulamaya olan saydamlığını sağlamış olurlar.

Kural dosyasının tanımlanmasında kullanılan JSON [47] formatı programlama dilleri arasında veri değişimi amacıyla ortaya çıkmış bir metin formatıdır. JSON formatının oluşturulma ve çözümlenme işlemi diğer rakiplerine göre kolaylık ve hız sağladığından tez kapsamında tercih edilmiştir.

Toplayıcılar, düğümelerde çalışacak uygulama alt görevlerine ilişkin kuralı "Rule.json" adıyla uygulama ana dizininde arayıp çözümlerler. Kural dosyasının içerisinde kurala ilişkin içerikler sekmeler halinde sıralanır.

Kurallarda tanımlanan içeriklerin yönetimi, içerik sınıfı vasıtasıyla olmaktadır.

### 3.4.1 İçerik

Kurallar sahip oldukları farklı içerikleri içerik sınıfından genişleterek tanımlamaktadırlar. İçerik sınıfı kendi başına soyut bir sınıf olup sadece içeriğin geçerli bir içerik olup olmadığının işlevselliğini sağlamaktadır. İçerik sınıfından üç farklı tipte içerik sınıfı türetilmiştir.

#### 3.4.1.1 Dosya içeriği

Bileşenler arası aktarımı yapılacak dosyalara ilişkin bilgilerin kontrolünü sağlayan içeriktir. Dosyanın, dosya sistemindeki yolunu, MD5 bilgisini, geçerli bir dosya olup olmadığı ve aktarım için işaretlenme bilgisini tutmaktadır. Yeni bir dosya içeriği oluşturulduğunda, içerik nesnesi dosyanın sistemde olup olmadığını kontrol etmekte, sistemde varsa dosyanın bulunduğu yerde dosyaya ilişkin MD5 bilgisinin olduğu dosyayı kontrol etmekte yoksa oluşturmaktadır. Böylece mükerrer MD5 hesaplamalarının önüne geçilmiş olmaktadır.

Kural dosyalarında dosyalar "**files**" sekmesinin altında sıralanırlar. Her dosya sekmesi, dosyanın referans yolu ve varsa MD5 bilgisini barındırmaktadır.

#### 3.4.1.2 Değiştirge içeriği

Çalıştırılabilir dosyalara değiştirge olarak girilebilecek karakter dizilerini sağlayan içeriktir. Mevcut durumda üç farklı değiştirge tipi desteklenmektedir; **String**, **Double** ve **Long**.

Kural dosyalarında dosyalar "**parameters**" sekmesinin altında sıralanırlar. Her değiştirge sekmesi, değiştirge tipi ve değerini barındırmaktadır.

- Long tipinden değiştirge tipi ,"**l**" öneki ile tanımlanmıştır.
- Double tipinden değiştirge tipi ,"**d**" öneki ile tanımlanmıştır.
- String tipinden değiştirge tipi ,"**s**" öneki ile tanımlanmıştır.

### 3.4.1.3 Çalıştırıcı içeriği

Görevin düğümde nasıl çalıştırılacağına ilişkin komut dizisinin tanımlandığı içeriktir. İçerisinde bir makro yapısına sahip olup komut dizisi bu makro vasıtasıyla dosya ve değiştirge bilgilerini kullanabilmektedir. Makronun yapısı;

“\$(F/P)INDEX”

şeklinde tanımlanmıştır. Detayı;

- “\$” : Makro başlangıcı
- “F” : Kural dosyasındaki dosya listesine referans
- “P” : Kural dosyasındaki değiştirge listesine referans
- “INDEX” : Listedeki sıraya referans

Örnek bir kural;

```
"Rule": {  
  
  "files": [  
    ["MatrixInput_0", ""],  
    ["MatrixInput_1", "1234567890ABCDEF"],  
    ["matrixcalc_pi_4.8.3", ""]  
  ],  
  
  "parameters": [  
    "s:-f"  
  ],  
  
  "executors": [  
    "$F2 $P0 $F0",  
    "$F2 $P0 $F1",  
  ]  
}
```

Burada "\$F2 \$P0 \$F1" çalıştırıcı karakter katarı "matrixcalc\_pi\_4.8.3 -f MatrixInput\_1" olarak açılıp işletim sistemi kabuğunda çalıştırılmaktadır.

### 3.5 Kullanıcı Arayüzünün Tasarımı

Uygulamanın, tüm süreç boyuncaki aktarım ve çalışma adımlarını göstermek ve elle giriş yapılması gereken durumları gerçekleyebilmek için bir kullanıcı arayüzü tasarlanması düşünülmüştür. Ana bileşenlerin hepsinin bağımsız olarak farklı bilgisayarlarda çalışma olasılığından dolayı, kullanıcı arayüzü her bileşene özel ekran tasarımına ve işlevselliğe sahip olacaktır. Kullanıcı arayüzü temel olarak aşağıdaki işlevselliğe sahip olacaktır;

- Ana bileşenlerin, birbirleriyle olan haberleşme arabirimlerinin seçimi,
- Bileşenlerin ilklendirilmesi,
- Dağıtıcının sistemdeki toplayıcı ve düğümleri sorgulaması ve sıfırlaması,
- Toplayıcının süreci başlatması,
- Tüm mesaj akışlarının gösterimi,
- Süreç çıktılarının sağlanması olarak özetlenebilir.

Kullanıcı arayüzü kütüphanesi olarak, birçok platformda desteklenmesi ve açık kaynak kodlu olmasından dolayı **WXWidgets** kütüphanesi seçilmiştir. Fakat özellikle düğümlerdeki işlemci kaynağının en optimum şekilde kullanılabilmesi için düğümlerde terminal arayüzü kullanılacaktır.

#### 3.5.1 WXWidgets arayüzü

WXWidgets [48], çok sayıda platform ve programlama dilinin desteklendiği açık kaynak kodlu, bedelsiz dağıtılan ve kararlı bir kullanıcı arayüzü kütüphanesidir. Platformların kendi doğal programlama arayüzlerini kullandığı için diğer benzerlerine göre daha iyi bir kullanıcı arayüzü tecrübesi sunar.

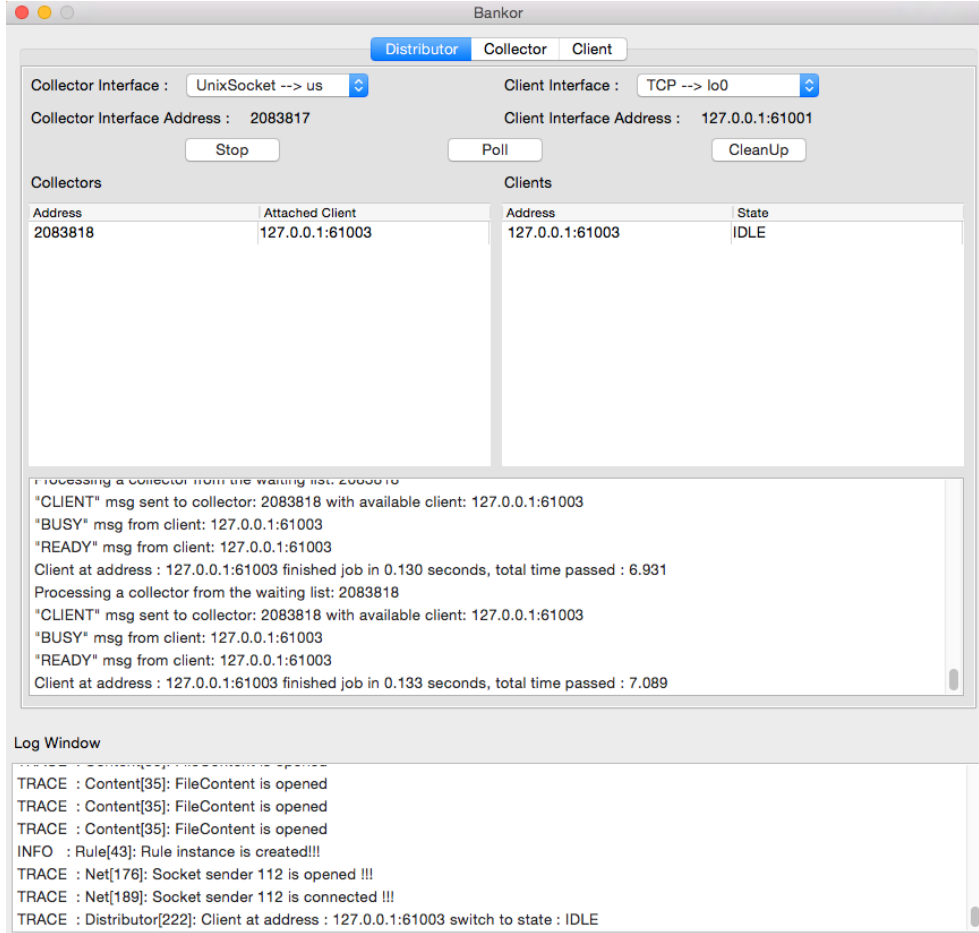
Ana bileşenlerin kullanıcı arayüzündeki temsili, kütüphanenin sunduğu sayfa yapısıyla sağlanmıştır. Böylece her bileşen farklı sayfalarda kendine özgü arayüze sahip

olabilmiştir. Bütün bileşenlerde ortak olarak kullanılan arayüzler ise aktarılan mesajların listelendiği mesaj penceresi ve sistemin aldığı kayıtların listelendiği kayıt penceresidir. Bileşenlere özgü arayüzler aşağıdaki gibi sıralanmıştır;

### 3.5.1.1 Dağıtıcı arayüzü

Dağıtıcı arayüzünde, dağıtıcının ilklendirilmesi, sistemdeki toplayıcı ve düğümlerin sorgulanması ve sıfırlanması için birer düğme bulunmaktadır. Toplayıcı listesinde toplayıcıların adresleri ve varsa kendilerine atanan düğümlerin adresleri listelenmektedir. Düğüm listesinde de, düğümlerin adresleri ve o anki durumları listelenmektedir.

Dağıtıcı arayüzünün ekran çıktısı Şekil 3.12’de gösterilmiştir.

