KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİMDALI
YÜKSEK LİSANS TEZİ

MÜHENDİSLİK HESAPLAMALARI İÇİN MASAÜSTÜ KİŞİSEL BİLGİSAYAR BİLEŞENLERİ İLE GİRİŞ SEVİYESİ YÜKSEK BAŞARIMLI HESAPLAMA SİSTEMİNİN KURULUMU

Ahmet ÖZCAN

Ağustos 2016

ÖZET

MÜHENDİSLİK HESAPLAMALARI İÇİN MASAÜSTÜ KİŞİSEL BİLGİSAYAR BİLEŞENLERİ İLE GİRİŞ SEVİYESİ YÜKSEK BAŞARIMLI HESAPLAMA SİSTEMİNİN KURULUMU

ÖZCAN, Ahmet

Kırıkkale Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi

Danışman: Yrd. Doç. Dr. Halil Murat ÜNVER

Ağustos 2016, 73 sayfa

Günümüzde yazılım ve donanım teknolojilerinde hızlı değişimler yaşanmaktadır. Donanım alanındaki gelişmeler yazılımın gerisinde kalmaktadır. Özellikle ar-ge çalışmalarında kullanılacak yazılımlar için çok güçlü ve hızlı donanımlara ihtiyaç bulunmaktadır.

Ar-ge çalışmalarında sayısal işlem ve hesaplamaların kısa sürede tamamlanarak sonuçlandırılması büyük önem taşımaktadır. Bu amaçla yüksek

başarım elde etmeye yönelik hazırlanmış özel bilgisayar sistemleri üzerinde çalışacak sayısal hesaplamalar da özel metotlar ile gerçekleştirilmektedir. Bu yöntem günümüzde paralel hesaplama olarak bilinmektedir. Bu yöntem için oluşturulan özel sistemlere ise Yüksek Başarımlı Hesaplama Sistemi veya bilinen adıyla Süper Bilgisayar denilmektedir.

Geçtiğimiz 50 yılda yüksek başarımlı hesaplama sistemleri, mühendislik, tıp, kimya, fizik gibi pek çok bilim dalında zor, zaman alıcı problemlerin çözümünde kullanılmaktadır. Bu bilgisayarlar, hesaplama gücü olarak kişisel masaüstü bilgisayarlardan binlerce kat daha hızlı olabilmektedir.

Ülkemizde süper bilgisayar oluşturma çabaları 2000'li yılların başında başlamıştır. Halen ülkemizin en hızlı bilgisayarı, İstanbul Teknik Üniversitesi bünyesinde bulunan Ulusal Yüksek Başarımlı Hesaplama Merkezinde bulunmaktadır. Bu bilgisayar 374 düğüm, 2724 çekirdekten oluşmakta ve saniyede 6 trilyondan fazla işlem yapma kapasitesine sahiptir.

Yüksek başarımlı hesaplama sistemleri sunucu bilgisayarlarla ve özel seçilmiş ağ bileşenleriyle yüksek maliyetlerle kurulabilmektedir. Bu çalışmada düşük bütçeli masaüstü bilgisayar bileşenleriyle giriş seviyesi yüksek başarımlı hesaplama sisteminin kurulması gerçekleştirilmiştir.

Anahtar kelimeler: Süper Bilgisayar, Yüksek Başarımlı Bilgisayar, Yüksek Başarımlı Hesaplama Sistemi, Küme Bilgisayarlar, Kümeleme

ABSTRACT

SETTING UP ENTRY LEVEL INSTALLATION OF HIGH PERFORMANCE COMPUTING SYSTEMS WITH DESKTOP PERSONAL COMPUTERS COMPONENTS FOR ENGINEERING CALCULATIONS

ÖZCAN, Ahmet

Kırıkkale University

Graduate School of Natural and Applied Sciences

Department of Computer Engineering, Master Thesis

Supervisor: Assist. Prof. Dr. Halil Murat ÜNVER

August 2016, 73 pages

Today, technology is experiencing rapid changes in software and hardware. Hardware is falling behind the developments in the field of software.Especially very strong and fast hardwareis needed for software to be used in R&D work.

It is important that finalization of digital transactions and calculations completed in a short time in R&D. For this purpose, it prepared for obtaining high performance computing systems to run on specific numerical calculations are carried out using special methods. This method is known as parallel

computing today. In the special scheme for this method it is called High-Performance Computing Systems Supercomputer with or known.

Over the past 50 years, high-performance computing systems are used in engineering, medicine, chemistry, physics and difficult areas of science for solving the problem of time-consuming. These computers, computing power as a desktop personal computer can be thousands of times faster.

In our country, the effort to build supercomputer began in the early 2000s. Currently, the fastest computer in our country, Istanbul Technical University, located in the National High-Performance Computing Centre is located on-site. This computer has the capacity to process more than 6 billion per second consists of 1000 processor.

High-performance computing systems can be installed with the high cost of server computers and specially selected network components. In this study, the entry-level computer with a low-budget component was performed to establish the high performance computing systems.

Keywords: Supercomputer, High Performance Computer, High Performance Computing Systems, Cluster Computers, condensation

TEŞEKKÜR

Tez çalışmam boyunca beni motive eden, yönlendiren, yardımlarını esirgemeyen danışman hocam Sayın Yrd. Doç. Dr. Halil Murat ÜNVER' e teşekkür ederim. Akademik bilgi birikimiyle çalışmalarımda destek olan Sayın Yrd. Doç. Dr. Atilla ERGÜZEN' e teşekkür ederim. Akademik çalışmalarımda destek olarak, deneyimlerini paylaşan ve tez çalışmalarımda bana yardımcı olan arkadaşım Uzman Erdal ERDAL' a teşekkür ederim.

Yüksek lisans eğitimim süresince sabırla bana katlanan çok değerli eşim Sevgi ÖZCAN' a, çok değerli çocuklarım M. Doğukan ÖZCAN, Ezgi Nur ÖZCAN ve Ali Korhan ÖZCAN' a tüm içtenliğimle teşekkür ederim.

İÇİNDEKİLER DİZİNİ

ŞEKİLLER DİZİNİ

Şekil Sayfa

ÇİZELGELER DİZİNİ

KISALTMALAR DİZİNİ

HPC	High Performance Computer
MPI	Message Passing Interface
SOA	Service Oriented Architecture
CUDA	Compute Unified Device Architecture
GPU	Graphical Processing Unit
CPU	Central Processing Unit
VM	Virtual Machine
3D	3 Dimension
4D	4 Dimension
NUMA	Non Uniform Memory Access
SMP	Symmetric Multiprocessing
TRUBA	Türk Ulusal e-Bilim e-Altyapısı
GBPS	Giga Bit Per Second
KBPS	Kilo Bit Per Second
PE	Process Environment
AR-GE	Araştırma Geliştirme
IBM	International Business Machines
YBH	Yüksek Başarımlı Hesaplama
YBHS	Yüksek Başarımlı Hesaplama Sistemi
WCF	Windows Communication Foundation
MGA	Mesaj Geçiş Arayüzü
SOM	Servis Odaklı Mimari
R&D	Research and Development

1. GİRİŞ

Teknolojideki gelişmeler son yirmi yılda büyük ivme kazanmıştır. Bilgisayarların icadı olan 1900'lü yılların başından bu yana bilişim sektörü sürekli kendini geliştirerek yenilemektedir.

Bilişim sektörünün bu hızlı ilerleyişinde 1975 yılından günümüze işlemcilerde kullanılan transistör sayılarının her yıl bir kat daha artması etkili olmuştur. Transistör sayısı ile birlikte bilgisayar hızlarının katlanarak artışı ile bilişim sektörüyle ilgili tüm alanlarda yeni fikirlerin doğmasına ve paralelinde bu alanda yeni yazılımlara olan ihtiyacı ortaya çıkarmıştır.

Araştırmacıların hesaplama işlemlerini kısa sürede tamamlamak istemesi beraberinde hız gereksinimlerini ortaya çıkarmış ve süper bilgisayarların temel fikrini oluşturmuştur [1]. Yüksek başarımlı hesaplama zor mühendislik problemlerinin çözümünde süper bilgisayarlar ve küme bilgisayarları kullanılmaktadır [2]. Yüksek başarımlı hesaplamada temel prensip bir işin birden fazla işlemciye veya birden fazla bilgisayara iş parçalarına bölünerek yaptırılmasıdır.

1.1. Yüksek Başarımlı Hesaplama Sistemi

Günümüzde yüksek başarımlı hesaplama sistemleri veya bilinen adıyla süper bilgisayarlar klimatoloji, hesaplamalı tıp, yüksek enerji fiziği gibi pek çok alanda kullanılmaktadır. Ar-ge çalışmaları için maliyetleri düşüren

yüksek başarımlı hesaplama sistemleri günümüzün en esnek araştırma ortamları olarak kabul edilmektedir [3].

Endüstride ve bilimsel araştırmalarda oldukça fazla kullanılan bu sistemlerin hızları içlerinde barındırdıkları bileşenlerine göre farklılık gösterebilmektedir. Yüksek başarımlı hesaplama sistemlerinde kullanılan seçilmiş ana kart, işlemci, bellek, grafik kartı ve disk bileşenleri oluşturulacak sistemin hesaplama hızını doğrudan etkilemektedir. Bu sebeple yüksek başarımlı hesaplama sistemleri oluşturulurken birbiri ile uyumlu bileşenler tercih edilmektedir.

Günümüzde çok yüksek maliyetlerle hazırlanan hesaplama sistemleri ülkelerin ulusal düzeyde araştırma faaliyetlerine hizmet vermektedir. Her yıl yapılan bir çalışma ile dünyada bulunan süper bilgisayarlar çalışma hızları bakımından sıralanmakta ve oluşan liste internet üzerinden tüm dünyaya duyurulmaktadır. Bu amaçla hazırlanmış olan www.top500.org sitesinden dünyanın en hızlı hesaplama sistemleri görülebilmektedir.

1.2. Çalışmanın Amacı

Yüksek başarımlı hesaplama sistemlerinin hazırlanabilmesi için büyük maliyetlere ve konu üzerinde uzmanlaşmış personele gereksinim duyulmaktadır. Ancak bu durum süper bilgisayarların düşük bütçelerle oluşturulamayacağı anlamına gelmemektedir.

Bu alanda yapılacak çalışmalar mevcut araştırmacıların konu hakkında tecrübelerinin artmasına, karşılaşılabilecek sorunların önceden

bilinmesine, ortaya çıkabilecek maliyetlerin tahmin edilebilmesini sağlayacaktır. Küçük işletmelerin ve sınırlı bütçeye sahip araştırmacıların maliyeti fazla olan yüksek başarımlı hesaplama sistemi kurmaları pek mümkün görünmemektedir.

Bu çalışmada masaüstü bilgisayar bileşenleriyle, çok düşük maliyetle oluşturulan giriş seviyesi yüksek başarımlı hesaplama sistemi kurulumu yapılacaktır.

1.3. Literatür Araştırması

Scott LATHROP ve Thomas MURPHY "High Performance Computing Education" isimli makalelerinde eğitim sistemi içinde öğrencilerin öğrenme ve mesleki gelişimlerini revize etmek için hesaplamalı düşünme ile ilgili önerilerde bulunmuşlardır [4].

Niranjan GOVIND ve arkadaşları "Utilizing High Performance Computing for Chemistry: Parallel Computational Chemistry" adlı makalede paralel hesaplamalı kimya yazılımlarının durumu gözden geçirilmiş ve kuantum kimyası metodolojileri ile birlikte algoritmalar ve yazılım geliştirmeye etkileri incelenmiştir [5].

Volodymyr KINDRATENKO ve Pedro TRANCOSO "Trends in High-Performance Computing" isimli makalede HPC sistem mimarisinin geleneksel homojen küme yapısından heterojen küme yapısına dönüşümünü incelemiştir [6].

Moustafa ABDELBAKY ve arkadaşlarının “Enabling High-Performance Computing as a Service” adlı makalelerinde, doğru yazılım altyapısı sayesinde bulut platformlarının bir hizmet olarak ihtiyaç duyulan yüksek başarımlı hesaplama sistemlerine dönüşebileceği ifade edilerek, IBM Blue Gene /P sistemi çalışmada prototip olarak kullanılmıştır [7].

Lin SHI ve arkadaşlarının 2012 yılında yayınladıkları “vCUDA: GPU-Accelerated High-Performance Computing in Virtual Machines ” isimli makalesinde yüksek başarımlı hesaplamada grafik kartlarının adanmış sunucular ile sanal makineler üzerinden çoklu grafik işlemcilerin kullanımı araştırılmıştır. Sanal makinelerde HPC uygulamaları için GPU hızlandırmanın mümkün olduğu ifade edilmiştir [8].

Donald L. BEATY “High Performance Computing Data Centers” isimli makalesinde ihtiyaca göre araştırmalar için tahsis edilebilecek yüksek başarımlı veri merkezleri konusunu incelemiştir. Önerilen çalışma ile her an ihtiyaç duyulacak hesaplama işlemleri için veri merkezi anlayışıyla çalışan yüksek başarımlı hesaplama sistemlerine ilişkin öneriler sunulmuştur [9].

Jean-Claude ANDRÉ ve arkadaşları 2014 yılında yayınlanan “High-Performance Computing For Climate Modeling” isimli çalışmalarında iklim değişiklikleri, küresel ısınma ile birlikte meteorolojik hava tahmin çalışmalarında yüksek başarımlı hesaplama sisteminin kullanımına ilişkin önerilerde bulunmuşlardır [10].

Xun JIA ve arkadaşlarının “Gpu-based high-performance computing for radiation therapy” isimli makalesinde radyo-terapi tedavisinde görüntü işleme metotları kullanılarak 3D/4D görüntüler ile radyasyon doz hesabının yapılması, yüksek çözünürlüklü görüntü işleme ile ilgili sorunların çözümünde grafik kartı tabanlı yüksek başarımlı hesaplama yöntemi kullanılmış ve öneriler getirilmiştir [11].

Thomas HERAULT ve Yves ROBERT “Fault-Tolerance Techniques for High-Performance Computing” isimli çalışmalarında yüksek başarımlı hesaplama sistemlerinin çalışması sırasında çökme, kapanma gibi sorunlarının hesaplamaya engel olmaması için yedekli yapılar oluşturulması hakkında öneriler getirmişlerdir [12].

James J. HACK ve Michael E. PAPKA “Next-Generation Machines for Big Science” isimli çalışmasında Amerikanın iki büyük araştırma laboratuvarında bulunan süper bilgisayarların ortak bir çalışmayla hızlarını artırarak elde edilen hesaplama gücünün enerji, biyoteknoloji, nanoteknoloji, malzeme alanlarında keşifler için ihtiyaç duyulan gereksinimi karşılayacağı ifade edilmiştir [13].

Muhammad SAEED ve arkadaşlarının 2015 yılında yayınladıkları “High Performance Computing Achieved in Personal Computers” isimli çalışmalarında kişisel bilgisayarlarla yüksek başarımlı hesaplama işleminin yapılabileceğini ve uygulama ile ilgili görüşleri sunulmaktadır [14].

Minakshi TRIPATHY ve Chandra TRIPATHY “A Comparative Analysis Of Some High Performance Computing Technologies” isimli makalede son 10 yılda uygulanan yüksek başarımlı hesaplama sistemi modellerinin karşılaştırmasını yapmış ve öneriler sunmuştur [15].

2. MATERYAL VE YÖNTEM

2.1. Yüksek Başarımlı Hesaplama (YBH) Biçimleri

Günümüzde bilimsel, teknolojik, askeri pek çok alanda kullanılan YBH kümeleri farklı şekillerde tasarlanabilmektedir. Bu tasarımların oluşmasında gizlilik, maliyet, iş yükü ve ortak çalışma faktörleri belirleyici olmaktadır.

Yüksek başarımlı hesaplama sistemlerinde bilgisayar yaklaşımları şu şekilde olabilmektedir;

- Çoklu işleme (multiprocessing)
- Bilgisayar kümelemesi
- Paralel süper bilgisayarlar
- Dağıtık hesaplama
- NUMA, SMP ve massively paralel sistemler
- Izgara (grid) hesaplama

2.1.1 Geleneksel YBH Sunucular

Sadece yüksek başarımlı hesaplama işlemleri yapmak üzere tasarlanmış sunuculardan oluşan yapıya geleneksel yüksek başarımlı hesaplama sistemi denilmektedir. Geleneksel yüksek başarımlı hesaplama sistemleri yüksek maliyetli donanım bileşenleriyle oluşturulan YBH küme bilgisayarlarını barındırmaktadır. Oluşturulan YBH kümeleri sadece bu işe adanmıştır.

YBH kümelerinin ilk oluşturulan biçimi olduğundan geleneksel olarak ifade edilmektedir. Özellikle gizlilik gerektiren uygulamaların hesaplama işlemlerinde tercih edilmektedir.