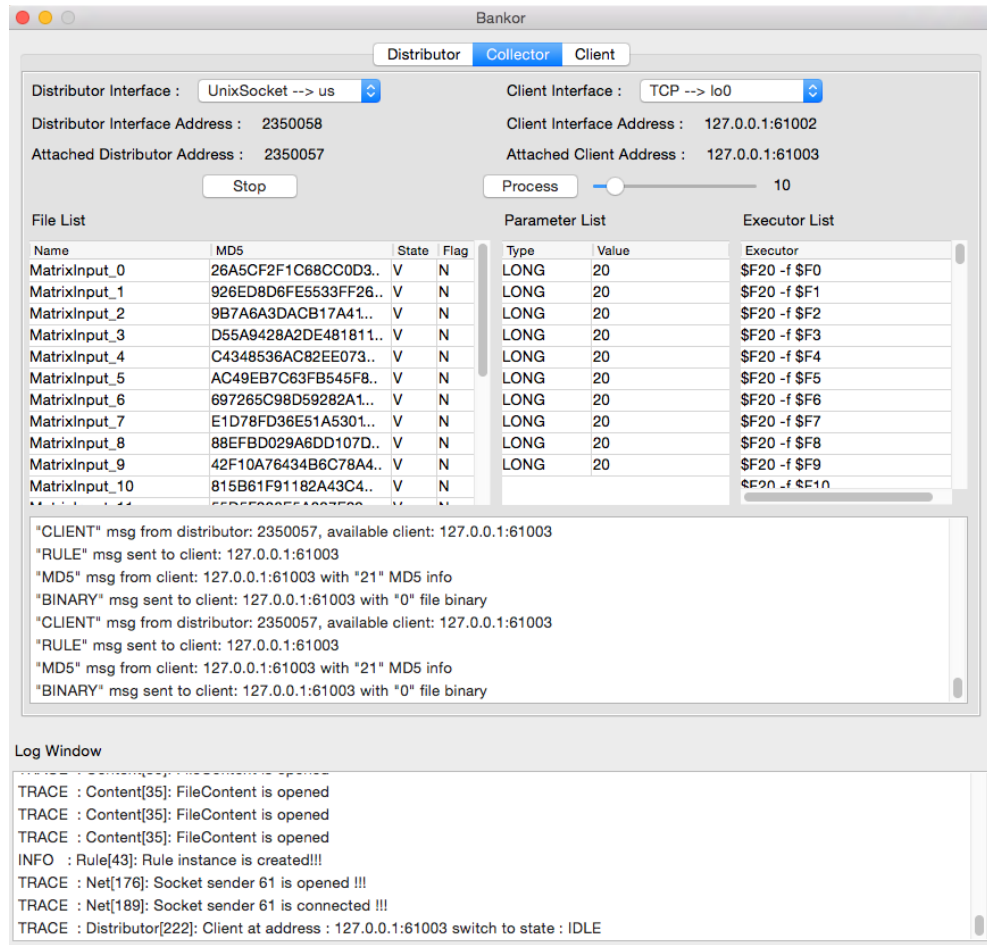


Şekil 3.12: Dağıtıcı kullanıcı arayüzü.

### 3.5.1.2 Toplayıcı arayüzü

Toplayıcı arayüzünde, toplayıcının ilklendirilmesi ve sürecin başlatılması için birer düğme bulunmaktadır. Bunların dışında bir de sürecin tekrar sayısını ayarlamaya yarayan kaydırıcı çubuğu yer almaktadır. Dosya listesinde kuralla ilişkin dosyaların adları, MD5 bilgileri, geçerlilik durumları ve gönderilme işaretleri listelenmektedir. Değiştirge listesinde değiştirgeler ve çalıştırıcı listesinde çalıştırıcı katarları listelenmektedir.

Toplayıcı arayüzünün ekran çıktısı Şekil 3.13’de gösterilmiştir.



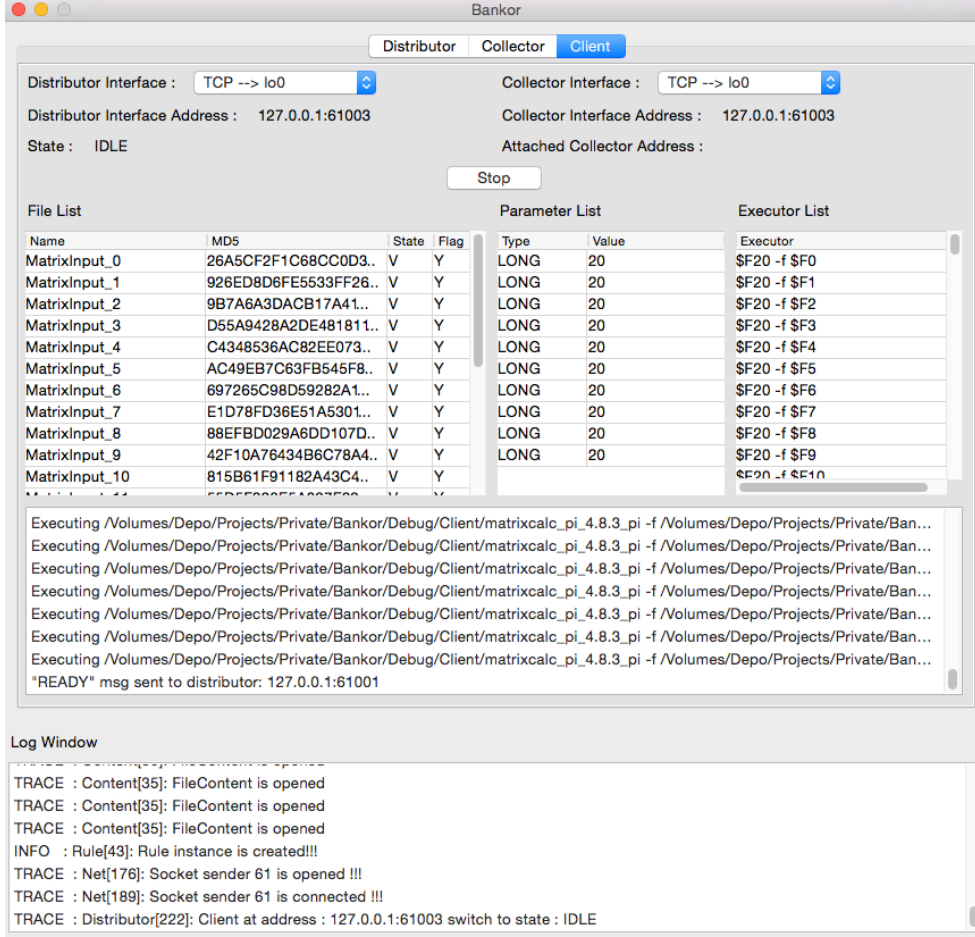
Şekil 3.13: Toplayıcı kullanıcı arayüzü.

### 3.5.1.3 Düğüm arayüzü

Düğüm arayüzünde, düğümün ilklendirilmesi için bir düğme bulunmaktadır. Dosya listesinde kuralla ilişkin dosyaların adları, MD5 bilgileri, geçerlilik durumları ve

gönderilme işaretleri listelenmektedir. Değiştirge listesinde değiştirgeler ve çalıştırıcı listesinde çalıştırıcı katarları listelenmektedir.

Düğüm arayüzünün ekran çıktısı Şekil 3.14’de gösterilmiştir.



Şekil 3.14: Düğüm kullanıcı arayüzü.

### 3.5.2 Terminal arayüzü

Özellikle düğümlerin sistem kaynaklarının en uygun şekilde kullanımı için düğümlerde WXWidgets kütüphanesi kullanmak yerine işletim sisteminin sağladığı kabuk, kullanıcı arayüzü olarak kullanılacaktır. Uygulama ile iletişim konsolda girilen karakterler dizileri olacak ve karakterlerin girişleri ile beraber bu karakterlere atanmış eylemler tetiklenecektir.

Terminal arayüzü uygulamaya girilen aşağıdaki değiştirgeler ile tetiklenmektedir;

- **l** : Haberleşme arabirimlerini listeler ve uygulamadan çıkılır.
- **c** : Uygulamayı düğüm kipinde çalıştırır.

- **q** : Uygulamayı kapatır.
- **d** : Uygulamanın kayıt seviyesini değiştirir.

### 3.6 Matris Uygulamasının Geliştirilmesi

Merkez yönetim yazılımının işleyişini sınamak üzere geliştirilecek uygulamanın sadece işlemci gücüne bağımlı olması ve kolay paralelleştirilebilmesi tercih edilmiştir. Böylece giriş / çıkış ya da ağ haberleşmesi gibi nedenlerden kaynaklanan gecikmelerin sistem başarımına etkisi en azda kalmış olacaktır. İstenen özelliği sağlamak için uygulama olarak matematiksel işlem tabanlı olan matris hesaplaması tercih edilmiştir.

Matris uygulaması varsayımsal olarak birçok dizin içerisinde farklı boyutlarda iki matrisin yer aldığı çok sayıda değiştirge dosyasının ardışıl olarak okunması, çarpımların yapılması ve çıktıların kabuk ekranında kullanıcıya sunulması işlemlerini gerçekleştirmektedir. Tez kapsamında, uygulama her bir dizindeki hesaplama sürecini tanımlanacak kurallarla beraber düğümlerde ayrı ayrı gerçekleyecektir.

Geliştirilecek uygulama iki yönlü olarak çalışacaktır;

- İlk çalışma kipinde uygulamanın komut satırından değiştirge almadığı ya da sadece tek bir dosyayı değiştirge aldığı durumdur. Bu çalışma kipi uygulamanın yukarıda bahsi geçen, bir dizindeki tüm ya da değiştirge girilen matris dosyalarının içeriklerinin çarpılarak sonucun kabuğa basıldığı durumdur.
- İkinci çalışma kipinde ise uygulama, rastgele ya da değiştirge girilen büyüklüklerde içerisinde iki matrisin yer aldığı dosya üretmektedir. Matrislerin içerdiği değerler ise 1 ile 600 arasındaki sayılardan oluşmaktadır.



#### 4. SINAMALAR ve SONUÇLAR

Geliştirilen merkez yönetim yazılımı, daha önce değinilen donanımlar kullanılarak sınanmış ve yaptığımız çalışmanın, benzer çalışmalara oranla sağladığı katkı noktasında üstün - eksik yönleri irdelenmiştir. Çalışma, özellikle başarımlar / fiyat dengesini en optimum noktada tutma gayretli olduğundan, çalışılan platformdan kaynaklanan bazı olumsuzluklardan çalışmanın tümüne etkisini görme anlamında bahsedilmiş, fakat başarımlar kriterleri arasında değerlendirilmemiştir.

Sınamaların gerçekleştirilmesi, yerel ağ ortamı oluşturularak gerçekleştirilmiş, bileşenler arası iletişim ethernet arabirimi üzerinden bir "switch" vasıtasıyla sağlanmıştır.

##### 4.1 Platform

Seçilen platformlarda işletim sistemi olarak, sunucu işlevi gören bilgisayarlarda Mac OS Yosemite, düğümlerde Linux türevleri tercih edilmiştir. Her iki işletim sisteminin de Unix tabanlı olması, geliştirilen uygulamanın hem x86 hem de ARM mimarilerinde derlenmesi işlemlerinde büyük kolaylık sağlamıştır.

Dağıtıcı, toplayıcı ve düğüm olarak sınanmak üzere,

- **Apple MacBook Pro dizüstü bilgisayar :**

İşlemci : Dört çekirdekli i7 2.0 Ghz

Bellek : 8 GB DDR3

İşletim Sistemi : Mac OS Yosemite

Derleme Ortamı : gcc 4.9

- **Masaüstü bilgisayar :**

İşlemci : Dört çekirdekli i7 3.4 Ghz

Bellek : 16 GB DDR3

İşletim Sistemi : Mac OS Yosemite

Derleme Ortamı : gcc 4.9

Sadece düğüm olarak sınanmak üzere,

- **Raspberry PI Model B** : (3 Adet)

İşlemci : Tek çekirdekli ARM11 700 Mhz

Bellek : 512 MB SDRAM

İşletim Sistemi : Debian Wheezy (Linux)

Derleme Ortamı : gcc 4.8.3

- **Raspberry PI 2 Model B** : (1 Adet)

İşlemci : Dört çekirdekli ARM Cortex A7 900 Mhz

Bellek : 1 GB DDR2

İşletim Sistemi : Debian Wheezy (Linux)

Derleme Ortamı : gcc 4.8.3

- **BeagleBone Black Model C** : (1 Adet)

İşlemci : Tek çekirdekli ARM Cortex A8 1 Ghz

Bellek : 512 MB DDR3

İşletim Sistemi : Debian Wheezy (Linux)

Derleme Ortamı : gcc 4.8.3

- **HardKernel ODROID U3** : (1 Adet)

İşlemci : Dört çekirdekli ARM Cortex A9 1.7 Ghz

Bellek : 2 GB DDR2

İşletim Sistemi : Ubuntu (Linux)

Derleme Ortamı : gcc 4.8.3

donanımları tercih edilmiştir.

## 4.2 Uygulama

Yapılan çalışmada, sinama uygulaması olarak 0 ile 600 arasında rasgele değerler alan, rastgele uzunlukta matrislerin yer aldığı dosyalar, düğümlerde çalıştırılıp, düğümlerin başarımları ölçülecektir. Matris uygulaması vasıtasıyla 20 adet matris girdisine sahip dosya oluşturulmuştur. Sinama uygulamasından beklenen, bütün dosyalardaki matris girdilerin hesaplamasının yapıp bir çıkış dosyasına yazılmasıdır.

Düğüme aktarılma işleminde işlemin tekrar sayısı, kullanıcı arayüzü üzerinden ayarlanabilmektedir. İlk aktarım matris çarpım işlemini gerçekleyen uygulama ile, matris girdilerinin yer aldığı dosyalar iken sonraki aktarımlarda geliştirilen yazılımın sahip olduğu MD5 kontrol mekanizması sayesinde uygulama dosyasının tekrar gereksiz aktarımı engellenmektedir.

Uygulamanın aktarım sürecinden sonra, düğümden çalışan uygulama sırayla tüm matris girdi dosyalarını okuyup çözümlemekte ve çarpım işlemini gerçeklemektedir. Oluşan sonucu ise bir çıkış dosyasına yazmaktadır.

Düğümlere aktarılacak uygulama dosyalarına ilişkin bilgiler tablo 4.1’de, girdi dosyalarına ilişkin bilgiler tablo 4.2’de gösterilmiştir.

**Çizelge 4.1:** Uygulama bilgileri

| Donanım                  | Dosya adı    | Dosya Boyutu |
|--------------------------|--------------|--------------|
| x86                      | matrixcalc_m | 16.6 KB      |
| Raspberry PI Model B     | matrixcalc_r | 89.8 KB      |
| Raspberry PI 2 Model B   | matrixcalc_p | 78.2 KB      |
| BeagleBone Black Model C | matrixcalc_b | 78 KB        |
| HardKernel ODROID U3     | matrixcalc_o | 78.6 KB      |

Matris Girdileri ;

Matris girdilerinin toplam boyutu : **3202.8 KB**

## 4.3 Sınamalar

Çalışmada ana bileşenlerin sinaması, hem aynı bilgisayar hem de farklı bilgisayarlar kullanılacak şekilde kurgulanmış ve sinama sonuçları elde edilmiştir.

Sinama ortamları üç ana yapılandırmadan oluşturulmuştur :

**Çizelge 4.2: Matris bilgileri**

| Dosya adı      | Matris 1 Boyutu | Matris 2 Boyutu | Dosya Boyutu |
|----------------|-----------------|-----------------|--------------|
| MatrixInput_0  | 223x285         | 285x223         | 451.8 KB     |
| MatrixInput_1  | 12x159          | 159x12          | 13.6 KB      |
| MatrixInput_2  | 3x205           | 205x3           | 4.4 KB       |
| MatrixInput_3  | 146x23          | 23x146          | 23.8 KB      |
| MatrixInput_4  | 279x78          | 78x279          | 154.9 KB     |
| MatrixInput_5  | 257x216         | 216x257         | 394.5 KB     |
| MatrixInput_6  | 97x21           | 21x97           | 14.4 KB      |
| MatrixInput_7  | 267x49          | 49x267          | 92.8 KB      |
| MatrixInput_8  | 252x228         | 228x252         | 408.5 KB     |
| MatrixInput_9  | 39x279          | 279x39          | 77.3 KB      |
| MatrixInput_10 | 149x5           | 5x149           | 5.3 KB       |
| MatrixInput_11 | 184x118         | 118x184         | 154.3 KB     |
| MatrixInput_12 | 1x257           | 257x1           | 1.8 KB       |
| MatrixInput_13 | 287x157         | 157x287         | 320.2 KB     |
| MatrixInput_14 | 152x147         | 147x152         | 158.9 KB     |
| MatrixInput_15 | 260x255         | 255x260         | 471.5 KB     |
| MatrixInput_16 | 81x209          | 209x81          | 120.3 KB     |
| MatrixInput_17 | 169x263         | 263x169         | 315.9 KB     |
| MatrixInput_18 | 2x159           | 159x2           | 2.2 KB       |
| MatrixInput_19 | 33x70           | 70x33           | 16.4 KB      |

1. Dağıtıcı -> Masaüstü Bilgisayar, Toplayıcılar -> Masaüstü Bilgisayar, Düğümler -> Masaüstü Bilgisayar
2. Dağıtıcı -> Masaüstü Bilgisayar, Toplayıcılar -> Masaüstü Bilgisayar, Düğümler -> Atanmış Bilgisayar
3. Dağıtıcı -> Masaüstü Bilgisayar, Toplayıcılar -> Dizüstü Bilgisayar, Düğümler -> Atanmış Bilgisayar

Ana bileşenlerin aynı bilgisayarda olduğu durumlarda, birbirleri arasındaki haberleşme yöntemi olarak Unix Soketleri, farklı bilgisayarlarda olduğu durumlarda ise İnternet Soketleri tercih edilmiştir.

Aktarımın sisteme getirdiği etkiyi en aza indirmek için sınaıa tekrarları aynı matris girdileriyle yapılmış, dolayısıyla düğümlerde çalıştırılacak uygulamaya ek olarak matris girdilerinin de ilk aktarım dışında gönderimi engellenmiştir.

Tablo gösteriminde kolaylık olması açısından,

Masaüstü Bilgisayar - **M** ile

Apple MacBook Pro Dizüstü Bilgisayar - **D** ile

Raspberry PI Model B - **R** ile

Raspberry PI 2 Model B - **P** ile

BeagleBone Black Model C - **B** ile

HardKernel ODROID U3 - **O** ile temsil edileceklerdir.

**Tekrar** : Matris çarpım işleminin düğümler tarafından toplamda kaç kere tekrarlanacağını temsil etmektedir. Yapılan sınamalarda tekrar sayısı **10** değerine atanmıştır.

**İş Süresi** : Her bir alt görevin düğümde başlaması ve bitmesi arasında geçen süreyi

**Toplam süre** : Toplayıcının süreci başlatmasıyla beraber, aktarımlar ve mesajlaşmalar dahil geçen toplam süreyi temsil etmektedir.

#### 4.3.1 Sınama 1

Yapılandırma :

Dağıtıcı -> Masaüstü Bilgisayar

Toplayıcılar -> Masaüstü Bilgisayar

Düğümler -> Masaüstü Bilgisayar

Tek düğümlü yapılan sınamanın sonucu Çizelge 4.3’de, iki düğümlü yapılan sınamanın sonucu Çizelge 4.4’de ve üç düğümlü yapılan sınamanın sonucu Çizelge 4.5’de gösterilmiştir.

**Çizelge 4.3:** Sınama 1 - Tek düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | M     | 1.102 s   | 1.448 s     |
| 2      | M        | M         | M     | 0.693 s   | 2.180 s     |
| 3      | M        | M         | M     | 0.710 s   | 2.930 s     |
| 4      | M        | M         | M     | 0.710 s   | 3.680 s     |
| 5      | M        | M         | M     | 0.745 s   | 4.465 s     |
| 6      | M        | M         | M     | 0.744 s   | 5.249 s     |
| 7      | M        | M         | M     | 0.759 s   | 6.050 s     |
| 8      | M        | M         | M     | 0.753 s   | 6.842 s     |
| 9      | M        | M         | M     | 0.746 s   | 7.622 s     |
| 10     | M        | M         | M     | 0.760 s   | 8.418 s     |

**Çizelge 4.4:** Sınama 1 - İki düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | M     | 1.178 s   | 1.625 s     |
| 2      | M        | M         | M     | 1.183 s   | 1.629 s     |
| 3      | M        | M         | M     | 1.040 s   | 2.721 s     |
| 4      | M        | M         | M     | 1.040 s   | 2.721 s     |
| 5      | M        | M         | M     | 1.180 s   | 3.953 s     |
| 6      | M        | M         | M     | 1.180 s   | 3.953 s     |
| 7      | M        | M         | M     | 1.633 s   | 5.638 s     |
| 8      | M        | M         | M     | 1.632 s   | 5.638 s     |
| 9      | M        | M         | M     | 0.988 s   | 6.673 s     |
| 10     | M        | M         | M     | 0.988 s   | 6.673 s     |

**Çizelge 4.5:** Sınama 1 - Üç düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | M     | 1.859 s   | 2.199 s     |
| 2      | M        | M         | M     | 1.861 s   | 2.201 s     |
| 3      | M        | M         | M     | 1.861 s   | 2.201 s     |
| 4      | M        | M         | M     | 1.399 s   | 4.140 s     |
| 5      | M        | M         | M     | 1.535 s   | 4.276 s     |
| 6      | M        | M         | M     | 1.562 s   | 4.303 s     |
| 7      | M        | M         | M     | 1.413 s   | 5.606 s     |
| 8      | M        | M         | M     | 1.371 s   | 5.686 s     |
| 9      | M        | M         | M     | 1.402 s   | 5.752 s     |
| 10     | M        | M         | M     | 0.827 s   | 6.472 s     |

Sınama 1 sonucu göstermektedir ki, aynı bilgisayarda düğüm sayısını ne kadar arttırsak arttıralım, simule edilen düğümlerin tek işlemciyi paylaşımlı olarak kullanmaları ve girdi dosyalarını dosya sisteminden ancak sıralı okuyabilmeleri gibi birlikte çalışmayı engelleyen seri çalışma zorunluluklarından dolayı;

$$\text{Toplam Başarım} \neq \text{Düğüm Sayısı} \times \text{Düğüm Başarımı} \quad (4.1)$$

olmaktadır.

Sınama sonucundan da görüleceği üzere tek düğümlü sisteme göre iki düğümlü sistemin başarımlar katkısı yaklaşık **%21** iken düğüm sayısı üçe çıktığında bu katkı **%23**'de kalmıştır. Üç düğümlü sistemin beklenenin daha da altında kalmasının nedeni son tekrarın tek bir düğüm tarafından işlenmesi, diğer iki düğüme yaptırılacak işlemin kalmamış olmasıdır. Dolayısıyla aslında elde edilen toplam bitirilme süresi 10 - 11 ve 12 tekrar için üç düğümlü sistemde birbirine çok yakın sonuç üreteceğidir.

Üç düğümlü sistemin tek düğümlü sisteme göre en optimum oranda sağlayacağı katkıyı üçe bölünen en yüksek tekrar sayısında (9) bakacak olursak, katkı **%25** olmaktadır.

Sınamada ortaya çıkan sonuç, tek bilgisayarda düğümlerin çok kanallı olarak simüle edilmesi, beklenen başarımın yarısının da altında bir başarım sağlamaktadır. Burada başarımın düşük çıkmasındaki diğer önemli bir değişirge, düğümler referans bilgisayarda bağımsız görevler olarak çalıştırıldıkları için işlemcinin çekirdeklerini verimli kullanabilmeleri, işletim sisteminin dağıtım başarısına dayanmaktadır. Sınama kapsamında, bu konu ile ilgili ayrı bir çalışma yapılmamıştır.

#### 4.3.2 Sınama 2

Yapılandırma :

Dağıtıcı -> Apple MacBook Pro Dizüstü Bilgisayar

Toplayıcılar -> Apple MacBook Pro Dizüstü Bilgisayar

Düğümler -> Apple MacBook Pro Dizüstü Bilgisayar

Tek düğümlü yapılan sınamanın sonucu Çizelge 4.6'de gösterilmiştir.