2.1.2 Bulut Tabanlı YBH Sunucular

Son yıllarda yüksek başarımlı hesaplama işlemlerinde yüksek maliyetli adanmış YBH sunucular yerine bulut tabanlı YBH sistemleri yoğun şekilde kullanılmaya başlamıştır. Araştırmacılar geleneksel HPC sistemlerinin yerine çok küçük maliyetler ve sınırlı süre ile kiralanan bulut platformlarında istedikleri ölçekte YBH kümeleri kurabilmektedir [16].

Bulut tabanlı YBH sistemleri küçük ölçekli deneyler ve araştırmalar için gereksiz araştırma ön maliyetlerini ortadan kaldırmıştır. Ancak bu yöntemin getirdiği birtakım olumsuz yönlerde bulunmaktadır. Geleneksel YBH sunucularda küme bilgisayarları arasındaki ağ ortamında sınırsız bant olarak ifade edilen 40 Gbps ve üzeri hızlarda iletişim mümkün iken bulut tabanlı YBH kümelerinde en iyi bulut platformlarında 10 Gbps hız seviyelerine ulaşılabilmektedir.

2.1.3. Grid Tabanlı YBH Sunucular

Farklı konumlarda bulunan farklı bileşenlere sahip yüksek başarımlı hesaplama sistemlerinin internet veya geniş alan ağı bağlantılarıyla ortaklaşa oluşturduğu yapıya verilen genel isimdir. Bu yapıda büyük küçük tüm YBH kümeleri ağ ortamı aracılığıyla verilen büyük ölçekli görevleri paylaşabilmekte ve üzerine düşen görevleri yerine getirmektedir.

Ülkemizde TR-GRID Truba isimli grid tabanlı bir yüksek başarımlı hesaplama platformu bulunmaktadır. Grid üzerinde çalışabilir uygulamalara sahip Türkiye'deki tüm üniversite, araştırma kurumu ve enstitülerden araştırmacılar, öğretim üyeleri, öğrenciler TRUBA kaynaklarını kullanmak üzere TRUBA kullanıcısı olabilmektedir [17].

Altyapının kullanılması için çalışabilir uygulamalar aşağıdaki özelliklerden en az birini içermesi beklenmektedir:

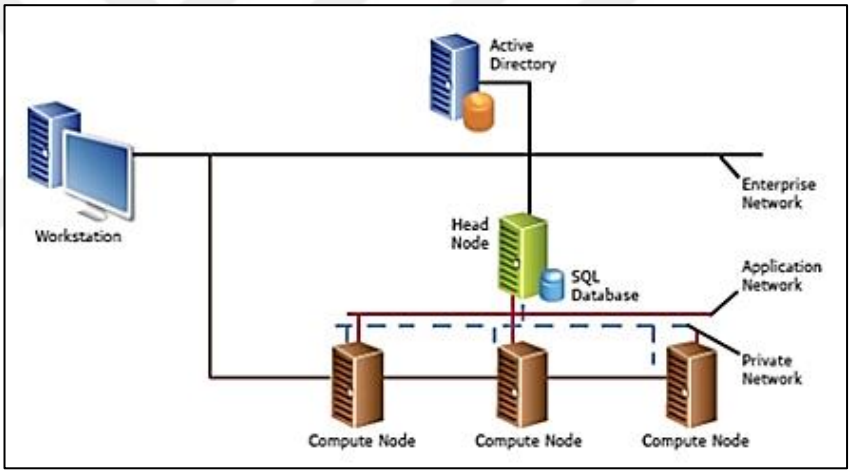
- Birçok farklı bağımsız alt göreve bölünebilen yüksek geçişli hesaplamalar,
- Büyük miktarda verinin işlenmesini gerektiren yüksek performanslı hesaplamalar,
- Küçük bir zaman diliminde çok sayıda işlemin yapılmasını gerektiren işlemci merkezli hesaplamalar,
- Verilerin bir araya getirilmesi, saklanması ve çözümlenmesini de gerektirebilen veri merkezli hesaplamalar.

2.2. YBH' yi Oluşturan Donanımsal Birimler

YBH kümelerinin bulunduğu her veri merkezi kendine özgün bir tasarıma sahiptir. YBH kümelerinde sorun çok parçaya bölünerek hesaplama için ayrılmış birden çok sunucu tarafından sorunun bir bölümünün çözülmesi ve çözümlerin birleştirilmesi karakteristiği bulunmaktadır. Bu sebeple çözümde çok sayıda paralel sunucu kullanılmaktadır. Sunucular arasındaki iletişim hızı sorunun çözümünde kritik bir rol oynamaktadır.

Oluşturulan tasarımda gerçekleştirilecek işlerin dağılım desenleri ve iş yönetim şemaları da etkili olmaktadır. Bunların yanında bütçedeki sınırlamalar tasarımda etkilidir.

YBH kümelerinin tasarımında işlemci mimarisi ve işlemci modeli, platform mimarisi, bellek sistemi, iletişim altyapısı, depolama altyapısı ve yönetim platformu bileşenleri dikkate alınmaktadır. Temel YBH küme yapısı Şekil **2.1.** de görülmektedir.



Şekil 2.1. Temel YBH küme yapısı [1].

2.2.1 İşlemciler

Kısa sürede bitirilmesi istenen, çözüm süresi haftalar veya aylar alabilen karmaşık problemlerin en kısa sürede çözümün hesaplanabilmesi için paralel çalışabilen yüksek hızlı işlemcilere ihtiyaç duyulmaktadır.

Bu amaçla mikro işlemci üretimi yapan firmaların özel işlemci modellerinin yanında sunucu veya masaüstü işlemci modelleri de yüksek başarımlı hesaplama sistemlerinde kullanılabilmektedir.

Çok çeşitli paralel bilgisayar (işlemci) yapıları vardır. Bu çeşitler, işlemciler (işleme elemanı olarak adlandırılır-PE) arasındaki veya işlemci ve hafıza arasındaki bağlantıya göre belirlenir. Paralel işlemci makineleri simetrik (tüm işlemcilerin aynı seviyede olması) ve asimetrik (işlemcilerin bazı görevler için ayrılması ve önceliklerinin olması) çoklu işlemciler olarak ikiye ayrılır [18].

Bu konudaki genel yaklaşım, n adet paralel işlemciden oluşan bir sistemin, n kat hızlı tek bir işlemciden daha az verimli fakat çok daha ucuz olmasıdır.

Çok fazla hesaplama gerektiren, bitirilmesinde zaman kısıtları olan ve özellikle n adet thread`e bölünebilen görevler için paralel hesaplama mükemmel bir çözüm olarak görülmektedir. Aslında geçtiğimiz yıllarda, süper bilgisayar olarak bilinen yüksek performanslı hesaplama sistemleri paralel bir mimariye sahiptir.

2.2.2. Düğümler

Yüksek başarımlı hesaplama sistemlerinde sistemin işlevselliğini belirleyen ve hesaplama kümesini oluşturan her sunucu veya iş istasyonu düğüm adını almaktadır. Oluşturulan hesaplama sisteminin büyüklüğüne göre sistem içerisinde Başdüğüm, Hesaplama Düğümü, İş İstasyonu Düğümü ve Aracı Düğümü gibi düğümler bulunabilmektedir.

Başdüğüm (HeadNode): Kümenin yönetimi ve iş planlaması hizmeti vermektedir. Kurumsal ağa bağlı olarak çalışır. Kurumsal ağdaki kullanıcı bilgisayarlarından gelecek Servis Odaklı Mimari (SOA) isteklerini dinler ve gönderilen işlerin hesaplama düğümleri arasında dağıtımını gerçekleştirir.

Hesaplama Düğümü (ComputeNode): Küme işlemlerini kabul ve çalıştırma işlevlerini gerçekleştirir. Servis Odaklı Mimari (SOM) işlemlerini gerçekleştirir. Aynı zamanda Mesaj Geçiş Arayüzü (MGA) işlemlerini de gerçekleştirir. Hesaplama işlemlerinin gerçekleştirildiği birimdir.

İş İstasyonu Düğümü (WorkstationNode): Küme içerisinde sürekli yer almayan fakat yüksek işlem gücüne sahip sunucular istenildiği takdirde yüksek başarımlı hesaplama kümesine dahil edilebilmektedir. Bu sunucular da hesaplama düğümü gibi SOM ve MGA hesaplama işlemlerini gerçekleştirebilmektedir.

Aracı Düğümü (BrokerNode): Kurumsal ağa bağlı olarak çalışır. İstemcilerin SOM isteklerini alır, küme düğümlerine dağıtır ve ardından cevapları toplayarak istemciye geri gönderir. SOM oturumlarını yönetir ve görüntüler [19].

2.2.3. Ağ Yapısı

Yüksek başarımlı hesaplama sistemlerinde parçalara bölünen hesaplama işleri ağ üzerinden gerçekleştirilmektedir. İş parçalarının hesaplandığı sunucuların performansının yanında ağ içerisindeki iletişim

performansı yüksek başarımlı hesaplama sisteminin başarımını doğrudan etkileyecektir. Bu nedenle yüksek başarımlı hesaplama sistemlerinin ağ tasarımında akılcı davranmak gerekli olacaktır [20].

Yüksek başarımlı hesaplama sistemlerinde kullanılan ağ tasarımında iki önemli faktör bulunmaktadır. Bunlar gecikme ve bant genişliğidir.

Gecikme, küçük bir mesaj verisinin ağ üzerindeki bir bilgisayardan diğerine gitmesi için geçen süredir. Gecikme mikrosaniye veya nanosaniye gibi çok küçük zaman dilimleri ile ölçülür. Tasarlanan sistemde günümüzde popüler olan sınırsız bant (infiniband) gibi düşük gecikmeli bir bağlantı söz konusu ise performans daha iyi olacaktır.

Bant genişliği ise, ağ üzerinde birim zamanda ne kadar bilginin taşınabileceğidir ve saniyede megabayt, gigabayt cinsinden veri miktarıyla ölçülmektedir. Günümüzde yerel ağ bağlantılarında Ethernet ile 1 Gigabit veya 10 Gigabit verinin taşınabildiği bilinmektedir. Performansın yüksek olması istenen sistemlerde ise Infiniband (Sonsuz band) bağlantılarla 20 Gigabit ile 100 Gigabit arasında değişebilmektedir.

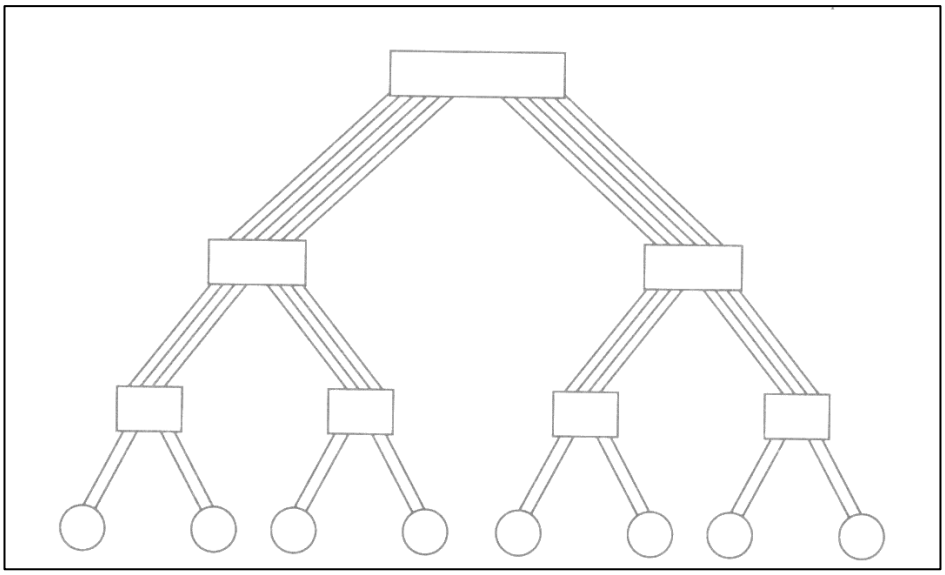
Çizelge 2.1. Sınırsız bant anahtar hızları [21]

	Tek Veri Oranı (SDR)	Çift Veri Oranı (DDR)	Dörtlü Veri Oranı (QDR)	Dolu Veri Oranı (FDR)	Gelişmiş Veri Oranı (EDR)
Sinyal Oranı (Gb/sn)	2.5	5	10	14	25
Port başına teorik başarım hızı (Gb/sn)	2	4	8	13.64	24.24
4 Port için hız (Gb/sn)	8	16	32	54.54	96.97
12 Port için hız (Gb/sn)	24	48	96	163.64	290.91
Kodlama (Bitler)	8/10	8/10	8/10	64/66	64/66
Gecikme (Mikrosaniye)	5	2.5	1.3	0.7	0.5
Geliştirildiği Yıl	2001	2005	2007	2011	2014

Yüksek başarımlı hesaplama sistemlerinde hangi tür ağ bağlantısının kullanılacağının belirlenmesi kadar topolojide önem taşımaktadır.

Topoloji ağa bağlanacak hesaplama düğümlerinin yerleşim biçimlerini ifade etmektedir. Süper bilgisayarların tasarımında en çok kullanılan üç topoloji bulunmaktadır. Bunlar; Fat tree, Torus ve Dragonfly olarak adlandırılmaktadır.

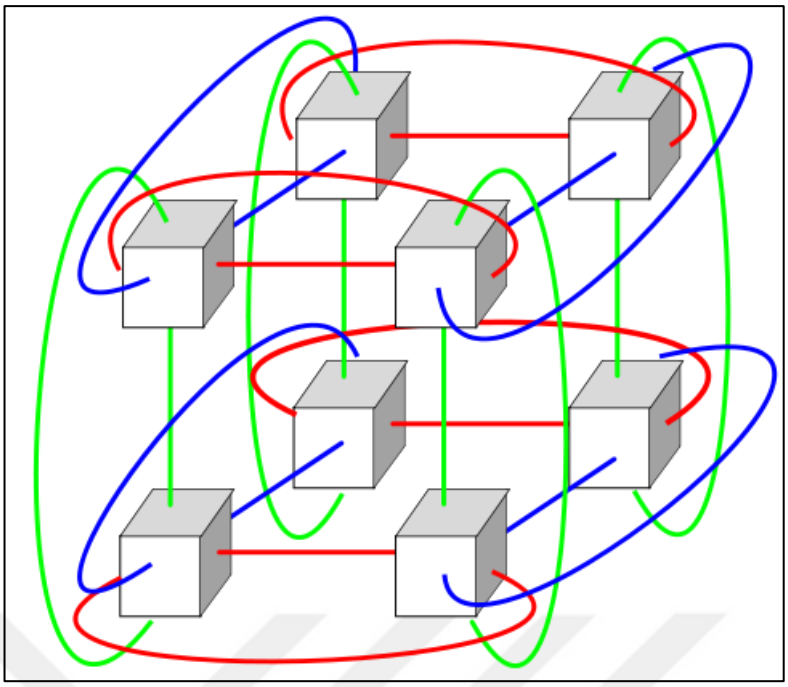
Şişman Ağaç (Fat Tree) topolojisinde tıpkı bir ağaç gibi, ağacın uç noktalarından gövdeye uzanan ağ bağlantılarının sayısı artarak ilerlemektedir. Yerleşim genişletilmiş yıldız topolojisine benzemektedir. Yaprak düğümlerdeki isteğin ağacın köküne doğru ilerledikçe artan yol sayısı sayesinde ihtiyaç duyulan bant genişliğine ulaşılmış olacaktır. Topolojiye ait yerleşim örneği Şekil **2.2.** de görülmektedir.



Şekil 2.2. Şişman ağaç ağ topolojisi [22]

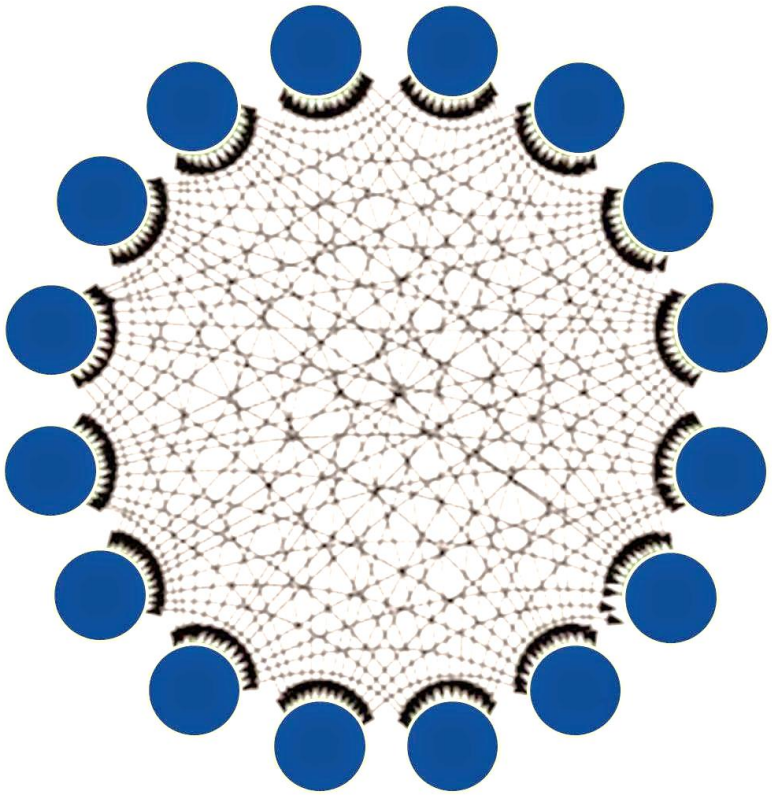
Yumru (Torus) topolojisinde ise düğümler silindir veya küre oluşturacak biçimde birbirlerine ağ kablolarıyla bağlıdır. Bu yerleşim oldukça maliyetli olmasının yanında önemli performans artışı sağlamaktadır. Bu topoloji istenilen boyutlarda gerçekleştirilebilmektedir. Bu yönüyle ölçeklenebilirliği en yüksek sistemdir. Düğüm durumuna göre iki boyutlu (2D), veya üç boyutlu (3D) olarak adlandırılabilmektedir.

İki boyutlu yumru yerleşiminde, düğümler satır ve sütun olarak iki boyutlu bir dikdörtgen kafes oluşturduğu düşünülebilir. Üç boyutlu yumru yerleşiminde, düğümlerin dikdörtgen prizma şeklinde üç boyutlu kafes oluşturduğu düşünülebilir. Şekil **2.3.** de görülmektedir.



Şekil 2.3. 3D Yumru ağ topolojisi [23]

Yusufçuk (DragonFly) topolojisinde, birkaç gruptan oluşan bilgisayar kümeleri birbirlerine en az bir ağ kablosuyla bağlanmıştır. Her grubun içindeki topolojiler farklı olabilmektedir. Örneğin bir gruptaki küme bilgisayarları Şişman ağaç topolojideyken bir başkasında iki boyutlu yumru olabilmektedir. Böylece herhangi bir düğüm diğer düğümlerin tamamına fiziksel olarak bağlanmış olacaktır. Yusufçuk topolojisi kablolama maliyetlerini artırmanın yanında bant genişliğine olumlu katkıda bulunmaktadır. Şekil **2.4.** de örnek topoloji şeması görülmektedir.



Şekil 2.4. Yusufçuk ağ topolojisi [24]

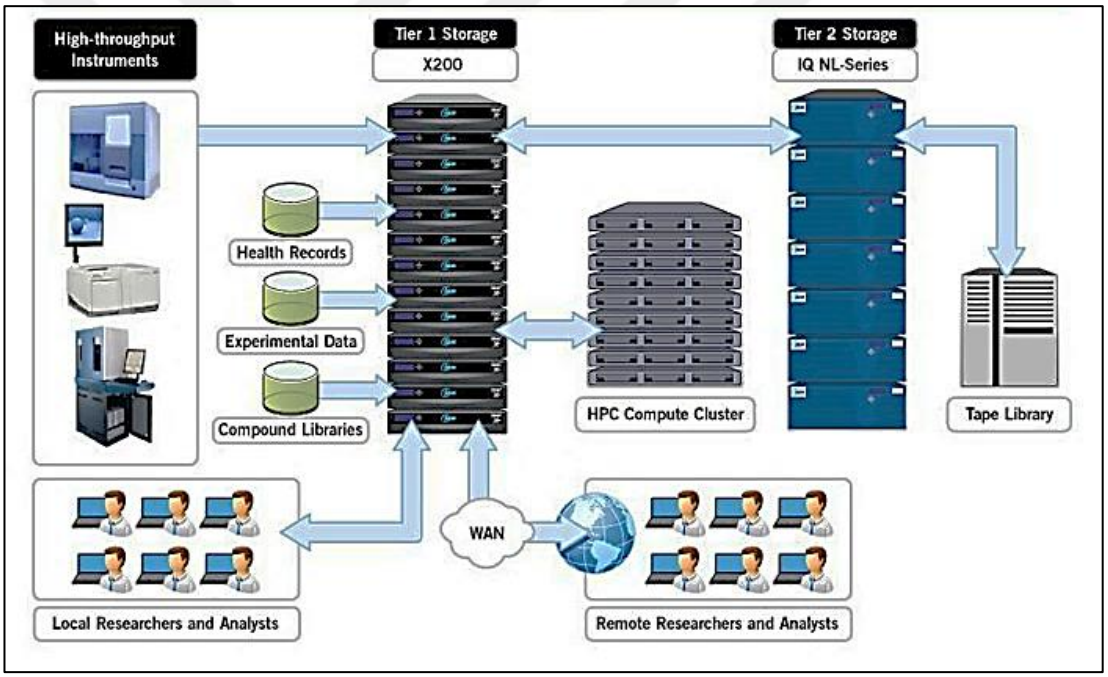
2.2.4. Depolama Alanları

Yüksek başarımlı hesaplama sistemi için kullanılan veri depolama alanları klasik veri merkezlerinde kullanılan veri depolama alanlarının ötesinde yüksek performanslı ve yüksek bant genişliğine sahip depolama alanlarıdır.

Her depolama bileşenlerini oluşturan, donanım, yazılım, küme, dosya paylaşımı, depolama aygıtı ve sunucu düğümü seçiminde yüksek performanslı bileşenler tercih edilir. Bu durum Rastgele I/O performansı olarak bilinen sistem performansını olumlu etkilemektedir.

Modern depolama mimarilerinde bilinen dosya sistemlerinin yanında, açık dosya sistemleri, XF, GFS, MogileFS, Lustre, Glustre gibi özel dosya sistemleri de kullanılabilmektedir.

Yüksek başarımlı hesaplama sistemlerinde ideal depolama mimarisi, yüksek performanslı disklerle dolu çoklu depolama denetleyicilerinden oluşabilir. Böyle bir tasarımda her denetleyici kendi işlemci, kapasite ve ön belleğine sahip olmalıdır. Şekil **2.5.** de örnek bir depolama alanı görülmektedir.



Şekil 2.5. Tipik YBH iş akışı [25]

2.2.5. Yerleşim ve İklimlendirme

Yüksek başarımlı hesaplama sistemlerinin yerleşimleri klasik veri merkezi yerleşiminden pek farklı olmamaktadır. Ancak bazı durumlarda özel yerleşim şekilleri uygulanabilmektedir. İletişim ve soğutma özellikleri yerleşim tercihlerinde belirleyici olabilmektedir.

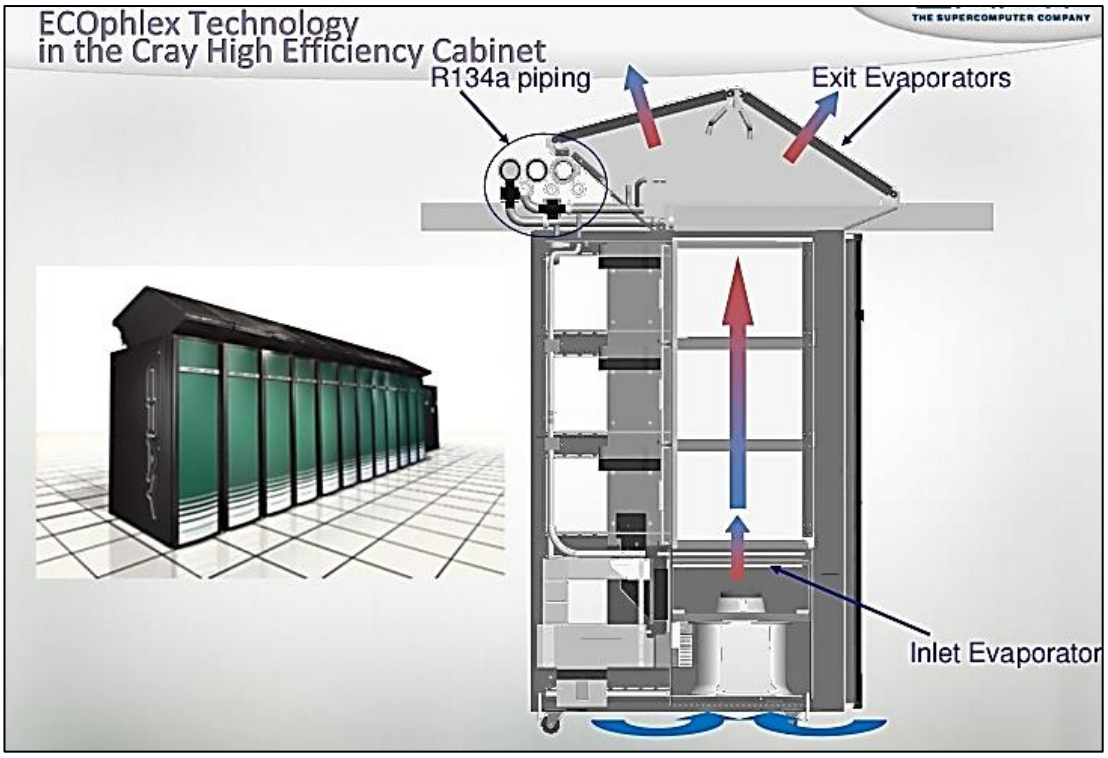
Günümüz veri merkezlerinin genelinde iklimlendirme ünitesinde soğutulan hava veri merkezine alt tarafta bulunan bir kanal tarafından üflenmektedir. Başka bir modelde ise sunucu kabinetlerinin ön tarafına üflenerek soğuk koridor oluşturulmaktadır.

Kabinetler tarafından ısınan hava ise üst taraftan emilerek dışarı atılmakta veya kabinetlerin arka tarafında bulunan sıcak koridordan emilerek yeniden soğutulmaktadır.

Klasik veri merkezi iklimlendirme sistemlerinin yanında bazı süper bilgisayarların soğutulmasında özel yöntemler kullanılabilmektedir. Örneğin Jaguar ve Kraken süper bilgisayarlarında Cray ECOplex sistemi kullanılmaktadır. Bu sistemde faz dönüşümlü sıvı aktarımı metodu kullanılmaktadır.

Bu metotta sıvı soğutma akışkanı düğümlerin bulunduğu kabinetlerde bulunan ısı eşanjörlerine pompalanmakta ve soğuyan eşanjör yüzeyindeki hava kabinet içerisine üflenmektedir. Bir başka deyişle kabinet içerisine merkezi olarak kontrol edilen klima sistemi bağlanmış durumdadır.

Yüksek hesaplama gücü beraberinde yüksek ısı oluşturmakta ve sistemin performansı bu ısının uzaklaştırılmasıyla doğru orantılı olmaktadır. Soğutma sisteminin yapısı Şekil **2.6.** da görülmektedir.



Şekil 2.6. Cray EcoPhlex Soğutma Sistemi [26]

2.3. YBHS' de Kullanılan Yazılımlar

Paralel bilgisayarlarda programlama için işletim sistemi seviyesinde ve programlama dili seviyesinde pek çok yazılım sistemi geliştirilmiştir. Bu sistemler, problemin parçalara bölümlenmesini ve işlemcilere atanmasını sağlayan çeşitli mekanizmalar içermelidir. Dolaylı paralellik (implicit parallelism) derleyici ya da diğer programın problemi bölümlemesi ve işlemcilere otomatik olarak atamasıdır.

Dolaysız paralellik (explicit parallelism) ise programcının problemin nasıl bölümleneceğini bildirmesidir. Şu an pek çok paralel işleme derleyici uygulamaları tek-katmanlı paralelliği desteklemektedir. Çok-katmanlı paralellik de paralel çalışan iş parçacıkları (thread) daha fazla paralellik için daha da bölünürler. Semafor (Semaphore) ve izleme (monitör) adı verilen senkronizasyon yapıları ile işlemlerin kaynakları paylaşımında bir anlaşmazlık olması durumu engellenmiştir [27].

Bir paralel programlama modeli, paralel algoritmaları açıklayan bir yazılım teknolojileri kümesidir. Bu model, uygulamalar, diller, derleyiciler, kütüphaneler, iletişim sistemleri ve paralel giriş/çıkış alanlarını kapsar.

Programcılar, kendileri ve uygulamaları için uygun bir model veya karma bir model seçip, uygulamalarını geliştirirler. Paralel modeller çok farklı şekillerde uyarlanırlar: klasik sıralı dillerden çağrılan kütüphaneler şeklinde, dil uzantıları şeklinde ya da tamamen yeni işleme modelleriyle. Bu modeller kabaca ikiye ayrılırlar: paylaşımlı hafıza sistemleri ve dağıtık hafıza sistemleri. Günümüzde bu iki sistem arasındaki çizgi oldukça bulanıklaşmıştır.

Sık kullanılan paralel programlama modelleri; PVM, MPI, OpenMP, Global Arrays, Co-Array Fortran, UPC, HPF, SHMEM, Occam, Linda, Cilk olarak sıralanabilmektedir.

2.3.1. İşletim Sistemleri

YBH sunucu kümeleri yaygın olarak Linux işletim sistemi kullanmaktadır. Windows tabanlı sunucu işletim sistemleri de YBH kümelerinde kullanılmaktadır. Microsoft firması piyasada yaygın olarak kullanılan sunucu işletim sistemlerine yüklenen yama yazılımlar ile kolayca YBH kümeleri oluşturulmasına olanak sağlamaktadır.

Bunun yanında Linux işletim sisteminin YBH kümelerine özel onlarca sürümü de özel satıcılar ve açık kaynak topluluğunca desteklenmektedir.

2.3.2. YBH Ek Yazılımları

Yüksek başarımlı hesaplama sistemlerinde kullanılan işletim sistemi üzerine yüklenerek, sistem üzerinde çalıştırılacak işlerin planlanmasını, verilerin planlanmasını, izlenmesi gerçekleştiren yardımcı yazılımlar kullanılmaktadır.

Bu yazılımlar lisanslı ve ücretli olabildiği gibi pek çoğu açık kaynak kodlu ve ücretsiz olabilmektedir. Özellikle Linux işletim sisteminde kullanılan pek çok ek yazılımın ücretsiz olarak kullanıldığı bilinmektedir.

Bu yazılımlar, iş planlama (Job Scheduling), veri planlama (Data Scheduling), izleme (monitoring) ve hepsi bir arada (All in One) yazılımlar olarak sınıflandırılabilmektedir. HPC kümelerinde kullanılan yardımcı yazılımlardan bir kısmı Çizelge **2.2.** de görülmektedir.

Çizelge 2.2. YBH kümesinde kullanılan yardımcı yazılımlar [28]

Yazılım	Kategori	Desteklenen İşletim Sistemi	Ücretli/Ücretsiz
Stacki	Hepsi bir arada	RHEL, CentOS, Oracle, Scientific Linux	Ücretli
DIET	Hepsi bir arada	Unix-like, Mac OS x, AIX	Ücretsiz
Ganglia	İzleme	Linux, Windows, Free BSD, Mac OSx, AIX, HPUX	Ücretsiz
Globus Toolkit	İş/Veri Planlama	Linux	Ücretsiz
Grid MP	İş Planlama	Windows, Linux, Mac OSX, Solaris	Ücretli
LanderCluster	İş Planlama / İzleme	Windows, Linux, Unix	Ücretli
Moab Cluster Suite	İş Planlama	Linux, Mac OSx, Windows, AIX, Solaris, Others	Ücretli
OpenLava	İş Planlama	Linux	Ücretsiz
ProActive	Hepsi bir arada	Unix-Like, Windows, Mac OSx	Ücretsiz
SLURM	İş Planlama	Linux, *nix	Ücretsiz
UniCloud	Hepsi bir arada	Oracle Linux, RHEL, CentOS	Ücretli
Univa Grid Engine	İş Planlama	*nix, Windows	Ücretli

2.3.3. Dosyalama Sistemleri

Yüksek başarımlı hesaplama sistemlerinde bilinen işletim sistemlerine ait dosyalama sistemleri yanında bu alanda çalışmaları bulunan firmaların özgün dosyalama sistemleri de bulunmaktadır. Özellikle depolama (storage) tarafında çalışan bu dosyalama sistemlerinde, veri tekilleştirme, veri bütünlüğü ve doğruluğu, kayıpsız sıkıştırma, hızlı okuma ve yazma gibi özelliklerin ön plana çıktığı görülmektedir.

Depolama alanında kullanılan dosya sistemlerinden birkaçı, Hadoop Distributed File System (HDFS), IBM General Parallel File System(GPFS), Google File System(GFS), Oracle Cluster File System(OCFS), StorNext, Global File System 2 (GFS2), Lustre File System, Panasas File System (PanFS), Fraunhofer, Gluster olarak sıralanabilmektedir.

Dosyalama sisteminin seçiminde ise yapılacak hesaplama işleminin özelliği belirleyici faktör olmaktadır. Örneğin dosyalama sisteminden veri tekilleştirme, kayıpsız sıkıştırma gibi özellikler isteniyorsa buna uygun dosyalama sistemi kullanımı uygun olabilmektedir.

Yüksek performanslı hesaplama sistemine uygun dosyalama sisteminin seçiminde bir takım testler kullanılmaktadır. Bu testler, write(dd), write(untar), write(small writes), read(grep), read(recursive ls), read(recursive directory info with ls and du), mixed IO test şeklinde sıralanabilmektedir.