**Çizelge 4.6:** Sınama 2 - Tek düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | D        | D         | D     | 1.069 s   | 1.120 s     |
| 2      | D        | D         | D     | 0.930 s   | 2.082 s     |
| 3      | D        | D         | D     | 0.759 s   | 2.873 s     |
| 4      | D        | D         | D     | 0.851 s   | 3.756 s     |
| 5      | D        | D         | D     | 0.672 s   | 4.457 s     |
| 6      | D        | D         | D     | 0.554 s   | 5.042 s     |
| 7      | D        | D         | D     | 0.836 s   | 5.907 s     |
| 8      | D        | D         | D     | 0.957 s   | 6.893 s     |
| 9      | D        | D         | D     | 0.770 s   | 7.693 s     |
| 10     | D        | D         | D     | 0.796 s   | 8.518 s     |

Apple MacBook Pro dizüstü bilgisayarı sahip olduğu işlemci ve bellek donanımı olarak kağıt üstünde masaüstü bilgisayarın **%58**'i bir başarıma sahip olsa da sınama sonucu göstermektedir ki, standart bilgisayarlar arka planda çalıştırmak zorunda oldukları diğer yazılım bileşenleri, kontrol ettikleri farklı giriş / çıkış birimleri gibi

faktörler yüzünden, çalıştırılmak istenen uygulamaya kendilerinden beklenen işlemci gücünü ayıramamaktadırlar.

Yukarıdaki örnekte de görüleceği üzere, masaüstü bilgisayarı aslında çok daha iyi bir başarımları sergileyebilecekken, sahip olduğu ek donanım ve bunlara bağımlı yazılım bileşenleri yüzünden beklenenin altında bir başarımları göstermiştir. Durum tespiti yapıldığında, masaüstü bilgisayarı, dizüstü bilgisayarında olmayan ve arka planda sürekli çalışan

- Ek iki adet aktif kablolu ve kablosuz ağ sürücüsüne ve uygulamaların bu sürücüler vasıtasıyla internette sürekli bir bağlantı kurma durumuna,
- NTFS ve EXT dosya sistemleri desteği için FUSE disk sürücüsüne,
- Çeşitli amaçlarla arkaplanda kalıp, daimi işlemci gücü tüketen servislere,

sahiptir. Bunun dışında dizüstü bilgisayarı sahip olduğu SSD disk ile disk okuma / yazma başarımları olarak masaüstü bilgisayarından çok daha iyi başarımları göstermektedir.

Sonuç itibarıyla yapılan çalışmada sınanan uygulama kapsamında dizüstü bilgisayarı, masaüstü bilgisayarının yaklaşık **%99**'u gibi bir başarımları sergilemiştir.

### **4.3.3 Sınama 3**

Yapılandırma :

Dağıtıcı -> Masaüstü Bilgisayar

Toplayıcılar -> Masaüstü Bilgisayar

Düğümmler -> Raspberry PI Model B (1 - 3)

Tek düğümlü yapılan sınamanın sonucu Çizelge 4.7'de, iki düğümlü yapılan sınamanın sonucu Çizelge 4.8'de ve üç düğümlü yapılan sınamanın sonucu Çizelge 4.9'de gösterilmiştir.

Bu sınamada ortaya çıkan sonuç, Raspberry PI donanımı her ne kadar masaüstü bilgisayarının **%4**'ü kadar başarımları gösterse de, düğümlerin birbirinden bağımsız oluşu ve seri çalışmaya yönelik yegane kısıtın, uygulamanın ağ üzerinden düğüme aktarım işlemi gibi göz ardı edilebilecek bir işlem olması; düğümlerden elde edilen başarımları

**Çizelge 4.7:** Sınama 3 - Tek düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | R1    | 22.303 s  | 22.497 s    |
| 2      | M        | M         | R1    | 21.949 s  | 44.699 s    |
| 3      | M        | M         | R1    | 22.089 s  | 67.039 s    |
| 4      | M        | M         | R1    | 21.816 s  | 89.132 s    |
| 5      | M        | M         | R1    | 21.996 s  | 111.585 s   |
| 6      | M        | M         | R1    | 21.553 s  | 133.577 s   |
| 7      | M        | M         | R1    | 22.134 s  | 156.165 s   |
| 8      | M        | M         | R1    | 21.964 s  | 178.387 s   |
| 9      | M        | M         | R1    | 21.944 s  | 200.727 s   |
| 10     | M        | M         | R1    | 22.089 s  | 223.198 s   |

**Çizelge 4.8:** Sınama 3 - İki düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | R1    | 23.594 s  | 24.055 s    |
| 2      | M        | M         | R2    | 24.098 s  | 24.561 s    |
| 3      | M        | M         | R1    | 22.780 s  | 47.092 s    |
| 4      | M        | M         | R2    | 23.077 s  | 47.898 s    |
| 5      | M        | M         | R1    | 22.292 s  | 69.755 s    |
| 6      | M        | M         | R2    | 23.393 s  | 72.069 s    |
| 7      | M        | M         | R1    | 22.612 s  | 92.950 s    |
| 8      | M        | M         | R2    | 22.718 s  | 95.153 s    |
| 9      | M        | M         | R1    | 22.422 s  | 115.744 s   |
| 10     | M        | M         | R2    | 22.963 s  | 118.953 s   |

**Çizelge 4.9:** Sınama 3 - Üç düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | R3    | 21.860 s  | 22.501 s    |
| 2      | M        | M         | R1    | 22.053 s  | 22.702 s    |
| 3      | M        | M         | R2    | 22.266 s  | 22.880 s    |
| 4      | M        | M         | R3    | 21.280 s  | 44.021 s    |
| 5      | M        | M         | R1    | 21.470 s  | 44.414 s    |
| 6      | M        | M         | R2    | 21.500 s  | 44.611 s    |
| 7      | M        | M         | R3    | 21.124 s  | 65.575 s    |
| 8      | M        | M         | R1    | 21.163 s  | 66.023 s    |
| 9      | M        | M         | R2    | 21.365 s  | 66.406 s    |
| 10     | M        | M         | R3    | 21.434 s  | 87.363 s    |

toplamdaki başarıma etkisinin, düğümlerin sayısı ile doğru orantı göstermesini sağlamıştır.

Uygulamaların çalıştırılmaları esnasında, süre anlamında karşılaşılan farklılıklar;

- Sadece ilk tekrar için geçerli; düğümde çalıştırılacak uygulama dosyalarının transferi, **3 MB**'lık dosya yığını için yaklaşık **300 ms** sürmektedir.
- Donanımlar her ne kadar aynı yerden ve aynı donanım / yazılım özellikleriyle temin edilseler de fabrika ortamında üretimden kaynaklanan farklılıklardan dolayı birebir aynı başarıımı göstermemektedirler. Buradaki örnekte iki numaralı Raspberry PI donanımı, diğer donanımlardan daha yavaş çalışmakta, üç numaralı olan ise diğerlerinden daha hızlı çalışmaktadır.
- Donanımların açık olduğu süre boyunca ısılarının artışından kaynaklanan yan etkilerden dolayı hız farklılıkları oluşması da olasıdır.

Sınama sonucuna göre, ikinci düğümün eklenmesi sistemde **%92**, üçüncü düğümün eklenmesi ise **%199**'luk bir başarıım artışı sağlamıştır.

Sonuç olarak düğümlerin gösterdiği toplam başarıım, beklenene yakın olarak gerçekleşmektedir;

$$\text{Toplam Başarıım} \approx \text{Düğüm Sayısı} \times \text{Düğüm Başarıımı} \quad (4.2)$$

#### 4.3.4 Sınama 4

Yapılandırma :

Dağıtıcı -> Masaüstü Bilgisayar

Toplayıcılar -> Apple MacBook Pro Dizüstü Bilgisayar

Düğümler -> Raspberry PI Model B (1 - 3)

Üç düğümlü yapılan sınamanın sonucu Çizelge 4.10'de gösterilmiştir.

Sınama sonucuna göre, uygulamanın toplam bitiş süresi dağıtıcı ve toplayıcının sadece masaüstü bilgisayar olduğu duruma göre çok az daha uzun sürmüştür. İhmal edilebilecek bir süre de olsa bu fark, dizüstü bilgisayarın çalışma hızının masaüstüne göre biraz daha yavaş olmasına ve de toplayıcı ile dağıtıcı arasındaki bağlantının unix soketi yerine internet soketi ile sağlanmasına bağlanabilir.

**Çizelge 4.10:** Sınama 4 - Üç düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | D         | R2    | 22.240 s  | 22.376 s    |
| 2      | M        | D         | R3    | 21.843 s  | 23.225 s    |
| 3      | M        | D         | R1    | 22.094 s  | 24.737 s    |
| 4      | M        | D         | R2    | 21.206 s  | 43.826 s    |
| 5      | M        | D         | R3    | 21.305 s  | 44.775 s    |
| 6      | M        | D         | R1    | 21.160 s  | 46.137 s    |
| 7      | M        | D         | R2    | 21.661 s  | 65.904 s    |
| 8      | M        | D         | R3    | 21.432 s  | 66.642 s    |
| 9      | M        | D         | R1    | 21.119 s  | 67.990 s    |
| 10     | M        | D         | R2    | 21.726 s  | 88.109 s    |

#### 4.3.5 Sınama 5

Yapılandırma :

Dağıtıcı -> Masaüstü Bilgisayar

Toplayıcılar -> Masaüstü Bilgisayar

Düğüm -> Raspberry PI 2 Model B

Tek düğümlü yapılan sınamanın sonucu Çizelge 4.11’de gösterilmiştir.

**Çizelge 4.11:** Sınama 5 - Tek düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | P     | 3.864 s   | 3.891 s     |
| 2      | M        | M         | P     | 3.541 s   | 7.540 s     |
| 3      | M        | M         | P     | 3.591 s   | 11.234 s    |
| 4      | M        | M         | P     | 3.519 s   | 14.857 s    |
| 5      | M        | M         | P     | 3.579 s   | 18.548 s    |
| 6      | M        | M         | P     | 3.837 s   | 22.490 s    |
| 7      | M        | M         | P     | 4.080 s   | 26.673 s    |
| 8      | M        | M         | P     | 4.104 s   | 31.247 s    |
| 9      | M        | M         | P     | 3.686 s   | 35.047 s    |
| 10     | M        | M         | P     | 4.310 s   | 39.476 s    |

Raspberry PI 2’nin başarımı, sınanan ilk versiyonuna göre yaklaşık **%565** daha iyi olsa da, masaüstü bilgisayarın başarımının yaklaşık **%21**’i kadar bir başarıma karşılık gelmektedir.

#### 4.3.6 Sınama 6

Yapılandırma :

Dağıtıcı -> Masaüstü Bilgisayar

Toplayıcılar -> Masaüstü Bilgisayar

Düğüm -> BeagleBone Black Model C

Tek düğümlü yapılan sınavın sonucu Çizelge 4.12’de gösterilmiştir.