2.3.4. İş Planlama Yazılımları

İş planlama yazılımları yüksek başarımlı hesaplama sistemlerinde çalışan bilgisayar uygulamalarının katılımsız olarak kontrol edilmesini ve arka planda işin yürütülmesinden sorumlu programlardır.

Bir iş, iş planlama yazılımına gönderildiği zaman, iş daha önce kuyruğa gönderilmiş iş bitene kadar bekler. İşin tamamlanması için harcanan zaman, kuyruktaki iş miktarına, bekleme süresine, işin önceliğine bağlıdır.

Sistemin başarımı, her hesaplama düğümündeki tamamlanan iş sayısının bitirilme süresiyle doğru orantılıdır. Daha kısa zamanda daha fazla işin tamamlanması performansın yüksek olduğunu belirtmektedir.

Yüksek başarımlı hesaplama sistemlerinde planlama özel bir dikkat gerektirir. Çünkü paralel işler birkaç alt görevden oluşur. Her alt görev ayrı bir hesaplama düğümüne gönderilerek işletilir. Bu işlem esnasında o düğüme ait işlemciler o iş için atanır. Bu durum işlem sonuçlandırılıncaya kadar devam eder.

İş planlama yazılımı düğümlerle hızlı bir ağ bağlantısına sahip olmak zorundadır. Çünkü her düğümde hızlıca yapılan hesaplama işleminin sonuçlarının işlemciye paralel hızlarda planlama yazılımınca bir araya getirilmesi ve sonuçlandırılması gerekmektedir. Bu durumda düğümler arasındaki ağ yapısı sistemin performansını doğrudan etkilemektedir.

İş planlama yazılımlarına gönderilecek paralel işlerden iyi sonuçlar alınabilmesi için yazılacak kodların iyi bir tasarım sürecinden geçmesi gereklidir. Böylece sistem kaynakları verimli kullanılarak iş verimliliği sağlanmış olacaktır.

Yüksek başarımlı hesaplama sisteminin yoğun kullanıldığı zamanlarda, sistem kaynaklarının adil dağıtılabilmesi her kullanıcı için önemlidir. Bu özelliğe sahip bir planlama yazılımı sayesinde kuyruk planlaması ve daha önceden yürütülen işlere ait geçmiş verileri daha dinamik bir planlama için kullanılabilmelidir. Böylece gönderilen işlerin önceliklerinin dinamik olarak yapılandırılması kullanıcıların sistemi daha adil kullanmalarını sağlayacaktır.

Çoğu iş planlama yazılımında iş kuyruklarının ve planlama algoritmalarının yapılandırılması için birkaç parametre bulunmaktadır. Bu parametrelerin kullanımıyla işler için farklı yanıt süreleri ve kullanım yüzdeleri alınabilmektedir. Genellikle sistemin kullanım süresinin artması, işler için ortalama yanıt süresinin artması anlamına gelmektedir.

Ortalama yanıt süresinin artması ise iş için hazırlanan paralel programın algoritmasının yetersiz olması ve iş profilinin başarısız olduğu anlamına gelmektedir.

Yanıt süresinin kısaltılması için paralel kodları oluşturan algoritmanın iyileştirilmesi ve iş profilinin uygun hale getirilmesi gereklidir.

Yüksek başarımlı hesaplama sistemi sahibi olan kuruluşlar sistemin verimli kullanılabilmesi için kabul edilebilir ortalama yanıt süreleri

kullanmaktadır [29]. Günümüzde pek çok yüksek başarımlı hesaplama sisteminde kullanılan iş planlama yazılımlarından birkaç tanesi Çizelge **2.3.** de gösterilmiştir.

Çizelge 2.3. YBH kümesinde kullanılan iş planlama yazılımları [30]

Ürünün Adı	Arayüz	Veri Depolama	Platform
Job Arranger for Zabbix	Fat Client, CLI	MySQL, PostgreSQL	Linux, Windows
Job Scheduler	Browser, GUI, Fat Client, Command Line, Web Services	XML, Oracle, MySQL, SQL Server, DB2, PostgreSQL	Linux, Windows
Job Server	Browser, CLI	Oracle, MySQL, PostgreSQL	Linux, Windows, MacOS
Obsidian Scheduler	Java API, REST API, Browser	İlişkisel Veritabanı	JVM çalıştıran herhangi sistem
Open Lava	C/C++, Python	Built-in	Linux
ProActive Workflows & Scheduling	Java API, REST API, CLI, Web Portals, R, Matlab, SciLab	HSQLDB veya Harici Veritabanı	JVM çalıştıran herhangi sistem
RUNDECK	WebGUI, CLI, API	İlişkisel Veritabanı ile JDBC Desteği	Unix, Linux, Windows
Schedulix	WebGUI, CLI, API	İlişkisel Veritabanı ile JDBC Desteği	Unix, Linux, Windows
Visual Cron	Windows Client, Web GUI, .Net API, REST API, CLI	SQL CE	Windows

2.3.5. Uygulama Yazılımları

Yüksek başarımlı hesaplama sistemlerinde kullanılan uygulama yazılımları, sık kullanılan uygulama kütüphanelerinden oluşmaktadır. Bu yazılımların en önemli özelliği ise matematiksel işlemler için işlemciyi verimli kullanmaları ve optimize edilmiş olmalarıdır.

MATLAB, güçlü sayısal hesaplama uygulamasıdır. MATLAB ve yardımcı araç kutuları, teknik hesaplama uygulamalarında mühendisler, bilim adamları, matematikçiler ve akademisyenler için çalışma ortamı sağlamaktadır.

ANSYS, mühendis ve bilim adamlarının dayanım, titreşim, mekanik, ısı transferi ve elektromanyetik konularındaki etkileşimini canlandırmak için kullanılan uygulama yazılımıdır.

Alanda çok bilinen bu yazılımların yanında; PGI Fortran/C/C++, Intel C/C++, Intel Fortran, g77, g++ gibi derleyiciler hesaplama sistemlerine yüklenmekte ve paralel kodlarla yazılmış programların derleme işlemleri gerçekleştirilmektedir.

Ayrıca görüntü işleme gibi oldukça zaman uygulamaların YBH kümelerinde paralel olarak işlenmesiyle medical alanında kısa zamanda etkili sonuçlar alınabilmektedir [31].

Aynı zamanda Atlas, Asap, CUDA, Freeglut, HDF5, Intel MKL, Intel MPI, Lapack, Matplotlib, Mpich, Mvapich, OpenCL, PAPI gibi kütüphaneler ile paralel işlemler gerçekleştirilmektedir.

Ayrıca YBH kümelerinde Abinit, Amber, Atomic Simulation Environment, Cclm, Dacapo, Fluent, Gaussian, Gpaw, Jacapo, Material Studio gibi programlar paralel olarak koşturulabilmektedir [32].

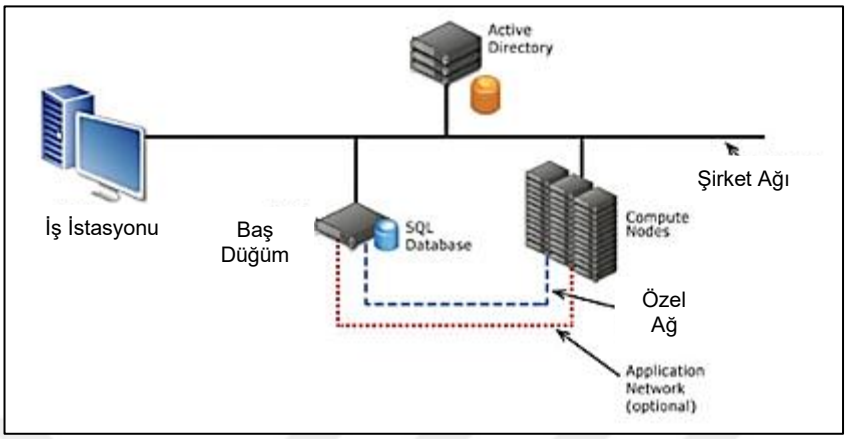
2.4. YBH Sisteminin Kurulumu ve Test Edilmesi

Yüksek başarımlı hesaplama sistemi (High Performance Computing) kurulumu öncesinde hangi işletim sistemi üzerinde çalışacağına karar verilmelidir. En yaygın kullanılan işletim sistemleri Linux ve Microsoft Windows işletim sistemleridir.

Bu çalışmada Microsoft Windows Server 2012 işletim sistemi üzerinde çalışmakta olan bir Yüksek Performanslı Hesaplama Sistemi kurulmaktadır. Yüksek başarımlı hesaplama sistemi olarak Microsoft küme mimarisi kullanmanın, mevcut kurumsal yapıya entegre olabilmesi, basitleştirilmiş küme yapısına sahip olması, şablona dayalı iş yönetimi, baş düğüm aracılığıyla küme dağıtımı ve yapılandırma özellikleri avantajlı yönleri olarak görülebilmektedir.

Windows tabanlı bir yüksek başarımlı hesaplama sisteminin kurulabilmesi için Active Directory Domain yapısına gereksinim bulunmaktadır. Hesaplama kümesinde kullanılan düğümler (nodes) ve düğümlerin yönetiminden sorumlu baş düğüm (headnode) bu domain yapısına katılmak

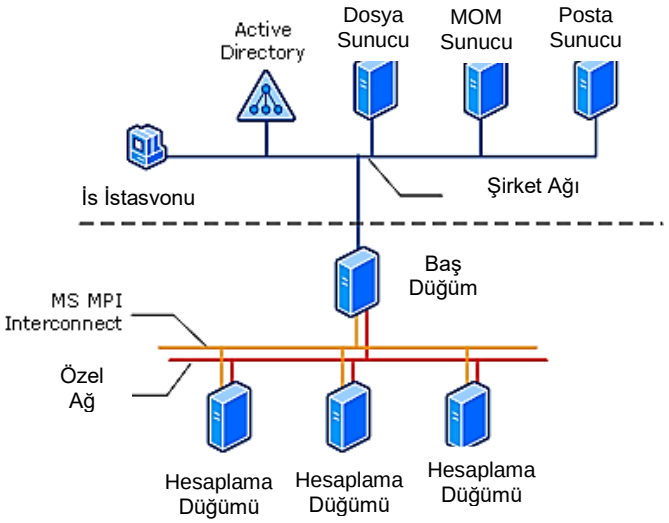
durumundadır. Oluşturulacak sistemde istenildiği takdirde bağımsız iş istasyonları da hesaplama işlerine katılabilmektedir.



Şekil 2.7. Temel YBH kümesi [33]

Oluşturulacak hesaplama sisteminin çalışma durumuna göre kümenin yerleşimi farklı olabilmektedir. Genellikle önem gerektiren uygulamaların çalıştırılacağı hesaplama sistemlerinde düğümler kurumsal ağdan yalıtılmış durumda olabilmektedir.

Ancak her iki yerleşimde de ana düğüm küme düzeyinde işleri yürütme, yapılandırma ve yönetme işlevlerine sahiptir.



Şekil 2.8. Yalıtılmış YBH küme yapısı [34]

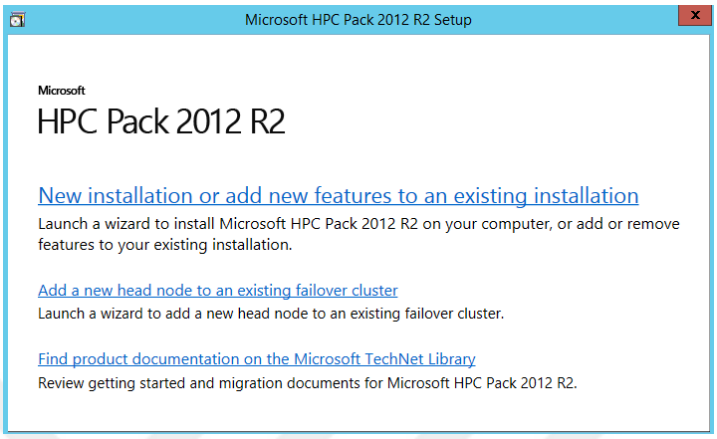
Yüksek başarımlı hesaplama kümesi kurulumu genel olarak üç adımdan oluşmaktadır; Kümeyi (Cluster) oluşturma, Kümeyi yapılandırma, Düğümleri oluşturma ve yapılandırmadır.

2.4.1. Kümenin Oluşturulması

Küme kurulma işleminde öncelikle sunucularımızdan birine Microsoft Server 2012 işletim sistemi kurulmalı ve Active Directory Domain Services rolü yüklenmelidir. Bu sunucumuz kurumsal ağımıza bağlanmalıdır.

Baş düğüm görevi üstlenecek sunucumuza da Microsoft Server 2012 işletim sistemi kurulmalı ve daha önce oluşturduğumuz Aktif dizin etki alanına (Active Directory Domain) katılmalıdır. Bu sunucumuzda da en az iki ağ kartı (tercihen 3 ağ kartı) bulunmalıdır. Bunlardan bir tanesi kurumsal ağa bağlanmalı, diğeri ise düğümlerin bulunduğu özel ağa bağlanmalıdır.

Güncelleme ve ağ yapılandırmaları sonrasında Microsoft firmasının geliştirdiği Microsoft HPC Pack 2012R2 yazılımı yüklenerek küme kurulumu başlatılmalıdır.



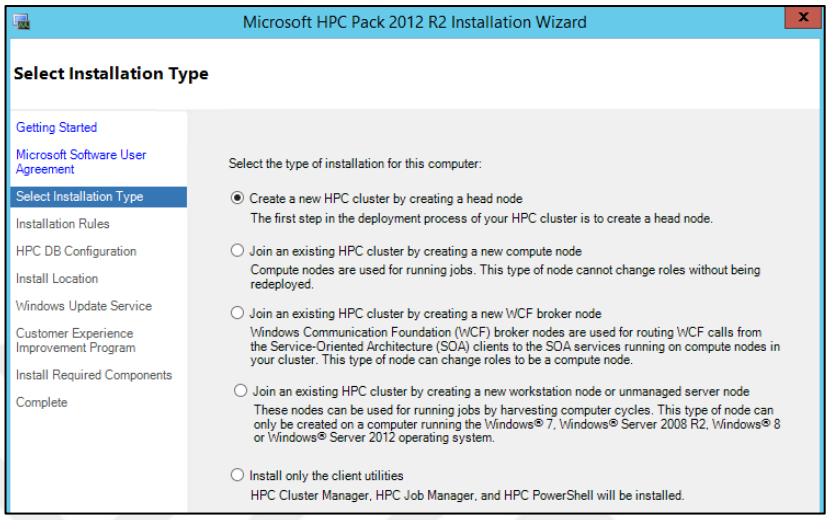
Şekil 2.9. HPC Pack 2012 R2 kurulumu

Kurulum işlemi başlatıldığında bir sihirbaz yardımıyla kurulum adımları gerçekleştirilecektir. Kurulum adımlarında sırasıyla yazılım kullanım anlaşması, kurulum tipi, kurulum ilkeleri, veri tabanı seçimi, kurulum yeri, güncelleme servisi ile ilgili ekranlardan seçimler alınarak kurulum işlemi için ön gereksinimlerin sağlanması kontrol edilecektir.

Kontrol işlemi başarılı olduğu takdirde kurulum başarıyla tamamlanacaktır. Bu işlemlerin hatasız ve başarılı olarak sonuçlanabilmesi için kurulum öncesinde mutlaka kontrol edilmesi gereken bileşenlerin başında işletim sistemi güncellemeleri gelmektedir.

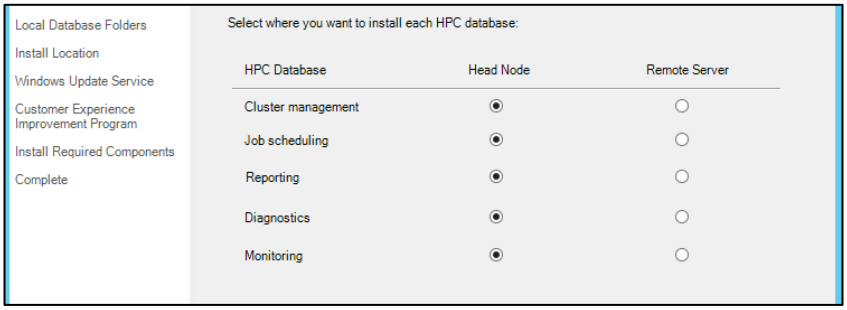
Bu adım eksik bırakılacak olursa kurulum aşamasında hatalar alınabilmektedir. Şekil **2.10.** da kurulum başlangıcında bulunan kurulum tipinin seçimi ekranı görülmektedir. Bu ekran ile yeni bir hesaplama kümesi

oluşturulabileceği gibi oluşturulan kümeye hesaplama düğümü eklenmesi işlemleri, hesaplama işi yaptıracak kullanıcı işlemleri de bu ekran aracılığıyla yapılabilmektedir.