**Çizelge 4.12:** Sınama 6 - Tek düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | B     | 7.461 s   | 8.023 s     |
| 2      | M        | M         | B     | 7.179 s   | 15.343 s    |
| 3      | M        | M         | B     | 7.131 s   | 22.615 s    |
| 4      | M        | M         | B     | 7.240 s   | 29.997 s    |
| 5      | M        | M         | B     | 7.203 s   | 37.343 s    |
| 6      | M        | M         | B     | 7.237 s   | 44.721 s    |
| 7      | M        | M         | B     | 7.173 s   | 52.037 s    |
| 8      | M        | M         | B     | 7.236 s   | 59.416 s    |
| 9      | M        | M         | B     | 7.172 s   | 66.730 s    |
| 10     | M        | M         | B     | 7.243 s   | 74.115 s    |

BeagleBone Black, giriş / çıkış özellikleri anlamında sınavan donanımlar arasında en zengin özellik yelpazesine sahip olsa da, işlemcisinin tek çekirdekli olması, beklenen başarımları verememesine neden olmuştur. İlk versiyon Raspberry PI’ye oranla **%300** daha iyi başarımlar sergilemiş olsa da, ikinci versiyon Raspberry PI’nin **%53**’ü ve masaüstü bilgisayarın **%11**’i kadar başarımlar sergileyebilmiştir.

#### 4.3.7 Sınama 7

Yapılandırma :

Dağıtıcı -> Masaüstü Bilgisayar

Toplayıcılar -> Masaüstü Bilgisayar

Düğüm -> HardKernel ODroid U3

Tek düğümlü yapılan sınavın sonucu Çizelge 4.13’de gösterilmiştir.

**Çizelge 4.13:** Sınama 7 - Tek düğümlü

| Tekrar | Dağıtıcı | Toplayıcı | Düğüm | İş Süresi | Toplam Süre |
|--------|----------|-----------|-------|-----------|-------------|
| 1      | M        | M         | O     | 1.878 s   | 1.904 s     |
| 2      | M        | M         | O     | 1.631 s   | 3.590 s     |
| 3      | M        | M         | O     | 1.574 s   | 5.219 s     |
| 4      | M        | M         | O     | 1.711 s   | 6.996 s     |
| 5      | M        | M         | O     | 1.682 s   | 8.731 s     |
| 6      | M        | M         | O     | 1.634 s   | 10.417 s    |
| 7      | M        | M         | O     | 1.742 s   | 12.216 s    |
| 8      | M        | M         | O     | 1.750 s   | 14.022 s    |
| 9      | M        | M         | O     | 1.699 s   | 15.777 s    |
| 10     | M        | M         | O     | 1.720 s   | 17.558 s    |

HardKernel ODroid U3, sınanan donanımlar arasında en başarılı donanım olarak ön plana çıkmıştır. En yakın rakibi Raspberry PI 2'den yaklaşık **%224** daha iyi başarımla sergilemiş, buna karşın masaüstü bilgisayarın **%48**'i kadar bir başarımda kalmıştır.

#### 4.4 Maliyet Analizi

Yapılan sınamalar sonucunda ortaya çıkan başarımla değerleri ile sınamaya tabi tutulan bilgisayarlar ve atanmış bilgisayar donanımlarının başarımla / fiyat dengeleri karşılaştırılmıştır. Sınamalarda referans alınan donanım masaüstü bilgisayar olmuş, diğer bütün donanımlar masaüstü bilgisayarına oranla karşılaştırmadaki yerlerini almışlardır.

Masaüstü bilgisayar, günümüz piyasasında yapılan fiyat araştırmasında; işlemci ve belleğin sınamaya giren değerlerde kalması koşuluyla, diğer zorunlu donanım bileşenleri en az fiyatlı bileşenlerden seçilmiştir. Buna göre seçilen donanıma ilişkin bileşen Çizelge 4.14'de gösterilmiştir.

**Çizelge 4.14:** Masaüstü bilgisayar donanım bileşenleri

|            | Bileşen                                    | Fiyat   |
|------------|--------------------------------------------|---------|
| İşlemci    | Intel Core i7-3770 3.40 Ghz, Dört Çekirdek | 1051 TL |
| Bellek     | Corsair Vengeance 16 GB (2x8 GB)           | 547 TL  |
| Anakart    | Gigabyte GA-B75M-D2V                       | 226 TL  |
| Sabit Disk | Seagate 1TB 7200RPM Sata 3                 | 189 TL  |
| Kasa       | Boost 400W ATX                             | 155 TL  |
| Toplam     |                                            | 2168 TL |

Sınanan donanımlara ilişkin bileşen ve fiyat bilgisi ise Çizelge 4.15'de gösterilmiştir.

**Çizelge 4.15:** Diğer donanım bileşenleri

| Donanım                  | İşlemci                                     | Bellek       | Depolama     | Fiyat   |
|--------------------------|---------------------------------------------|--------------|--------------|---------|
| Apple MacBook Pro        | Intel Core i7 Quad 2.0 Ghz                  | 8 GB DDR3    | 256 GB SSD   | 4699 TL |
| Raspberry PI Model B     | BCM2835 ARM11 Single 700 Mhz                | 512 MB SDRAM | 8 GB mikroSD | 125 TL  |
| Raspberry PI 2 Model B   | BCM2836 ARM Cortex-A7 Quad 900 Mhz          | 1 GB DDR2    | 8 GB mikroSD | 125 TL  |
| BeagleBone Black Model C | Sitara AM335x ARM Cortex-A8 Single 1 Ghz    | 512 MB DDR3  | 4 GB eMMC    | 139 TL  |
| HardKernel ODroid U3     | Exynos4412 Prime ARM Cortex-A9 Quad 1.7 Ghz | 2 GB DDR2    | 8 GB mikroSD | 230 TL  |

Sınanan donanımlara ilişkin başarımlar ve fiyat dengesi uyarınca çıkarılan maliyet indeksi aşağıdaki gibi ifade edilmiştir;

**DA** : Gerekli en az donanım adedi

**MP** : Masaüstü bilgisayar başarımı

**DP** : Donanım başarımı

$$DA = \left\lceil \frac{MP}{DP} + 0.5 \right\rceil \quad (4.3)$$

**GY** : Havuzda yedekte tutulacak donanımın yüzdesi (Güvenilirlik yüzdesi)

**TDA** : Toplam gerekli donanım adedi

$$TDA = DA + \lceil DA \times GY \rceil \quad (4.4)$$

**Mİ** : Maliyet indeksi

**MF** : Masaüstü bilgisayar fiyatı

**DF** : Donanım fiyatı

$$Mİ = \frac{MF}{DF \times TDA} \geq 1 \quad (4.5)$$

Maliyet İndeksi ne kadar yüksek olursa, seçilen donanımın başarımlar / fiyat oranı o kadar iyi olmakta, **1**'in üzerinde olması ise, seçilen donanımla oluşturulacak sistemin masaüstü bilgisayardan oluşan sisteme göre daha iyi başarımlar / fiyat değerine sahip olacağını göstermektedir.

Belirtilen denklemlere göre oluşturulan maliyet indeksi değerleri;

Güvenilirlik Yüzdesi (GY) = %0 için Çizelge 4.16'de gösterilmiştir.

**Çizelge 4.16:** Maliyet indeksi, GY = %0

| Donanım                  | MP / DP | GY  | TDA | Mİ    |
|--------------------------|---------|-----|-----|-------|
| Apple MacBook Pro        | 1.012   | 0.0 | 1   | 0.461 |
| Raspberry PI Model B     | 26.514  | 0.0 | 27  | 0.642 |
| Raspberry PI 2 Model B   | 4.689   | 0.0 | 5   | 3.469 |
| BeagleBone Black Model C | 8.804   | 0.0 | 9   | 1.733 |
| HardKernel ODroid U3     | 2.086   | 0.0 | 2   | 4.713 |

Güvenilirlik Yüzdesi (GY) = %10 için Çizelge 4.17’de gösterilmiştir.

**Çizelge 4.17:** Maliyet indeksi, GY = %10

| Donanım                  | MP / DP | GY  | TDA | Mİ    |
|--------------------------|---------|-----|-----|-------|
| Apple MacBook Pro        | 1.012   | 0.1 | 2   | 0.231 |
| Raspberry PI Model B     | 26.514  | 0.1 | 30  | 0.578 |
| Raspberry PI 2 Model B   | 4.689   | 0.1 | 6   | 2.891 |
| BeagleBone Black Model C | 8.804   | 0.1 | 10  | 1.560 |
| HardKernel ODroid U3     | 2.086   | 0.1 | 3   | 3.142 |

Güvenilirlik Yüzdesi (GY) = %20 için Çizelge 4.18’de gösterilmiştir.

**Çizelge 4.18:** Maliyet indeksi, GY = %20

| Donanım                  | MP / DP | GY  | TDA | Mİ    |
|--------------------------|---------|-----|-----|-------|
| Apple MacBook Pro        | 1.012   | 0.2 | 2   | 0.231 |
| Raspberry PI Model B     | 26.514  | 0.2 | 33  | 0.526 |
| Raspberry PI 2 Model B   | 4.689   | 0.2 | 6   | 2.891 |
| BeagleBone Black Model C | 8.804   | 0.2 | 11  | 1.418 |
| HardKernel ODroid U3     | 2.086   | 0.2 | 3   | 3.142 |

Güvenilirlik Yüzdesi (GY) = %30 için Çizelge 4.19’de gösterilmiştir.

**Çizelge 4.19:** Maliyet indeksi, GY = %30

| Donanım                  | MP / DP | GY  | TDA | Mİ    |
|--------------------------|---------|-----|-----|-------|
| Apple MacBook Pro        | 1.012   | 0.3 | 2   | 0.231 |
| Raspberry PI Model B     | 26.514  | 0.3 | 36  | 0.482 |
| Raspberry PI 2 Model B   | 4.689   | 0.3 | 7   | 2.478 |
| BeagleBone Black Model C | 8.804   | 0.3 | 12  | 1.300 |
| HardKernel ODroid U3     | 2.086   | 0.3 | 3   | 3.142 |

Güvenilirlik Yüzdesi (GY) = %50 için Çizelge 4.20’de gösterilmiştir.

**Çizelge 4.20:** Maliyet indeksi, GY = %50

| Donanım                  | MP / DP | GY  | TDA | Mİ    |
|--------------------------|---------|-----|-----|-------|
| Apple MacBook Pro        | 1.012   | 0.5 | 2   | 0.231 |
| Raspberry PI Model B     | 26.514  | 0.5 | 41  | 0.423 |
| Raspberry PI 2 Model B   | 4.689   | 0.5 | 8   | 2.168 |
| BeagleBone Black Model C | 8.804   | 0.5 | 14  | 1.114 |
| HardKernel ODroid U3     | 2.086   | 0.5 | 3   | 3.142 |

Güvenilirlik Yüzdesi (GY) = %80 için Çizelge 4.21’de gösterilmiştir.

Güvenilirlik Yüzdesi (GY) = %100 (Her donanım için havuzda bekleyen bir yedek)  
için Çizelge 4.22’de gösterilmiştir.

**Çizelge 4.21:** Maliyet indeksi, GY = %80

| Donanım                  | MP / DP | GY  | TDA | Mİ    |
|--------------------------|---------|-----|-----|-------|
| Apple MacBook Pro        | 1.012   | 0.8 | 2   | 0.231 |
| Raspberry PI Model B     | 26.514  | 0.8 | 49  | 0.354 |
| Raspberry PI 2 Model B   | 4.689   | 0.8 | 9   | 1.927 |
| BeagleBone Black Model C | 8.804   | 0.8 | 17  | 0.917 |
| HardKernel ODroid U3     | 2.086   | 0.8 | 4   | 2.357 |

**Çizelge 4.22:** Maliyet indeksi, GY = %100

| Donanım                  | MP / DP | GY  | TDA | Mİ    |
|--------------------------|---------|-----|-----|-------|
| Apple MacBook Pro        | 1.012   | 1.0 | 2   | 0.231 |
| Raspberry PI Model B     | 26.514  | 1.0 | 54  | 0.321 |
| Raspberry PI 2 Model B   | 4.689   | 1.0 | 10  | 1.734 |
| BeagleBone Black Model C | 8.804   | 1.0 | 18  | 0.867 |
| HardKernel ODroid U3     | 2.086   | 1.0 | 4   | 2.357 |

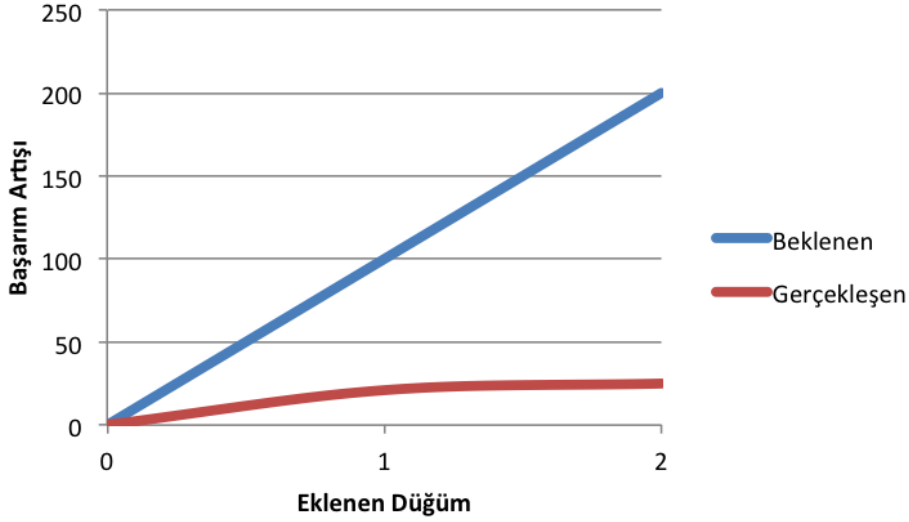
#### 4.5 Sonuç

Temin edilen dört farklı atanmış bilgisayar donanımı, bir masaüstü bilgisayar ve bir dizüstü bilgisayar ile yapılan sınamalar sonucunda ulaşılan sonuçlar şu şekildedir;

1. Uygulamanın tek bir bilgisayarda çalıştırılması durumunda, bilgisayarın işlemci başarımı ne kadar iyi olursa olsun sınırlı olduğundan, özellikle çok sayıda paralel ve birbirinden bağımsız alt görevlere sahip uygulamalar işlemciyi paylaşımlı olarak kullanmak zorundadırlar. O nedenle bu tarz uygulamalar daha hızlı çalıştırılma imkanına sahip olsa da işlemci bariyerine takılmaktadırlar. Sistemi genel anlamda engelleyen bir diğer değişirge de uygulamaların kullandığı depolama alanı ve ağ arabirimlerinin tekil oluşudur. Yani uygulamalar depo alanına ve ağ arabirimlerine sıralı erişebilmektedirler, bu da uygulamaların paralel çalışma imkanlarını kısıtlamaktadır.

Masaüstü bilgisayarın tek başına bütün rolleri üstlendiği sınama sonucunda, tek düğümlü sisteme ikinci bir düğüm eklendiğinde iki katı bir başarımlı beklenirken bu başarımlı %21’de kalmıştır. Yine benzer şekilde sisteme üçüncü düğüm eklendiğinde başarımlı artışı %200 yerine %25’de kalarak iki düğümlü de görülen artışın da altında kalmıştır. Başarımlı artışına ilişkin grafik Şekil 4.1’de gösterilmiştir.

2. Masaüstü ya da dizüstü bilgisayarlar, üzerlerindeki ek iş yükleri nedeniyle bir uygulamanın yüksek başarımlı olarak çalıştırılması amaçlandığında, sahip oldukları



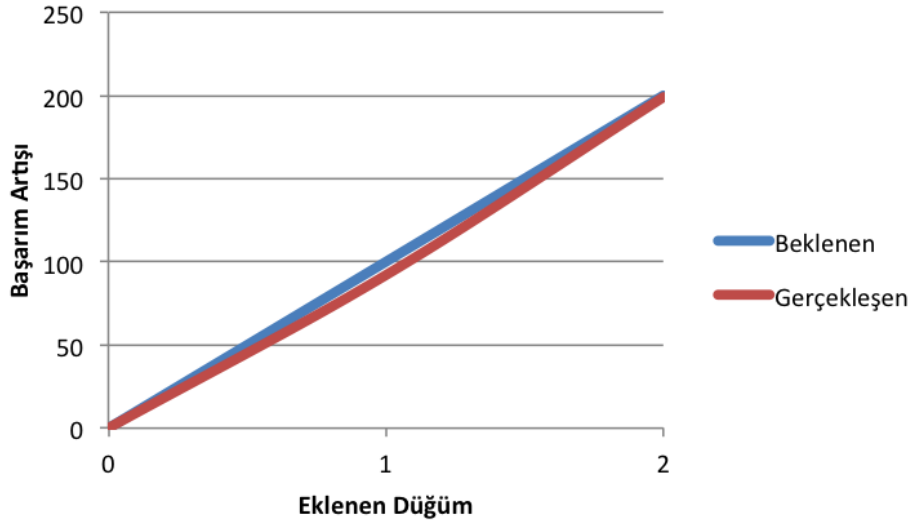
**Şekil 4.1:** Tek bilgisayar - Başarım artışı.

donanım bileşenlerinin kabiliyetlerinin gerisinde başarımlarını göstermektedirler. Dolayısıyla yüksek başarımlar gerektiren uygulamalar için belirli başarımların sağlanması noktasında ek maliyet getirmektedirler.

Masaüstü ve dizüstü bilgisayarlar arası yapılan başarımların sonucunda, masaüstü bilgisayarın çok daha üstün nitelikte donanıma sahip olmasına rağmen dizüstü bilgisayara yakın başarımlar göstermesi, harici uygulamaların sistemi beklenmedik oranlarda yavaşlatabileceğini göstermektedir.

3. Düşümlerde çalışacak uygulama bileşenlerinin boyutu yaklaşık **3 MB** olup, yapılan sınamalarda bunların toplayıcılardan düşümlere aktarımı, donanımın cinsinden bağımsız sabit olarak yaklaşık **300 ms**'de gerçekleşmiştir. Orantısız olarak en çok etkilenen donanım, uygulamanın çalıştırılmasını en hızlı gerçekleştiren donanım olduğu için, bu donanımdaki aktarımın toplamdaki başarıma etkisi **%1.7**'de kalmıştır. Sonuç olarak seçilen uygulama için düşümlere aktarım işlemi, başarımların değerlendirilmesinde ihmal edilebilecek bir düzeydedir.
4. Uygulamaların paralel alt görevlerinin birbirlerinden bağımsız olarak atanmış bilgisayar donanımlarında çalışabiliyor olması, her alt görevin herhangi bir bekleme maruz kalmadan olabilecek en yüksek başarımda tamamlanmasına imkan tanımaktadır. Dolayısıyla, donanımların işlemci gücü ve diğer bileşenlerinden en optimum oranda yararlanılabilmektedir. Bu durumda donanım bileşenlerinin özellikleri uygulamaların çalışma hızına doğrudan etki edecektir.

Üç adet Raspberry PI donanımıyla yapılan sına göstermiştir ki, sistemden elde edilen toplam başarımlar, donanımların teker teker elde ettikleri başarımların toplamına yakındır. Tek düğümlü sisteme ikinci bir düğüm eklendiğinde başarımların artışı **%92** olmakta, üçüncü düğümün eklenmesiyle beraber artış **%199**'u bulmaktadır. Başarımların artışına ilişkin grafik Şekil 4.2'de gösterilmiştir.



**Şekil 4.2:** Üç Raspberry PI - Başarımların artışı.

5. Toplayıcı ve dağıtıcı bileşenlerinin farklı bilgisayarlarda çalıştırılmasının olumlu ve olumsuz yönleri incelendiğinde, olumlu yönü işlemci ve diğer bileşenlerin paylaşımından kaynaklı bir bekleme durumunun olmamasına karşın, olumsuz yönü toplayıcı - dağıtıcı arası haberleşmenin bilgisayar içi çok hızlı unix soketleri yerine, ağ üzerinden internet soketleri vasıtasıyla yapılmasıdır.

Sınama sonuçları göstermiştir ki, hem toplayıcı ve dağıtıcının sistem genelindeki iş yüklerinin, düğümlere oranla çok az olması hem de aktarılacak verinin yapılan çalışma genelinde çok düşük olması, toplayıcı ve dağıtıcı bileşenlerinin aynı ya da farklı bilgisayarlarda çalıştırılmalarının sistem genelinde fazla bir etkisinin olmadığını göstermektedir. Ölçülen zamanlara göre farklı bilgisayarlarda çalıştırılmaları sadece **%1**'lik bir fark yaratmış, bu fark bahsedilen etkilerin yanısıra bilgisayarların yapılandırma farkından da kaynaklanabilmektedir.

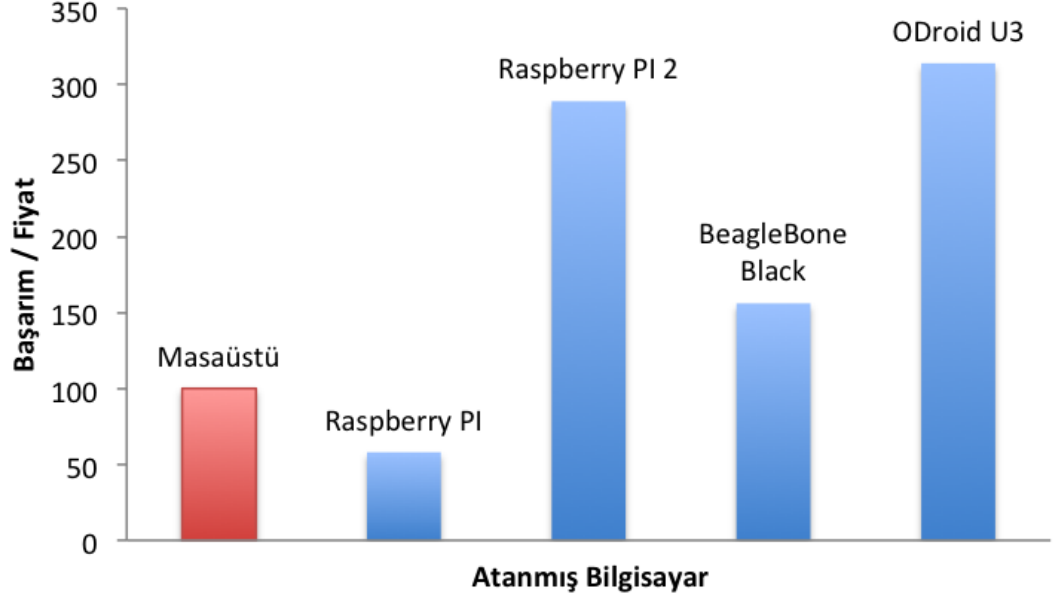
#### 4.5.1 Atanmış bilgisayar başarımları

Sınamaların sınırlı sayıda atanmış bilgisayarla yapılmasından dolayı birbirleri arasındaki karşılaştırmalar tek düğümlü olarak yapılmış ve sonuçları irdelenmiştir;