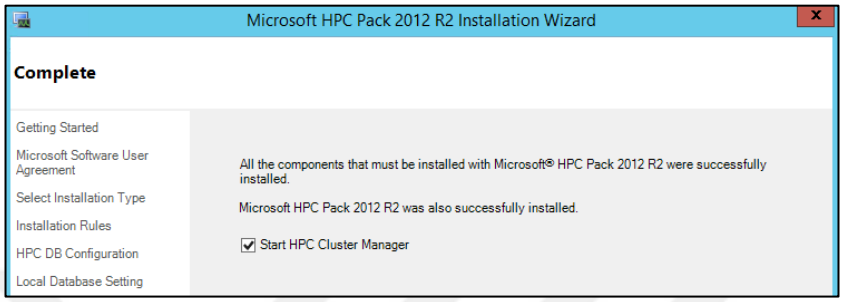
Şekil 2.10. HPC Pack 2012 R2 kurulum tipleri

Kurulum aşamalarının ilerleyen bölümlerinde kurulumunu gerçekleştirdiğimiz baş düğüm (HeadNode) için yüklenecek özellikler belirlenmektedir. Yazılımın hangi özelliklerinin hangi sunucularda çalışacağı bu ekranlardan özelleştirilebilmektedir. Şekil **2.11.** de baş düğüm (HeadNode) için yüklenebilecek roller görülmektedir.



Şekil 2.11. YBH Veri tabanı yapılandırma ekranı

Kurulum işlemi hatasız olarak tamamladığında küme yapımız hazır olacaktır. Bu durum Şekil **2.12.** de görülmektedir. Artık baş düğüm (HeadNode) üzerinde yapılandırma işlemleri sonrasında hesaplama düğümlerimiz eklenebilecektir.

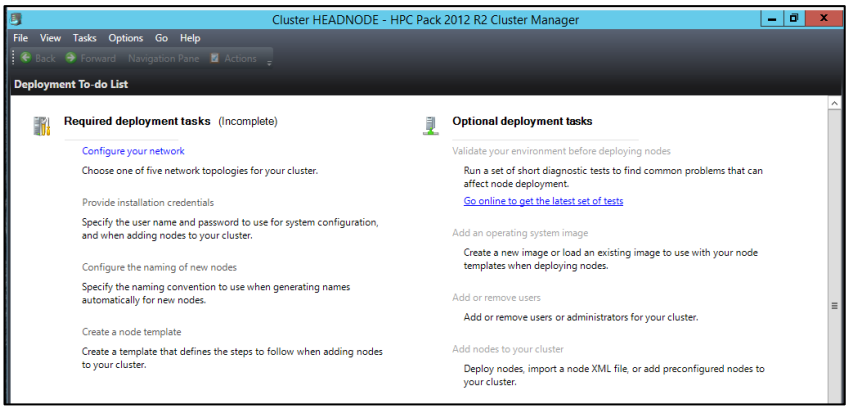


Şekil 2.12. YBH Pack kurulumunun tamamlanması

2.4.2. Kümenin Yapılandırılması

Küme oluşturma işlemlerinden sonra oluşturulan kümenin yapılandırma işlemleri gerçekleştirilecektir.

Bu aşamada sırasıyla küme ağ bağlantılarının yapılandırılması, yönetim ve yapılandırma işlevleri için yetkilerin yapılandırılması, düğümler için isimlendirme ilkesinin belirlenmesi, düğüm oluşturma şablonlarının yapılandırılması işlemleri yapılmalıdır.



Şekil 2.13. YBH Cluster yapılandırma ekranı

Küme yapılandırma işlemlerinin en önemli adımlarından biri olan ağ topolojisi seçiminde düğümlerin yerleşimi belirlenmektedir. Bu adım öncesinde baş düğüm (HeadNode) üzerinde ağ yapılandırmalarının uygun şekilde oluşturulması gerekmektedir. Yapılan ön çalışmada oluşturulan küme için Çizelge **2.4.** de bulunan IP adres yapılandırmaları kullanılmıştır.

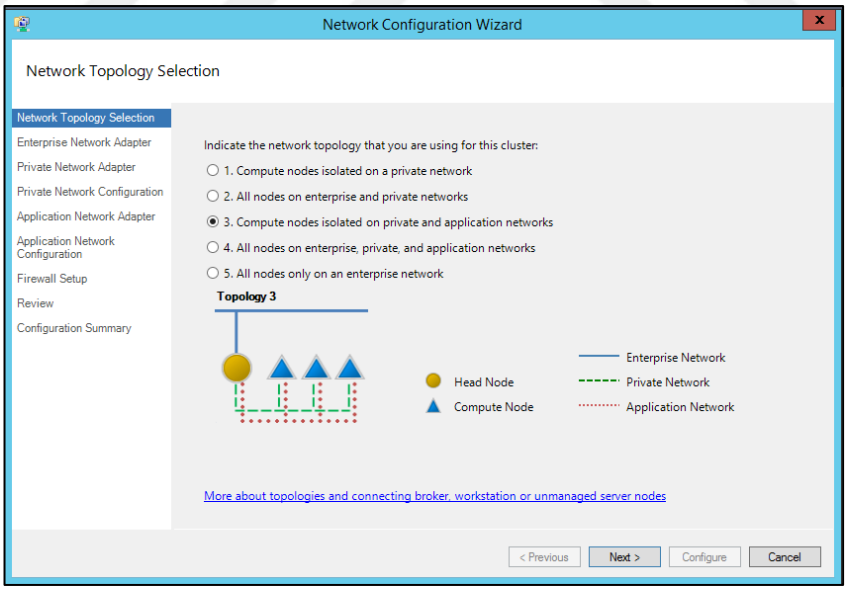
Çizelge 2.4. YBH Kümesi IP adres tablosu

Sunucu Adı	EXT Ağı	INT Ağı	APP Ağı	PRV Ağı
Aktif Dizin Sunucu	192.168.1.6 255.255.255.0	10.10.10.1 255.255.255.0		
Baş düğüm	192.168.1.7 255.255.255.0	10.10.10.2 255.255.255.0	172.16.1.1 255.255.255.0	172.16.10.1 255.255.255.0
Düğüm1			172.16.1.11 255.255.255.0	172.16.10.11 255.255.255.0
Düğüm 2			172.16.1.12 255.255.255.0	172.16.10.12 255.255.255.0
Düğüm 3			172.16.1.13 255.255.255.0	172.16.10.13 255.255.255.0
Düğüm 4			172.16.1.14 255.255.255.0	172.16.10.14 255.255.255.0

Örnek uygulamada Şekil **2.14**. de gösterilen ağ topolojisi benimsenmiştir. Bu topolojide hesaplama düğümleri kurumsal ağdan yalıtılmış durumdadır. Bu yapının sağlayacağı birkaç avantaj bulunmaktadır.

Kurumsal ağdan yalıtılmış düğümler virüs benzeri zararlı yazılımlardan korunmuş olacaktır. Ağ üzerinden gelebilecek saldırılardan uzak tutulacaktır.

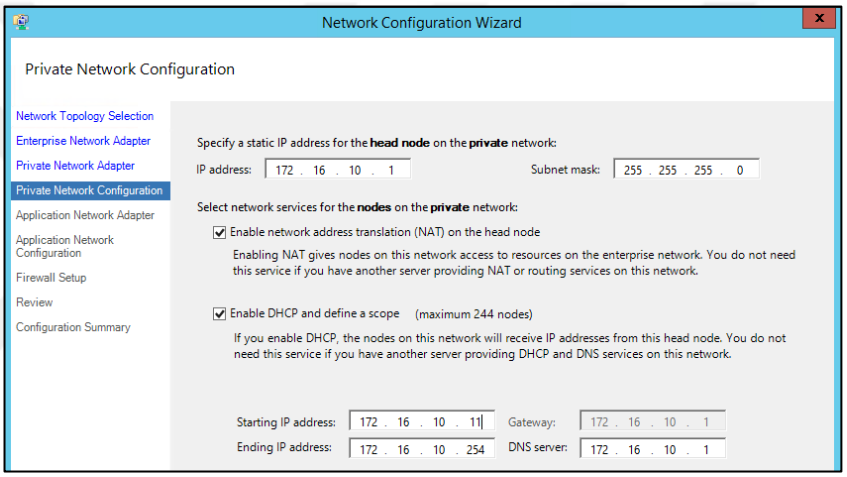
Hesaplama düğümlerinde gönderilen işlerin çabucak bitirilebilmesi için sunucular üzerinde bulunan güvenlik duvarları devre dışı tutulmaktadır. Bu sebeple saldırılara açık durumdadırlar. Bu zafiyetin giderilebilmesi kurumsal ağdan yalıtılması iyi çözüm olarak görülmektedir.



Şekil 2.14. YBH Küme ağı yapılandırma ekranı

Ağ topolojisi seçimi sonrasında baş düğüm üzerinde bulunan ağ kartlarının yapılandırılmasına bağlı olarak seçilen topolojide yer alacak düğümlerin IP adreslerinin otomatik yapılandırılması için birkaç yapılandırma ekranı karşımıza gelecektir.

Bu ekranlarda sırasıyla ağ ara yüzünün adı ve sahip olduğu IP adresi, eğer düğümlere bu ağ ara yüzünden IP adresi dağıtılacaksa IP havuzu ile ilgili yapılandırmalar bulunmaktadır. Şekil **2.15.** te görülmektedir.

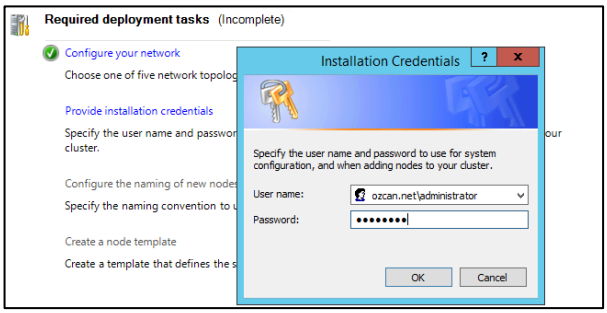


Şekil 2.15. Özel (Private) ağ yapılandırma ekranı

Ağ yapılandırma işlemleri sonrasında hesaplama düğümleri yapılandırılan ağların bulunduğu anahtarlara bağlandığı takdirde ağa uygun IP adreslerini otomatik olarak alacaklar ve ağ iletişimi başlayacaktır.

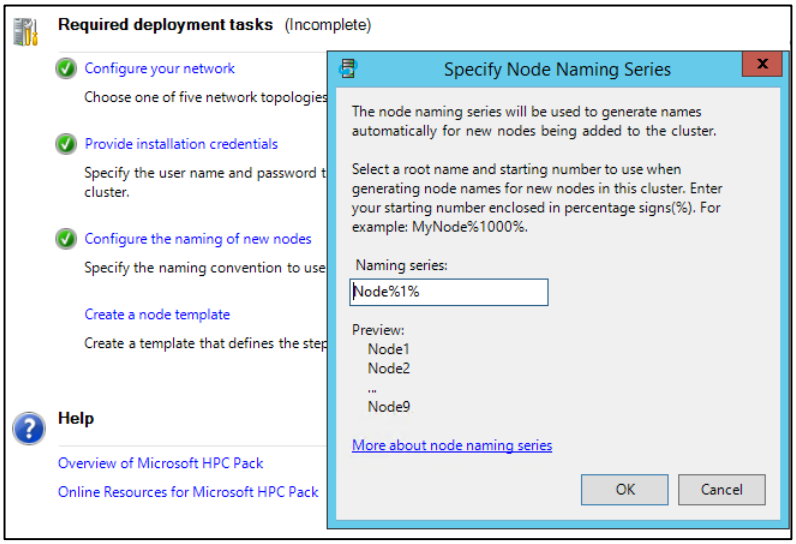
Yapılandırma işleminin bir sonraki adımında ise YBH Cluster yapısının yönetiminin hangi haklarla kim tarafından yapılacağının belirlenmesi

işlemidir. Bu işlemi Aktif dizin etki alanı (Active Directory Domain) yöneticisi haklarıyla yapılacağının belirtmesi uygun olacaktır.



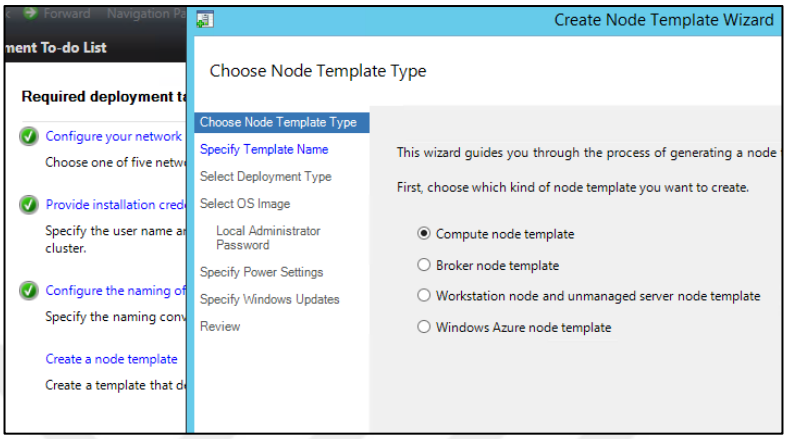
Şekil 2.16. YBH küme yetkilendirme ekranı

Yapılandırmanın sonraki adımında hesaplama sistemine yeni eklenecek düğümler için isimlendirme ilkesi belirlenecektir. Şekil **2.17.** de görülmektedir.



Şekil 2.17. YBH düğüm isimlendirme ekranı

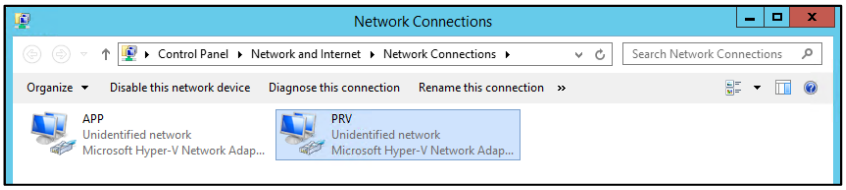
Bir sonraki yapılandırma adımında yeni eklenecek düğümler için şablonlar oluşturulabilmektedir. Hesaplama düğümleri, iş istasyonu düğümleri veya Microsoft Azure bulutu düğümleri için şablonlar oluşturulabilmektedir.



Şekil 2.18. YBH düğüm şablonu oluşturma ekranı

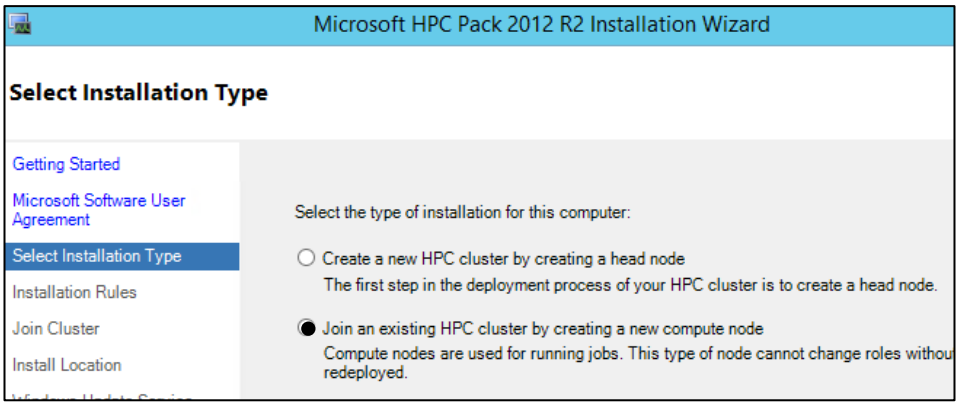
2.4.3. Düğümlerin Oluşturulması ve Yapılandırılması

İşletim sistemi yüklenerek ağa bağlanan hesaplama düğümleri bağlandıkları ağa uygun IP adresleri almış olacaklardır.



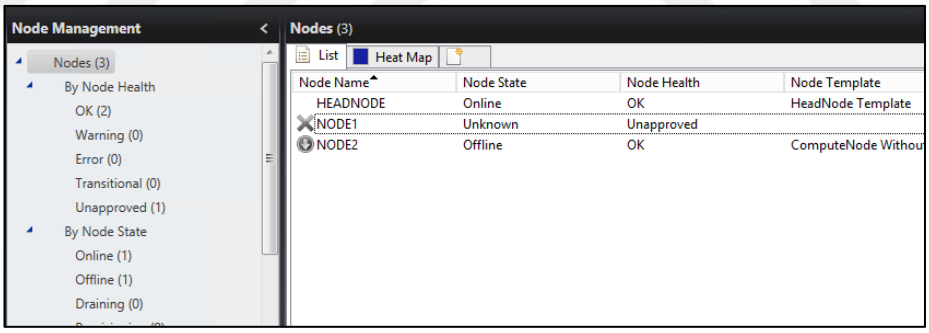
Şekil 2.19. Hesaplama düğümü ağ bağlantıları ekranı

Uygun IP adresleri ile ağa bağlanmış olan hesaplama düğümlerine HPC Pack 2012 yazılımının yüklenmesiyle kümeye katılabilmektedir. Kurulum esnasında kümeye eklenecek sunucuya ait göreve dikkat edilerek gerekli seçim Şekil **2.20.** de görülmektedir.