1. **Raspberry PI Model B** : İşlemci gücü itibarıyla diğer donanımların biraz gerisinde kalan Raspberry PI, tek düğümlü sınamada, sınamayı yaklaşık **223** s'de bitirerek, masaüstü bilgisayarın **%4**'üne karşılık gelen bir başarımlı sergilemiştir. Yapılan maliyet analizine göre, **%10** güvenilirlikli çalışma kipinde **30** adet donanım gereksinimiyle masaüstü bilgisayarın **%58**'ine karşılık gelen bir başarımlı / fiyat oranı sergilemiştir. Gösterdiği başarımlı ile masaüstü sistemine bir alternatif olamamıştır.
2. **Raspberry PI 2 Model B** : Önceki modele göre altı kat başarımlı vaat eden Raspberry PI 2, tek düğümlü sınamada, sınamayı yaklaşık **39** s'de bitirerek, masaüstü bilgisayarın **%21**'ine karşılık gelen bir başarımlı sergilemiştir. Yapılan maliyet analizine göre, **%10** güvenilirlikli çalışma kipinde **6** adet donanım gereksinimiyle masaüstü bilgisayarın **%289**'una karşılık gelen bir başarımlı / fiyat oranı sergilemiştir. Gösterdiği başarımlı ile masaüstü sistemine iyi bir alternatif olmuştur.
3. **BeagleBone Black Model C** : Genişleme yetenekleriyle diğer donanımların önüne geçen BeagleBone Black Model C, tek düğümlü sınamada, sınamayı yaklaşık **74** s'de bitirerek, masaüstü bilgisayarın **%11**'ine karşılık gelen bir başarımlı sergilemiştir. Yapılan maliyet analizine göre, **%10** güvenilirlikli çalışma kipinde **10** adet donanım gereksinimiyle masaüstü bilgisayarın **%156**'sına karşılık gelen bir başarımlı / fiyat oranı sergilemiştir. Gösterdiği başarımlı ile masaüstü sistemine iyi bir alternatif olmuştur.
4. **HardKernel ODroid U3** : İşlemci gücü itibarıyla diğer donanımların önüne geçen ODroid U3, tek düğümlü sınamada, sınamayı yaklaşık **18** s'de bitirerek, masaüstü bilgisayarın **%48**'ine karşılık gelen bir başarımlı sergilemiştir. Yapılan maliyet analizine göre, **%10** güvenilirlikli çalışma kipinde **3** adet donanım gereksinimiyle masaüstü bilgisayarın **%314**'üne karşılık gelen bir başarımlı / fiyat

oranı sergilemiştir. Gösterdiği başarımla masaüstü sistemine en iyi alternatif olmuştur.

Atanmış bilgisayarların masaüstü bilgisayarına oranla başarımlarına ilişkin grafik Şekil 4.3’de gösterilmiştir.



**Şekil 4.3:** Atanmış bilgisayar başarımları / fiyat grafiği

#### 4.6 Gelecekteki Çalışmalar

Tez kapsamında yapılan çalışma başarımla tabanlı olduğu için, tasarlanan yazılım, seçilen uygulama ve yapılan sınamalar başarımla en üst noktada tutma eğilimiyle gerçekleşmiştir. Sınanacak uygulamanın matematiksel işlem tabanlı bir uygulama seçilmesi, sınamaların sadece işlemci gücü çerçevesinde değerlendirilmesini yeterli kılmıştır. Fakat matematiksel uygulama harici, farklı özellikteki uygulamalarda işlemci haricinde bellek, depolama alanı hızı ve büyüklüğü, ağ transfer hızı gibi diğer bileşenlerin de önem kazanması olasıdır. Bu durumlarda yapılan sınamaların sonuçları ve ortaya çıkan başarımları / fiyat oranları bambaşka sonuçlar verebilir. Dolayısıyla bu çalışmanın sadece işlemci gücü bazlı değerlendirilmesi, farklı uygulamalar için sınamaların yenilenmesi ve gerekirse temin edilen donanımların farklı yapılandırılmış başka modellerle değiştirilmesi gerekmektedir.

Çalışmanın başarımlı tabanlı olmasının diğerk bir sonucu olarak, platform bağımsızlığı çalışma kapsamında devre dışı bırakılmıştır. Platform bağımsızlığının sağlanamaması, hem geliştirilen merkez yönetim yazılımının düğüm tarafında çalışan kısmının, hem de sınanacak uygulamanın, düğüm görevi gören atanmış bilgisayarların sahip olduğu işlemci mimarisine uygun olarak derlenmesini mecbur kılmıştır. Derleme sürecinin her farklı donanım için tekrarlanması tasarlanan sistemin önünde aşılması gereken en önemli sorundur. Yapılan literatür araştırmalarında, haberleşmede uzak yordam çağırısı ya da MPI gibi mesaj aktarım yöntemlerinin ve uygulama geliştirilmesinde Java gibi programlama dillerinin tercih edilmesi platform bağımsızlığını sağladığı için bu kapsamda merkez yönetim yazılımı yeniden tasarlanıp sınamalar yenilenmelidir. Buna karşılık hem tercih edilen haberleşme yöntemlerinin ek kütüphane ve altyapı gereksinimleri hem de platform bağımsız programlama dillerinin gerektirdiğı sanal makine gereksinimleri, atanmış bilgisayar donanımları üzerinde ek yük oluşturacağı için, çalışma bu kapsamda değerlendirilip ortaya çıkacak başarımlı / fiyat değerlerinin istenen düzeyde olup olmadığının değerlendirilmesi gerekmektedir.

Yapılan çalışmayı diğerk benzerlerinden ayıran yönü, eklediğı güvenilirlik ve verim özelliklerinin uzun vadede getirdiğı üstünlük ve eksikliklerinin sınanması noktasındadır. Mevcut durumda yapılan sınamalar olumlu sonuç verse de uzun vadedeki sonuçlarını görmek için gerçek hayat uygulamalarında sınanıp sonuçlarının görülmesi gerekmektedir.

## KAYNAKLAR

- [1] **Chandrakasan, A.P.** ve diğeri (1995). Optimizing Power Using Transformations, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **14**(1), 12–31.
- [2] **Yiqun, C.**, (2005). Parallel And Distributed Computing Techniques In Biomedical Engineering, *Yüksek Lisans Tezi*, National University Of Singapore.
- [3] **Oberhumer, T.**, Lempel–Ziv–Oberhumer Lossless Data Compression Algorithm, <http://www.oberhumer.com/opensource/lzo/>, alındığı tarih: 18.03.2015.
- [4] **Olson, E.**, (2002). Cluster Juggler – PC cluster virtual reality, *Yüksek Lisans Tezi*, Iowa State University.
- [5] **Bahcecioglu, T.**, (2011). Parallel Solution Of Soil-Structure Interaction Problems On Pc Clusters, *Yüksek Lisans Tezi*, Middle East Technical University.
- [6] **Kiepert, J.** (2013). RPiCLUSTER - Creating a Raspberry Pi-Based Beowulf Cluster, *Teknik Rapor*, Boise State University.
- [7] **Beane, G.**, (2002). The Effects Of Microprocessor Architecture On Speedup In Distributed Memory Supercomputers, *Yüksek Lisans Tezi*, The University of Maine.
- [8] Craft CFD, <http://www.craft-tech.com/>, alındığı tarih: 21.03.2015.
- [9] **Huang, B.**, (2004). Network Performance Studies in High Performance Computing Environments, *Yüksek Lisans Tezi*, The University of Western Ontario.
- [10] **Bennett, T.**, (2003). Development of a Parallel SAR Processor on a Beowulf Cluster, *Yüksek Lisans Tezi*, The University of Cape Town.
- [11] **Nielson, C.**, (2003). A Descriptive Performance Model of Small, Low Cost, Diskless Beowulf Clusters, *Yüksek Lisans Tezi*, Brigham Young University.
- [12] **Adali, E.** (1980). *Dağıtılmış Bilgisayarlarla Denetim Doçentlik Tezi*, İSTANBUL, TR.
- [13] **Steusloff, H.** (1976). Hardware Structures and Software for Distributed Computer Control Systems, *The 3rd Annual Control Conference*, 101, Purdue, Indiana, USA.
- [14] **Rivest, R.** (1992). *The MD5 Message-Digest Algorithm*, RFC Editor, United States.

- [15] **Amdahl, G.** (1967). Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), ACM, New York, NY, USA, s.483–485.
- [16] **Gustafson, J.** (1988). Reevaluating Amdahl's Law, *Communications of the ACM*, **31**, 532–533.
- [17] **Meek, C.**, (2000). Mainframe, *Encyclopedia of Computer Science*, John Wiley and Sons Ltd., Chichester, UK, s.1068–1068.
- [18] **Berson, A.** (1992). *Client/Server Architecture*, McGraw-Hill, Inc., New York, NY, USA.
- [19] **Ishikawa, Y. ve Tokoro, M.** (1984). The Design of an Object Oriented Architecture, *SIGARCH Comput. Archit. News*, **12**(3), 178–187.
- [20] **Vinoski, S.** (2000). Introduction to CORBA (Tutorial Session), *Proceedings of the 22Nd International Conference on Software Engineering*, ICSE '00, ACM, New York, NY, USA, s.822–.
- [21] **Redmond, F.** (1997). *Dcom: Microsoft Distributed Component Object Model*, IDG Books Worldwide, Inc., Foster City, CA, USA, 1st sürüm.
- [22] **Pfister, G.**, (2000). Cluster Computing, *Encyclopedia of Computer Science*, John Wiley and Sons Ltd., Chichester, UK, s.218–221.
- [23] **Meredith, M. ve diğerleri** (2003). Exploring Beowulf Clusters, *J. Comput. Sci. Coll.*, **18**(4), 268–284.
- [24] **Foster, I. ve diğerleri** (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *Int. J. High Perform. Comput. Appl.*, **15**(3), 200–222.
- [25] **Silcock, J. ve Goscinski, A.** (1995). Message Passing, Remote Procedure Calls and Distributed Shared Memory as Communication Paradigms for Distributed Systems, **Teknik Rapor**, Deakin University.
- [26] **Goscinski, A.** (1991). *Distributed Operating Systems: The Logical Design*, Addison-Wesley, Boston, MA, USA, 1. sürüm.
- [27] **Tanenbaum, A. ve Renesse, R.** (1985). Distributed operating systems, *ACM Computing Surveys*, **17**(4), 419–470.
- [28] **Stevens, W. ve diğerleri** (2003). *Unix Network Programming, Volume 1: The Sockets Networking API*, Addison-Wesley, 3. sürüm.
- [29] **Gropp, W. ve diğerleri**, (2014). Using MPI: Portable Parallel Programming with the Message-Passing Interface, MIT Press, Cambridge, USA, 3. sürüm.
- [30] **Geist, G.A. ve diğerleri** (1990). PICL: A portable instrumented communications library, C reference manual., **Teknik Rapor**, Oak Ridge National Laboratory.

- [31] **Beguelin, A. ve diğeri** (1991). A user's guide to PVM: Parallel virtual machine., **Teknik Rapor**, Oak Ridge National Laboratory.
- [32] **Mullender, S.** (1989). *Distributed Systems*, acm Press.
- [33] **Birrell, A. ve Nelson, B.** (1984). Implementing Remote Procedure Calls, *ACM Transactions on Computer Systems*, **2**(1), 39–59.
- [34] **Coulouris, G.F. ve diğeri** (2011). *Distributed Systems Concepts and Design*, Addison-Wesley, 5. sürüm.
- [35] **Singhal, M. ve Shivaratri, N.** (1994). *Advanced Concepts in Operating Systems*, McGraw-Hill, Inc., New York, NY, USA.
- [36] **Nitzberg, B. ve Lo, V.** (1991). Distributed Shared Memory: A Survey of Issues and Algorithms, *Computer - Distributed computing systems: separate resources acting as one*, **24**(8), 52 – 60.
- [37] **Tanenbaum, A.** (1995). *Distributed Operating Systems.*, Prentice Hall.
- [38] **Mosberger, D.** (1993). Memory Consistency Models., **Teknik Rapor**, Dept. of Computer Science, Univ. of Arizona.
- [39] OSCAR - Open Source Cluster Application Resources, <http://svn.oscar.openclustergroup.org/trac/oscar>, alındığı tarih: 29.03.2015.
- [40] ROCKS - Open Source Toolkit for Real and Virtual Clusters, <http://www.rocksclusters.org/>, alındığı tarih: 29.03.2015.
- [41] WareWulf - Scalable, Modular, Adaptable Systems Management, <http://warewulf.lbl.gov/trac>, alındığı tarih: 29.03.2015.
- [42] xCAT - An Extreme Cluster / Cloud Administration Toolkit, [http://sourceforge.net/p/xcat/wiki/Main\\_Page/](http://sourceforge.net/p/xcat/wiki/Main_Page/), alındığı tarih: 29.03.2015.
- [43] DCC - Debian Cluster Components, <http://www.irb.hr/eng/Research/Divisions-and-Centers/Centre-for-Informatics-and-Computing/Projects/CIR-Internal-projects/Debian-Cluster-Components-DCC>, alındığı tarih: 29.03.2015.
- [44] Raspberry PI, <http://www.raspberrypi.org/>, alındığı tarih: 29.03.2015.
- [45] BeagleBone, <http://beagleboard.org/bone>, alındığı tarih: 29.03.2015.
- [46] ODROID, <http://www.hardkernel.com/>, alındığı tarih: 29.03.2015.
- [47] **ECMA** (2013). *The JSON Data Interchange Format*, ECMA-404, Geneva, Switzerland.
- [48] **Smart, J., Hock, K. ve Csomor, S.** (2005). *Cross-Platform GUI Programming with wxWidgets (Bruce Perens Open Source)*, Prentice Hall PTR, Upper Saddle River, NJ, USA.



## **EKLER**

### **EK A.1 : Donanımlar**



## **EK A.1**

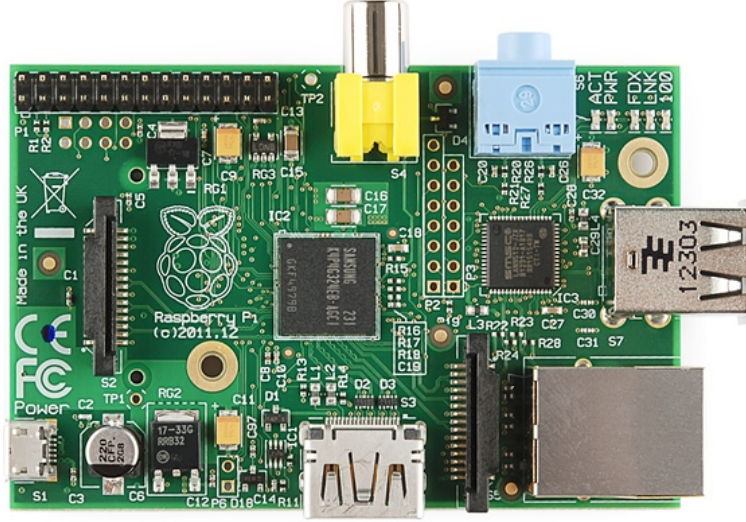
### **1.1 Raspberry PI Model B**

- **İşlemci** : Broadcom BCM2835  
ARM11 v6 mimari Tek Çekirdekli  
700 Mhz  
Kayan nokta desteği var
- **Grafik** : Videocore 4 GPU
- **Bellek** : 512 MB SDRAM
- **Çevre Birimleri** :  
SD Kart girişi  
2 USB portu  
Ethernet portu  
3.5mm jacklı ses çıkış portu  
HDMI çıkışı  
Composite video çıkışı
- **Giriş / Çıkış** :  
8 GPIO pini  
1 UART arabirimi  
1 SPI hattı  
1 I2C hattı
- **Fiyat** : 35\$

Üretici firmanın sağladığı sınama sonuçlarına göre pratikteki başarıımı 300 Mhz Pentium 2 işlemci ile eşdeğerdir.

### **1.2 Raspberry PI 2 Model B**

- **İşlemci** : Broadcom BCM2836  
ARM Cortex A7 Dört Çekirdekli  
900 Mhz  
NEON Kayan nokta desteği var
- **Grafik** : Videocore 4 GPU

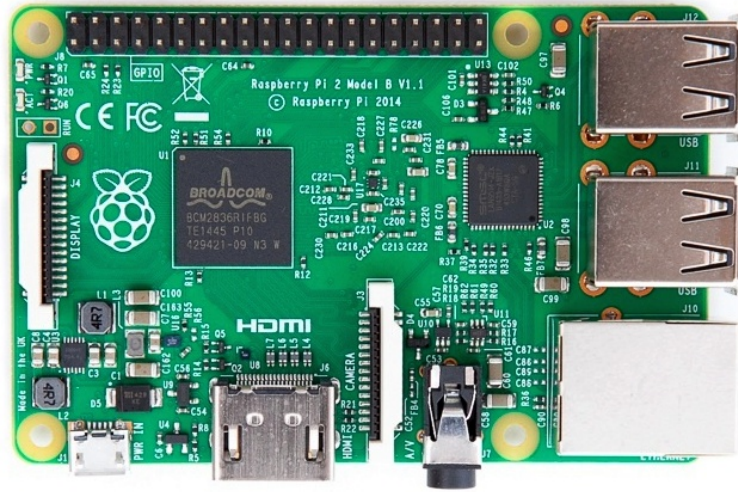


Şekil A.1: Raspberry PI Model B.

- **Bellek** : 1 GB DDR2
- **Çevre Birimleri** :
  - SD Kart girişi
  - 4 USB portu
  - Ethernet portu
  - 3.5mm jacklı ses çıkış portu
  - HDMI çıkışı
- **Giriş / Çıkış** :
  - 17 GPIO pini
  - 1 UART arabirimi
  - 1 SPI hattı
  - 1 I2C hattı
- **Fiyat** : 35\$

### 1.3 BeagleBone Black Model C

- **İşlemci** : Sitara AM335x
  - ARM Cortex A8 Tek Çekirdekli
  - 1 Ghz
  - NEON kayan nokta desteği var
- **Grafik** : PowerVR SGX
- **Bellek** : 512 MB DDR3

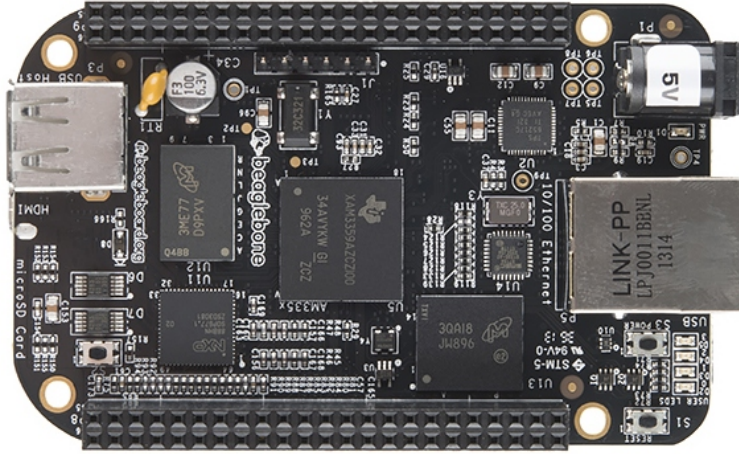


Şekil A.2: Raspberry PI 2 Model B.

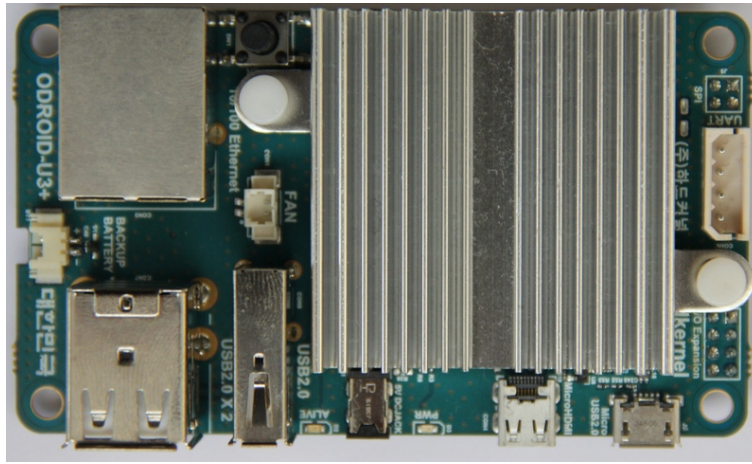
- **Çevre Birimleri :**
  - 4 GB onboard eMMC
  - SD Kart girişi
  - 2 USB portu
  - Ethernet portu
  - HDMI çıkışı (ses destekli)
- **Giriş / Çıkış :**
  - 3 I2C hattı
  - CAN hattı
  - SPI hattı
  - 4 zamanlayıcı
  - 5 seri port
  - 65 GPIO pini
  - 8 PWM çıkışı
  - 7 analog giriş (1.8V max 12 bit A/D dönüştürücü)
- **Fiyat : 45\$**

#### 1.4 HardKernel ODROID U3

- **İşlemci :** Exynos4412 Prime
  - ARM Cortex A9 Dört Çekirdekli
  - 1.7 Ghz
  - NEON kayan nokta desteği var
- **Grafik :** MALI 400



Şekil A.3: BeagleBone Black Model C.



Şekil A.4: HardKernel ODROID U3.

- **Bellek** : 2 GB DDR2
- **Çevre Birimleri** :
  - SD Kart girişi
  - eMMC girişi
  - 4 USB portu
  - Ethernet portu
  - 3.5mm jacklı ses çıkış port
  - HDMI çıkışı
- **Giriş / Çıkış** :
  - 2 UART arabirimi
  - SPI hattı
  - 3 GPIO pini
- **Fiyat** : 69\$

## ÖZGEÇMİŞ



**Ad Soyad:** Mustafa Haluk AKGÜNDÜZ

**Doğum Yeri ve Tarihi:** TRABZON 1979

**Adres:** Kartaltepe mh. Demirelli Sk. No: 5 D4 Bakırköy / İSTANBUL

**E-Posta:** akgunduz@gmail.com

**Lisans:** İstanbul Teknik Üniversitesi - Kontrol & Bilgisayar Mühendisliği

### Mesleki Deneyim ve Ödüller:

Televizyon, modem entegre ve mobil cihaz yazılımları ile araçlar arası haberleşme sistemleri (V2V), makineler arası haberleşme sistemleri (M2M) ve nesnelerin interneti (IoT) konularında toplam 13 yıllık bir tecrübeye sahiptir.

### Yayın ve Patent Listesi:

- **Akgündüz M.H.** (2009). A Broadcast Receiving Device And Control Method. Patent numarası: WO2009077898
- **Akgündüz M.H.** (2006). A method for resolution adjustment with using display detection technics. Patent numarası: TR200605918
- **Akgündüz M.H.** (2006). A method for resolution adjustment with using triggering technics. Patent numarası: TR200605917
- **Akgündüz M.H.** (2006). A method for resolution adjustment with using manual technics. Patent numarası: TR200605916

### TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR:

- **Akgündüz M.H. ve Adalı E.** (2015). Atanmış Sistemler ile Dağıtık Sistem Tasarımı ve Mevcut Sistemler ile Karşılaştırması. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*, Cilt 9, İstanbul, Türkiye.