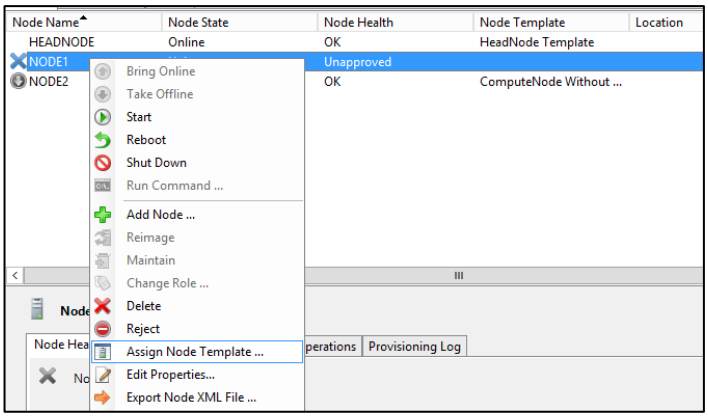
Şekil 2.20. Hesaplama düğümü ekleme ekranı

İşlemin başarıyla sonuçlanabilmesi için düğümlerin önceden Active Directory Domainine katılmış olması gerekmektedir. Hesaplama kümesine katılan düğümler hemen kabul edilmeyerek onaylama sürecinden geçmektedir. Bu aşamada katılacak olan düğüm için onaylanmadı (Unapproved) ifadesi Şekil **2.21.** de görülmektedir.



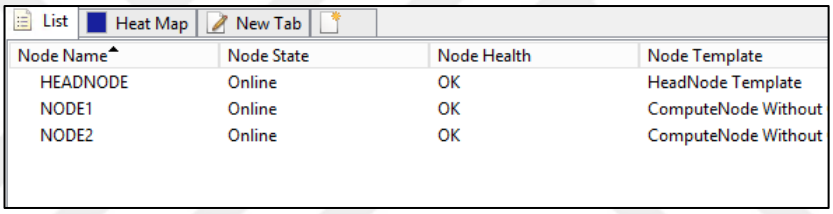
Şekil 2.21. Hesaplama düğümü onay ekranı

Yeni eklenecek hesaplama düğümünün kümede yer alabilmesi için uygun bir düğüm şablonunun uygulanması gereklidir. Bu amaçla eklenen düğüme Şekil **2.22.** de işletim sistemsiz düğüm şablonunu uygulanmaktadır.



Şekil 2.22. Hesaplama düğümü şablon atama ekranı

Düğüm şablonunun yeni eklenen düğüm üzerine uygulanmasıyla, artık bu düğümün kullanılabilir hale geldiği Şekil **2.23.** te görülmektedir.



Şekil 2.23. Hesaplama düğümü hazır ekranı

2.4.4. YBH Küme Konsolu Yönetim İşlemleri

YBH Küme oluşturma işlemlerinden sonra kümenin yönetilmesi, kümeye iş verilmesi için kullanıcıların oluşturulması gerekmektedir. Kullanıcıların seviyelerine göre yönetim yetenekleri bulunmaktadır.

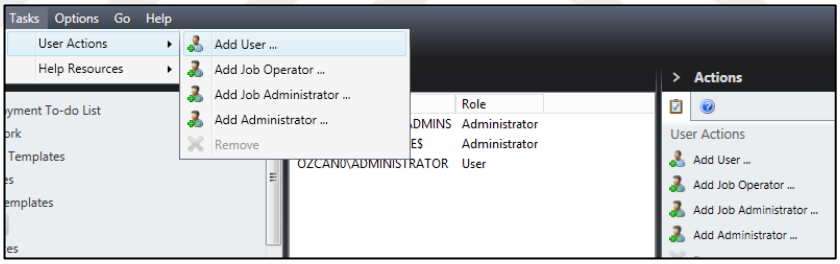
Yönetici (Administrator) kullanıcısı, YBH küme konsolu (HPC Cluster Management Console) yönetiminden sorumludur. Bu konsol aracılığıyla kümeye yeni hesaplama düğümlerinin katılması, düğüm

şablonlarının oluşturulması gibi üst seviye işlemlerin yanında diğer iş planlama işlemleri, Diyagnostik ve rapor işlemlerini yürütmektedir.

Kullanıcı (User) kullanıcısı, sadece iş yönetim konsolu ile YBH kümesine işin gönderilmesi, gönderilen iş üzerinde değişiklerin yapılması yetkilerine sahiptir.

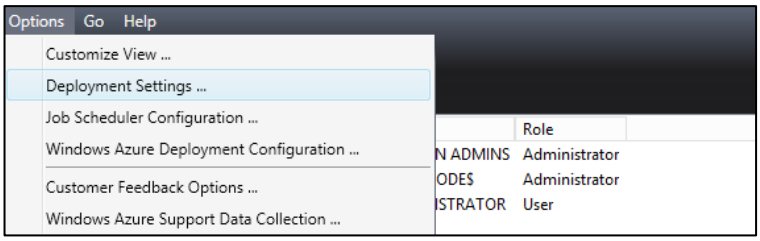
İş uygulama (Job Operator) kullanıcısı, YBH kümesine gönderilen işlerin çalıştırılması, sıralanması işlemlerinde yetkilidir.

İş yönetim (Job Administrator) kullanıcısı, YBH kümesinde İş uygulama kullanıcısının görevleri yanında, yeni işlerin tanımlanması yetkilerine de sahiptir.



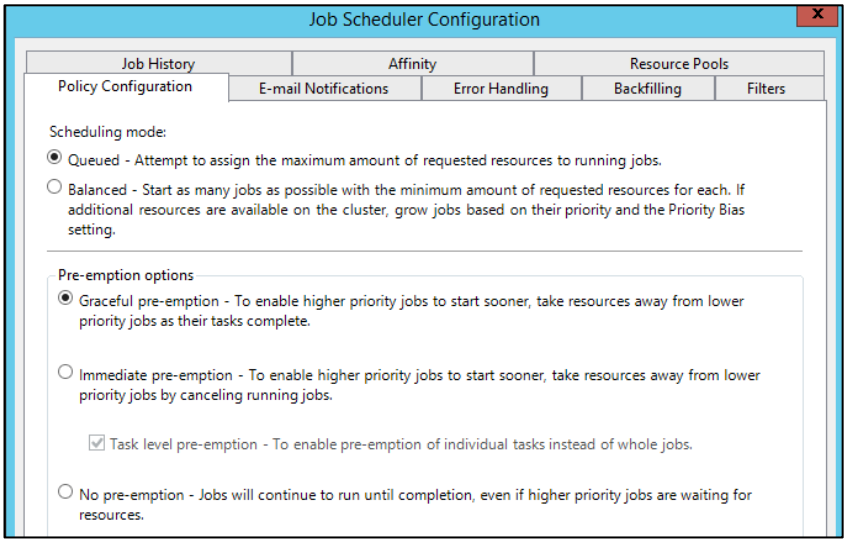
Şekil 2.24. Kullanıcı işlemleri ekranı

YBH konsolunda bir başka yönetim görevi ise yeni eklenecek düğümlerin yapılandırılması işlemidir. Bu işlemler için yayınlama (Deployment) ayarlar (Settings) bölümünden yeni eklenecek düğümler için şablonlar oluşturulabilir. Mevcut düğümlere bu şablonlar uygulanabilir.



Şekil 2.25. Seçenekler ekranı

YBH küme üzerinde önemli yapılandırma bölümlerinden biri de İş takvimi ayarlama (Job Scheduler Configuration) ekranıdır. Bu ekran üzerinden, YBH kümesine gönderilecek paralel işlerin nasıl işleneceği, işlerin sonuçlarının kaç gün hafızada tutulacağı, silme işlemlerinin hangi saatlerde yapılacağı, kaynak havuzunun nasıl kullanılacağı, kuyruğa gönderilecek işlemler için çalışma saatlerinin belirlenmesi, oluşabilecek hata durumlarında işin kaç kez tekrar edilebileceği, kullanıcılara YBH kümesinde iş başlatıldığında ve sonuçlandırıldığında e-posta hizmeti gibi konuların yapılandırılması sağlanmaktadır.



Şekil 2.26. İş takvimi yapılandırma ekranı

2.4.5 YBH Üzerinde Bir İşin Test Edilmesi

YBH kümesinde bir işin test edilebilmesi için öncelikle işlerin koşturulacağı baş düğüm ve diğer tüm düğümlerde işe uygun yazılımların yüklenmesi gereklidir.

Örneğin yazılmış olan bir programın YBH kümesi üzerinde derlenmesi söz konusu ise, baş düğüm ve diğer tüm düğümlerde bu yazılımın yüklenmiş ve doğru yapılandırılmış olması gerekmektedir.

Bu sebeple bir çalışma için bile olsa özel bir yazılım kullanılacak ise bu yazılımın tüm düğümlere yüklenmiş olması gerekecektir. Microsoft HPC Pack 2012 yazılımı ile Microsoft MPI yazılımı, SOA yazlımı yanında Microsoft Excel HPC servisi de yüklenmektedir. YBH kümesine bu yazılımlar yanında başka yazılımlar yüklenebilmektedir.

MPI mesaj aktarma yöntemi ile paralel uygulamaların yazılması bir endüstri standardıdır. Microsoft MPI çalıştırma sistemi ile ele alınan sorunun etki alanının farklı bölümlerini birden fazla düğüm üzerinde çalıştırarak, birden fazla işlem başlatır.

MPI kütüphaneleri uygulamaların iletileri alma, gönderme veya süreçlendirilmesi konusunda birtakım işlevler tanımlar. Kullanıcıların hesaplama düğümlerini belirtmesine gerek duymadan yaptığı MPI başvuruları YBH kümesi sorunsuz şekilde işler.

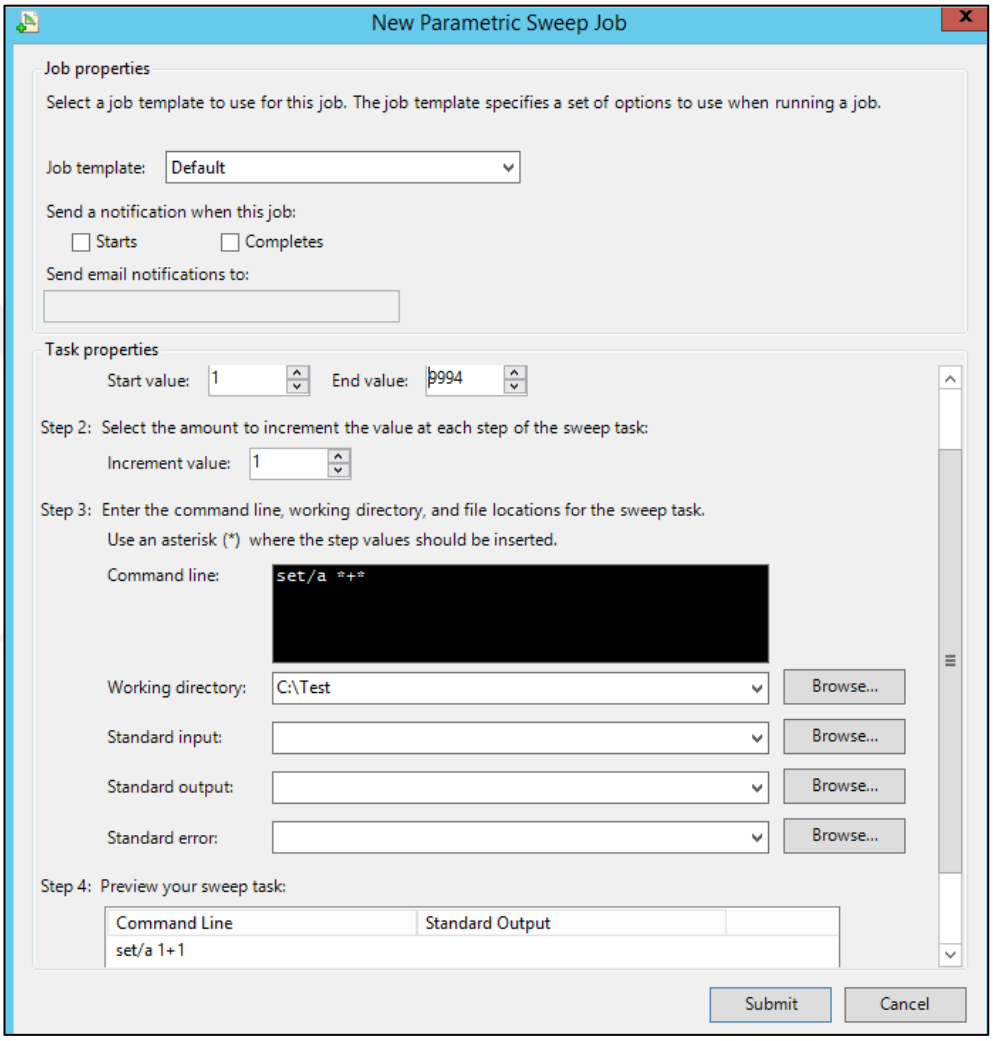
Microsoft Visual Studio yazılımı ile küme üzerinde çalışabilecek MPI uygulamaları geliştirilebilmektedir. Microsoft Visual Studio 2008 ve sonraki sürümlerde C++ dilinde MPI programlama yapılabilmektedir.

YBH kümesinde SOA kullanımı yazılım mimarilerinde yaygın olarak kullanılmaktadır. Microsoft YBH kümesi .NET Framework' e dayalı SOA tabanlı Windows Communication Foundation (WCF) uygulamalarını desteklemektedir. Microsoft Visual Studio programı WCF proje şablonları içerdiğinden yazılım geliştirme, yapılandırma, kullanım, hata ayıklama oldukça basitleştirilmektedir.

Programcılar YBH kümesinde SOA modeliyle yapacağı yazılımlarla düğümlerde bulunan servisleri paralel olarak çalıştırabilmektedir. Böylece farklı düğümlerde bulunan servisler, düğümlerdeki hesaplama birimlerini etkin olarak kullanabilmektedir. Bunun yanında uygulamaların kümedeki fiziksel yapıyı bilmesine gerek kalmadan bu işlem kolayca gerçekleştirilebilmektedir.

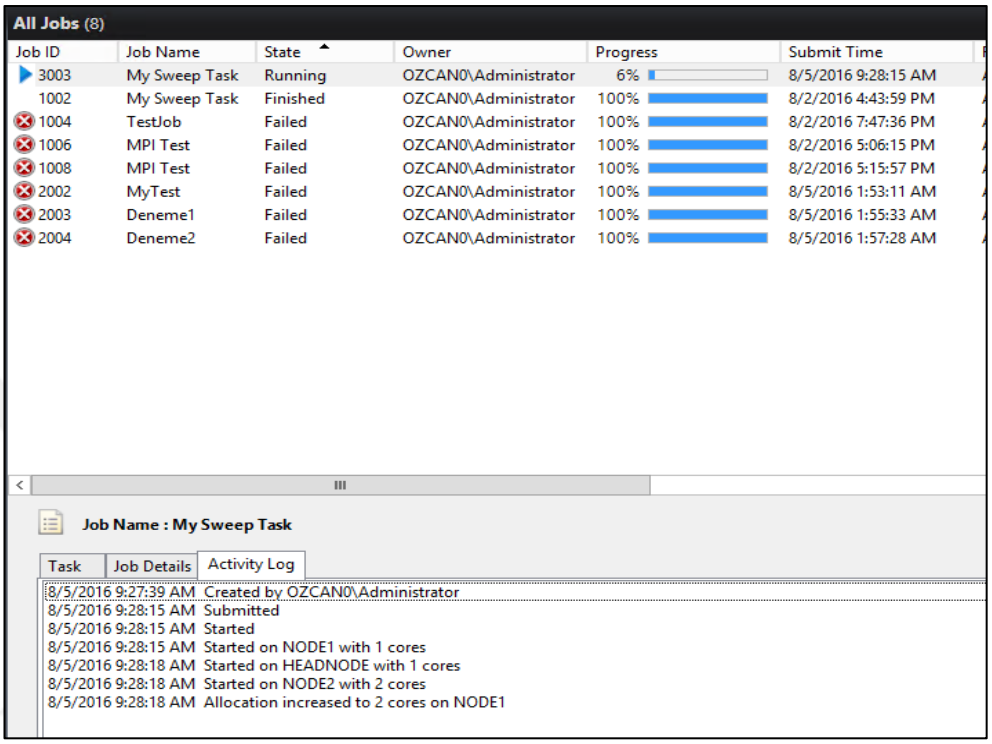
Günümüzde milyonlarca kullanıcı bir ofis hesaplama programı olan Excel' i kullanmaktadır. Excel programı istatistiksel analiz fonksiyonları da içeren bir programdır. Excel üzerinde istatiksel analizlerin kısa sürede tamamlanabilmesi için HPC Pack 2012 programının Excel desteği de bulunmaktadır. Böylece Excel de uzun süren hesaplama işlemleri YBH kümesi üzerinde koşturularak kısa sürede tamamlanabilmektedir.

Çalışma yapılan YBH kümesinin test edilmesi için basit bir parametrik iş tanımı yapılabilmektedir. Şekil **2.27.** de görülmekte olan ekran üzerinde yapılacak tanımlamalara ve belirlenen parametreye göre düğümler üzerinde bir hesaplama işlemi başlatılabilecektir.



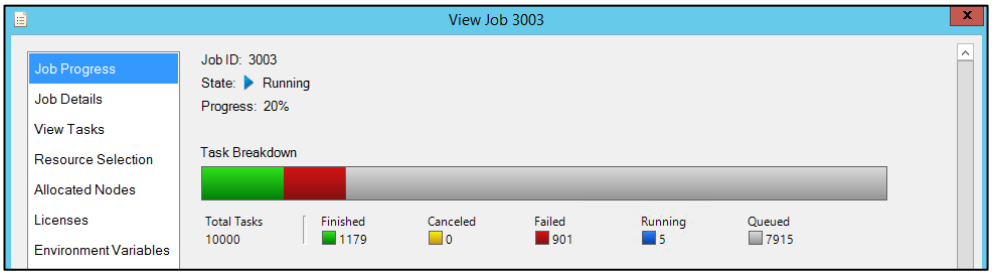
Şekil 2.27. Parametric Sweep Job gönderme ekranı

Kuyruğa alınarak başlatılan iş ekranda görüntülenecektir. Bu ekranda çalışmakta olan işin ilerleme durumu, işin kaç düğüm ve kaç çekirdek tarafından yapıldığı Şekil **2.28.** de görülmektedir.



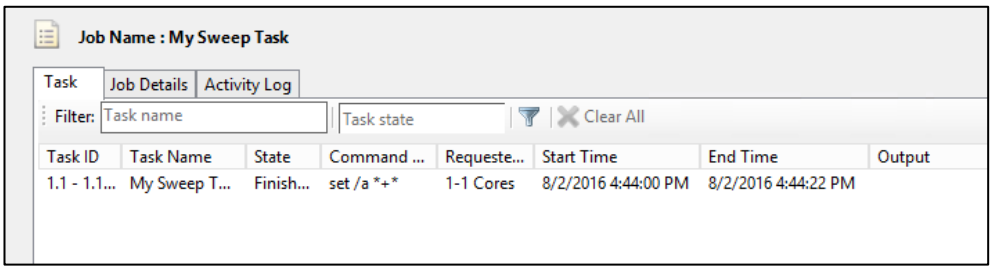
Şekil 2.28. Çalışmakta olan işin gerçekleşme oranı

Çalışmakta olan işin işleyişi canlı olarak izlenebilmekte, işin yapılması esnasında ortaya çıkan hatalar Şekil **2.29.** da anlık olarak izlenebilmektedir.



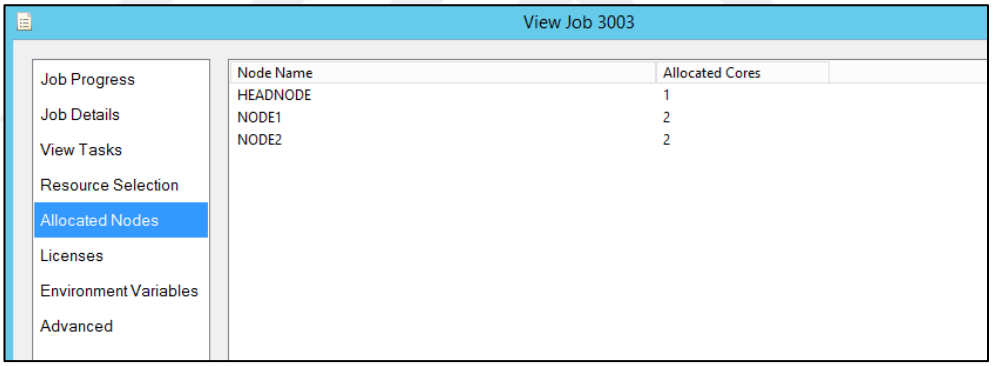
Şekil 2.29. İş izleme ekranı

Tamamlanan işlerin detayları Şekil **2.30.** da görülmektedir.



Şekil 2.30. İş planlama detay ekranı

Tamamlanan bir işin hangi düğümler tarafından gerçekleştirildiği ve kaç işlemci çekirdeği kullanıldığı işin detaylarının gösterildiği Şekil **2.31.** de görülmektedir.



Şekil 2.31. İş planlama düğüm detay ekranı

3. ARAŞTIRMA BULGULARI VE TARTIŞMA

Yüksek başarımlı hesaplama sistemlerinin genel kullanımına bakıldığında, uygulamaların hızlıca sonuçlandırılması veya tek bir sunucu tarafından baş edilemez sorunların çözümünde kullanıldığı görülmektedir.

Yüksek başarımlı hesaplama sistemleri (HPC) zorunlu olmamakla birlikte tercihen aynı donanım ve yazılım özellikleri taşıyan sunucu sistemlerinin paralel çalıştırılmasıyla oluşturulmaktadır. Günümüzde YBH kümelerinde yoğun olarak Linux türevi işletim sistemlerinin yanında Microsoft Windows tabanlı işletim sistemleri kullanılmaktadır.

3.1. YBH Mimarisi

Gerçekleştirilen çalışmada en bilindik küme mimarisi benimsenmiştir. Bu mimaride dışarıdan yalıtılmış hesaplama düğümlerinin dış dünya veya kurumsal ağ ile iletişimlerini sağlayan bir sunucu (Baş düğüm-Headnode – Masternode) bulunmaktadır. Yapıda baş düğüm sunucu düğümlerde hesaplatılacak işlerin yönetiminden sorumlu olmaktadır.

Çalışmada 10 adet masaüstü bileşenlerinden oluşan bilgisayar sistemi kullanılmıştır (Bkz.Ek-2). Bu bilgisayarlardan bir tanesi Active Directory Domain Controller olarak bir tanesi de baş düğüm olarak yapılandırılmıştır. Gerçekleştirilen sistemde sekiz adet bilgisayar hesaplama düğümü (Compute Node) olarak yapılandırılmıştır.

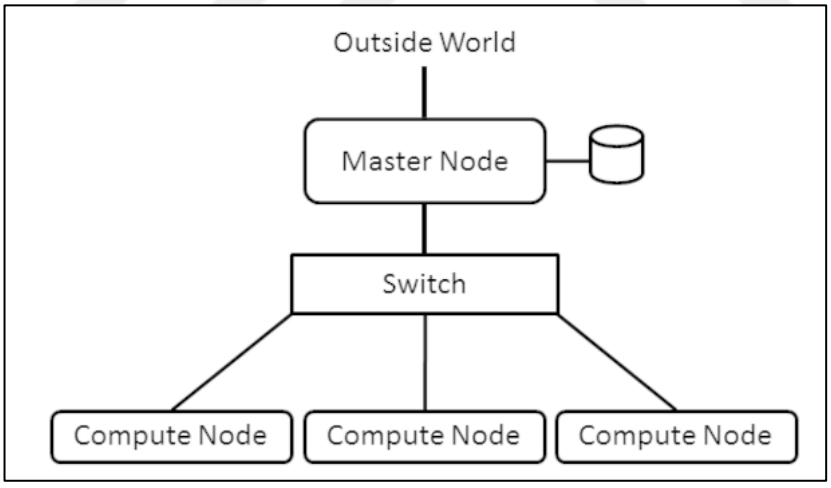
Çalıştığımız yüksek başarımlı hesaplama sisteminin teorik performansı hesaplandığında dünyadaki örneklerine göre oldukça düşük hıza sahip olduğu söylenebilir.

Hız (Gflops) = node(sockets/node)*(cores/sockets)*Core Speed* Core Flops*

*= 8 * (1) * (8) * 4.00 * 8*

= 2048 Gflops → 2 Tflops

2016 yılının Haziran ayında yapılan dünya sıralamasında 1. olan bilgisayarın 125.436 Tflops hızında olduğu düşünülürse kurulan sistemin giriş seviyesi olduğu görülmektedir.



Şekil 3.1. Basit YBH mimarisi [35]

3.2. YBH İşletim Sistemi ve diğer yazılımlar

YBH sisteminin oluşturulmasında kullanılacak işletim sistemi seçiminde, sistemi kullanacak kullanıcıların bu konudaki tecrübeleri ve deneyimleri etkili olmaktadır. Yapılan çalışmada oluşturulacak YBH kümesini çalıştıracak işletim sistemi olarak Microsoft Windows Server 2012 R2 işletim sistemi tercih edilmiştir.

İşletim sisteminin araştırmacı tarafından önceden kullanılmış olması, grafik ara yüze sahip olması oluşturulan YBH sisteminin kurulum aşamasında karşılaşılan problemlerin çözümünü kolaylaştırmıştır.

Active Directory Domain Controller sunucusu dışındaki tüm bilgisayarlara HPC Pack 2012R2 yazılım paketi ve Microsoft Visual Studio 2010 yazılımı yüklenmiştir. Çalışma sırasında yüklemelerin başarıyla gerçekleşmesi için yükleme öncesi işletim sistemi güncellemelerinin eksiksiz yapılması gerekli olduğu görülmüştür.

Ayrıca ağ yapılandırmaları sırasında IPv6 ayarlarının devre dışı bırakılması ve WINS yapılandırmalarının geçersiz kılınması gerektiği anlaşılmıştır.

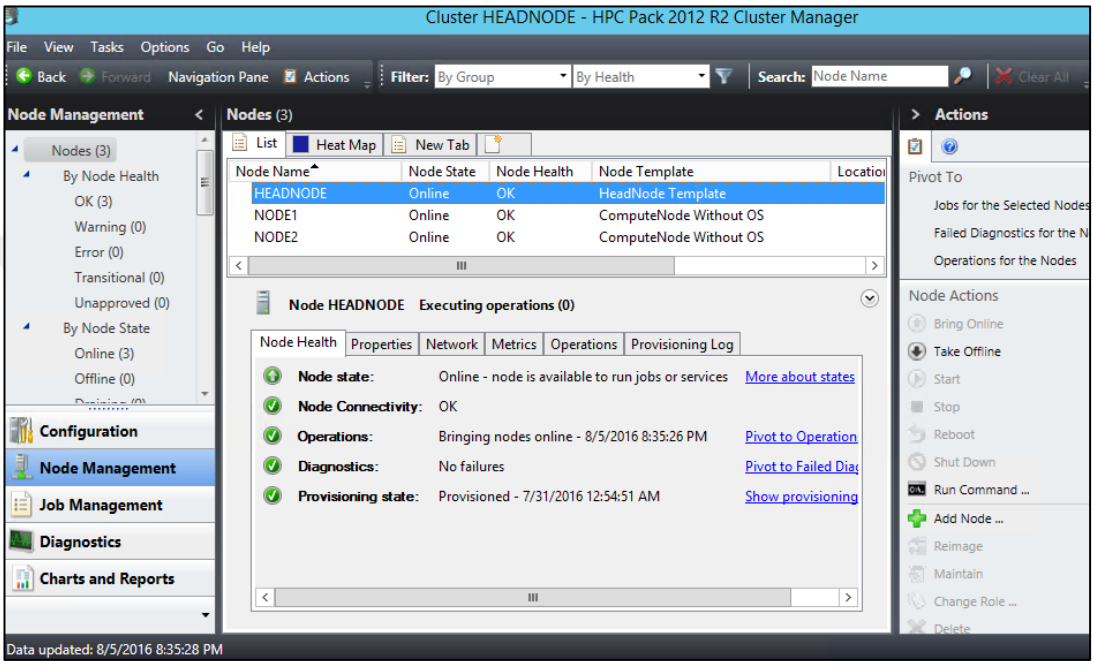
Çizelge 3.1. Çalışmada kullanılan IP adres tablosu

Sunucu Adı	EXT Ağı	INT Ağı	APP Ağı	PRV Ağı
DC	192.168.1.6 255.255.255.0	10.10.10.1 255.255.255.0		
Baş Düğüm	192.168.1.7 255.255.255.0	10.10.10.2 255.255.255.0	172.16.1.1 255.255.255.0	172.16.10.1 255.255.255.0
Düğüm1			172.16.1.11 255.255.255.0	172.16.10.11 255.255.255.0
Düğüm 2			172.16.1.12 255.255.255.0	172.16.10.12 255.255.255.0
Düğüm 3			172.16.1.13 255.255.255.0	172.16.10.13 255.255.255.0
Düğüm 4			172.16.1.14 255.255.255.0	172.16.10.14 255.255.255.0
Düğüm 5			172.16.1.15 255.255.255.0	172.16.10.15 255.255.255.0
Düğüm 6			172.16.1.16 255.255.255.0	172.16.10.16 255.255.255.0
Düğüm 7			172.16.1.17 255.255.255.0	172.16.10.17 255.255.255.0
Düğüm 8			172.16.1.18 255.255.255.0	172.16.10.18 255.255.255.0

3.3. YBH Yönetim Konsolu

HPC Pack 2012R2 yazılım paketlerinin yüklenmesiyle kümeye katılan hesaplama düğümleri gerekli düğüm şablonları kullanılarak sisteme entegrasyonu sağlanmaktadır. Bu işlem sonrasında düğümlerin durumu Online yapılarak YBH kümesi hazır hale gelmektedir. Bu aşama sonrasında sistemin tüm bileşenlerinin eksiksiz çalıştığından emin olunması için

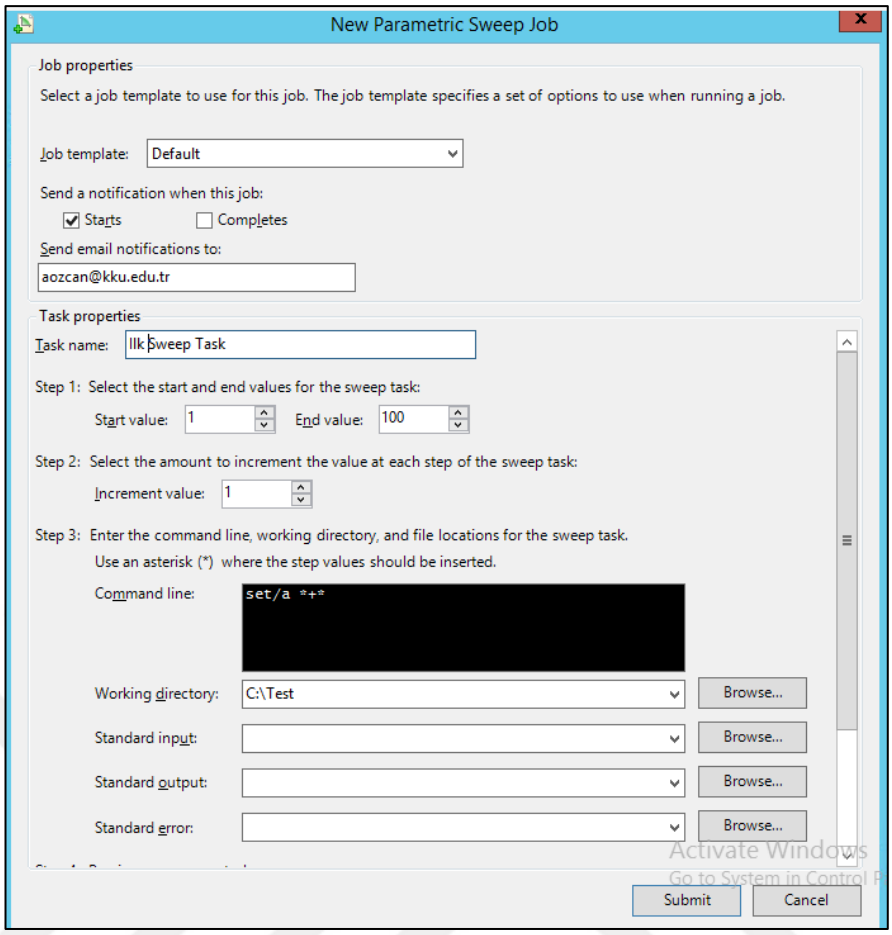
diyagnostik testleri yapılmalıdır. Ancak bu aşama hatasız tamamlanırsa hesaplama düğümlerine iş gönderilebilecektir. İşlem başarıyla tamamlandığında Düğüm listesinde sağlık durumları OK olarak görüntülenecektir.



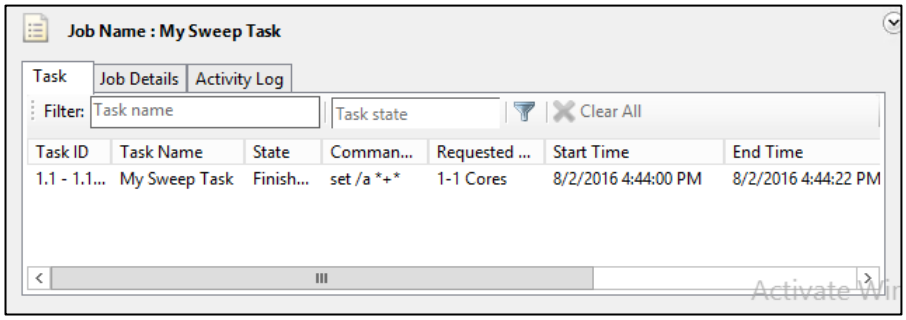
Şekil 3.2. YBH Yönetim konsolu

3.4. YBH Küme testi

Çalışma için hazırlanan yüksek başarımlı hesaplama sistemi kümesindeki bilgisayarları test edebilmek için basit bir parametrik test uygulanarak sistemin performansı kontrol edilebilmektedir. Bu aşamada birden yüze kadar sayılar ardışık olarak toplatılmaktadır. Buna uygun bir parametre yazarak işlem başlatıldığında Şekil **3.4.** te görülen sonuç ekranı alınmaktadır.

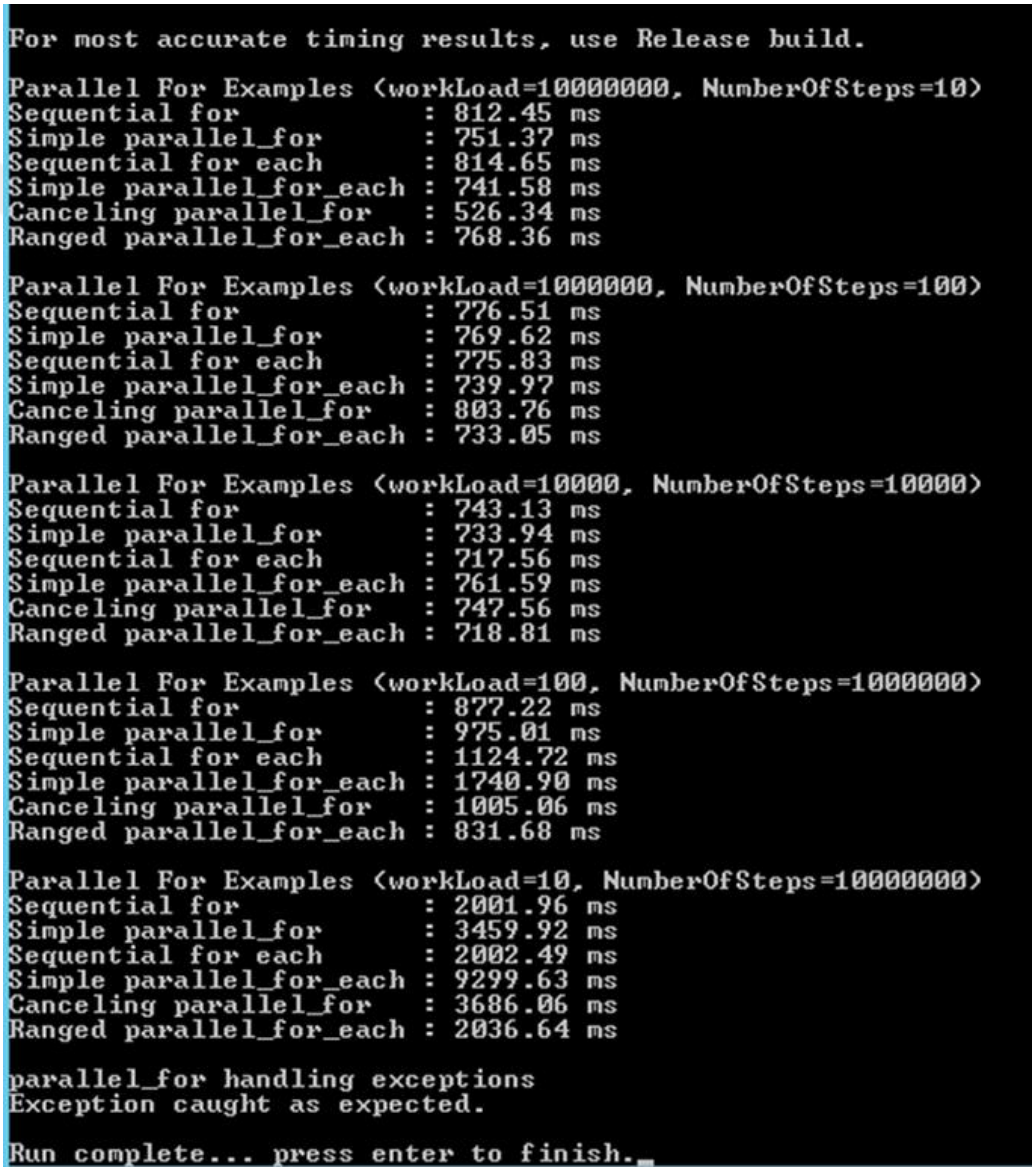


Şekil 3.3. Basit parametrik iş tanımlama ekranı



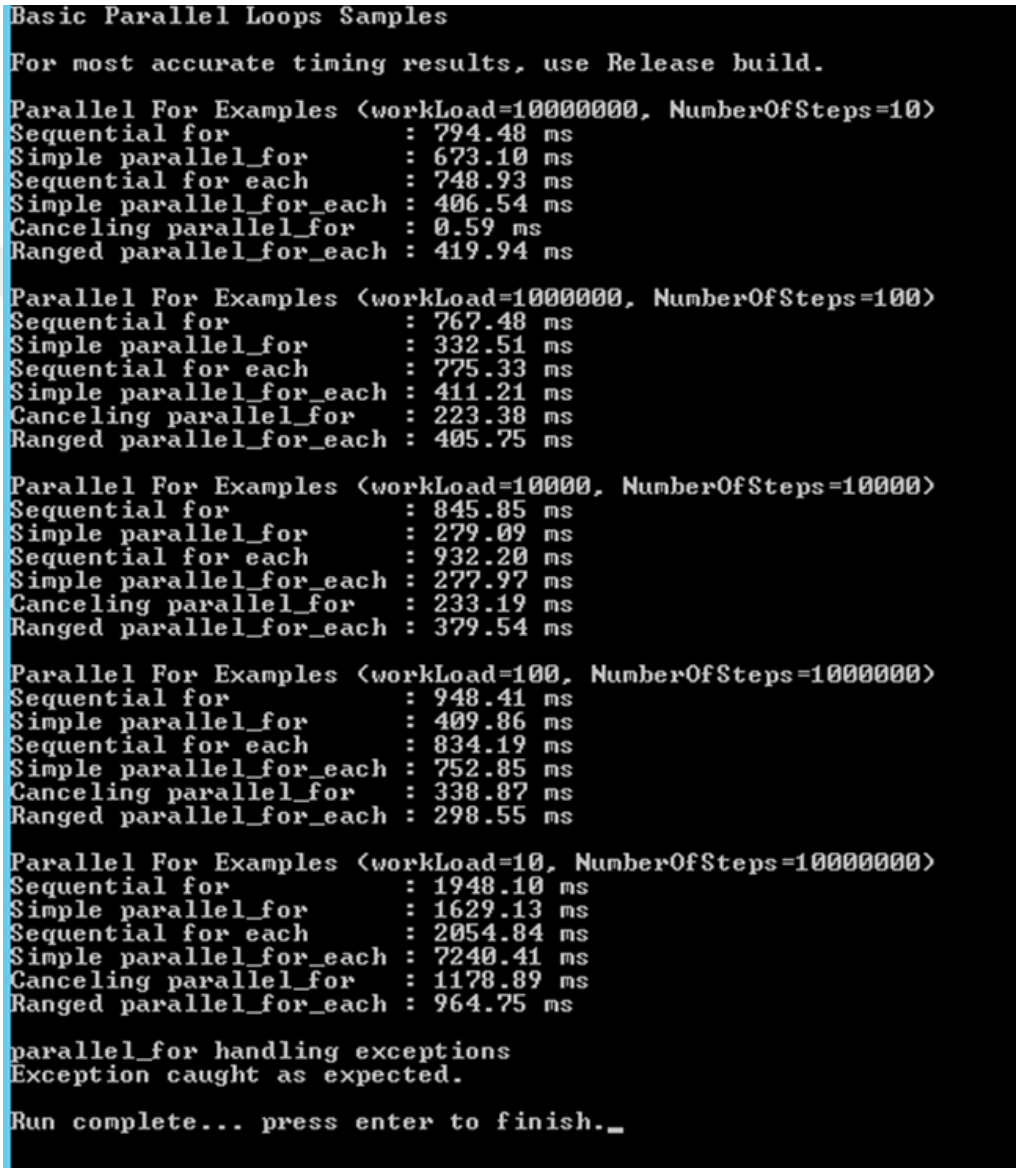
Şekil 3.4. Basit parametrik iş sonuç ekranı

Çalışmada YBH kümesinin testi için C++ dilinde paralel olarak yazılmış olan döngülerden oluşan programla sistemin testi gerçekleştirilmiştir [36]. Programın (Bkz.Ek-1) YBH kümesinde tek çekirdek üzerinde koşturulmasıyla elde edilen sonuçlar Şekil **3.5.** te görülmektedir.



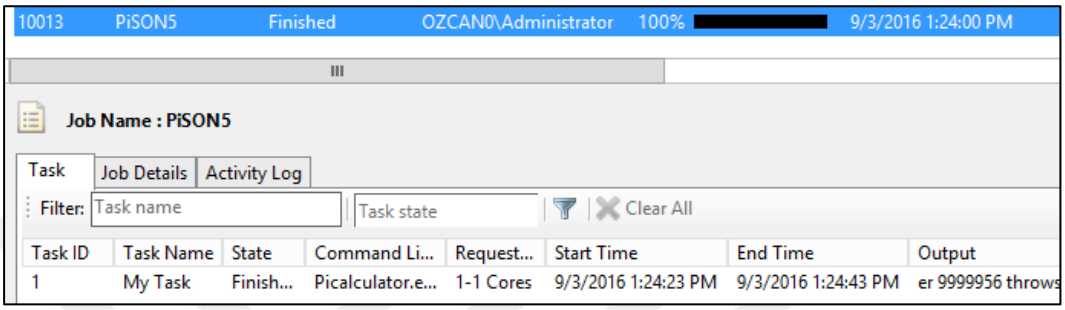
Şekil 3.5. Tek çekirdekte çalıştırılan işin sonuç ekranı

Programımızın YBH kümesinde üç çekirdekten oluşan düğümler üzerinde koşturulmasıyla elde edilen sonuç ekranı ise Şekil **3.6.** da görülmektedir.



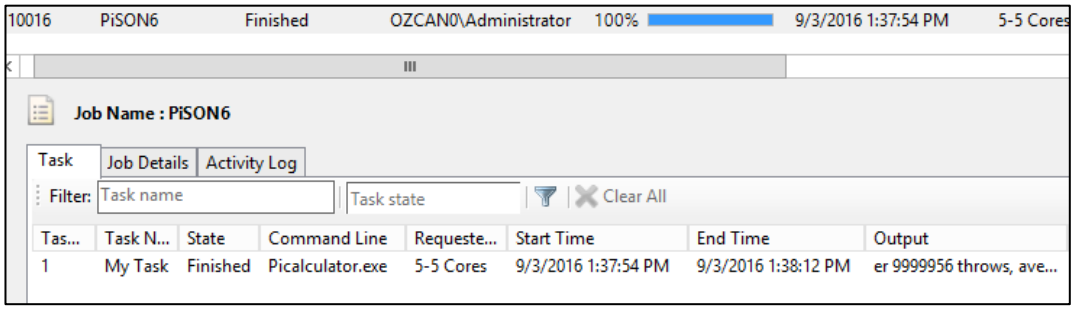
Şekil 3.6. Üç çekirdekle çalıştırılan işin sonuç ekranı

Yapılan çalışmada ayrıca Mesaj Geçiş Arayüzü (MGA) paralel hesaplama yöntemi kullanılarak, MonteCarlo metoduyla Pi sayısının hesaplaması gerçekleştirilmiştir [37]. Kullanılan programda iterasyon sayısı on milyon olarak belirlenmiştir (Bkz.Ek-3). Program önce tek çekirdekli bir düğüm üzerinde koşturulmuştur. Alınan sonuç Şekil **3.7.** de görülmektedir.



Şekil 3.7. Tek çekirdekle çalıştırılan MonteCarlo metodu

Program sonra beş çekirdekten oluşan sistem üzerinde koşturulduğunda ise Şekil **3.8.** de görülen sonuç alınmıştır.



Şekil 3.8. Beş çekirdekle çalıştırılan MonteCarlo metodu

Çalışma karşılaştırıldığında iki saniyelik bir zaman farkı görülmektedir. Uygulamanın çalışma süresi arttıkça aradaki farkın artabileceği görülmektedir.

4. SONUÇLAR

Gerçekleştirilen çalışmada akademik ortamlarda henüz yeni yaygınlaşan yüksek başarımlı hesaplama sistemlerinin mevcut olan masaüstü bilgisayar bileşenleriyle oluşturulan sistemler üzerine kurulumu gerçekleştirilmiştir.

Yüksek başarımlı hesaplama sistemleri, tüm bileşenleri özenle seçilmiş sunucu bilgisayarlar üzerine çok yüksek maliyetler ile kurulmaktadır. Çalışma ile düşük bir bütçe ile giriş seviyesi bir yüksek başarımlı hesaplama sisteminin oluşturulabileceği görülmektedir.

Çalışmada işletim sistemi ve uygulama yazılımlarının deneme sürümleri kullanıldığı için bu maliyetler göz ardı edilmiştir. Eğer sürekli çalışan bir hesaplama sistemi kurulumu gerçekleştirilecek ise yazılım masrafları maliyetleri artırabilecektir.

Çalışmanın başka olumlu yönü ise, bilgisayar yazılımı alanında paralel programların çalıştırılabileceği bir platformun oluşturulmuş olmasıdır. Bu platform ile farklı bilim dallarından araştırmacıların paralel kodlarla yazmış oldukları araştırma projeleri sistem üzerinde çalıştırılarak kısa zamanda sonuçlar alınabilmektedir.

Gerçekleştirilen çalışma henüz giriş seviyesi bir çalışma olduğundan ilerleyen süreçte yeni çalışmaların yapılmasına öncü olacaktır. Mevcut çalışmada kullanılan Microsoft HPC Pack2012R2 kullanılmıştır.

Gelecekte yapılacak YBH çalışmalarında Linux türevleri ile oluşturulabilecek küme bilgisayarları oluşturulabilecektir. Böylece oluşacak yapı ulusal TR-Grid yapısına entegre olabilir duruma gelecektir.

Yapılan bu çalışma, bu konuda gerçekleştirilecek akademik faaliyetlere ışık tutması için bir başlangıç adımı olarak görülebilir.

KAYNAKLAR

[1] Zack, B., Hpc Cluster, Microsoft, http://blogs.msdn.microsoft.com, (Erişim tarihi:21.04.2016)

[2] Anonim, Yüksek Başarımlı Hesaplama, Wikipedia, https://tr.wikipedia.org/wiki/Yüksek_başarımlı_hesaplama, (Erişim tarihi: 14.3.2016)

[3] J.J. Dongarra, A.J.v.d.S., High Performance Computing Systems: Status and Outlook. Acta Numerica, Cambridge University Press. 91, 2012.

[4] Lathrop, S., Murphy, T., High-Performance Computing Education. Computing in Science & Engineering. 10, 9-11, 2008.

[5] Govind, N., Janssen, C.L., Veryazov, V.V., Kowalski, K., Lindh, R., De Jong, W.A.B.E.J., van Dam, H.J.J., Muller, T., Nielsen, I., Institutionen för fysikalisk och analytisk, k., Kvantkemi, Uppsala, u., Kemiska, s., Teknisk-naturvetenskapliga, v., Utilizing High Performance Computing for Chemistry: Parallel Computational Chemistry. Physical Chemistry Chemical Physics. 12, 6896-6692, 2010.

[6] Kindratenko, V., Trancoso, P., Trends in High-Performance Computing. Computing in Science & Engineering. 13, 92-95, 2011.

[7] AbdelBaky, M., Parashar, M., Kim, H., Jordan, K.E., Sachdeva, V., Sexton, J., Jamjoom, H., Shae, Z.-Y., Pencheva, G., Tavakoli, R., Wheeler, M.F., Enabling High-Performance Computing as a Service. Computer. 45, 72-80, 2012.

[8] Shi, L., Chen, H., Sun, J., Li, K., Vcuda: Gpu-Accelerated High-Performance Computing in Virtual Machines. IEEE Transactions on Computers. 61, 804-816, 2012.

[9] Beaty, D.L., High Performance Computing Data Centers. ASHRAE JOURNAL. 55, 142-144, 2013.

[10] André, J.-C., Aloisio, G., Biercamp, J., Budich, R., Joussaume, S., Lawrence, B., Valcke, S., High-Performance Computing for Climate Modeling. Bulletin of the American Meteorological Society. 95, ES97-ES100, 2014.

[11] Jia, X., Ziegenhein, P., Jiang, S.B., Gpu-Based High-Performance Computing for Radiation Therapy. Physics In Medicine And Biology. 59, 151-182, 2014.

[12] Herault, T., Robert, Y., Fault-Tolerance Techniques for High-Performance Computing. Springer Verlag, New York, 2015.

[13] Hack, J.J., Papka, M.E., Big Data: Next-Generation Machines for Big Science. Computing in Science & Engineering. 17, 63-65, 2015.

[14] Saeed, M., Ali, S.A., Feroze, M., Touheed, N., High Performance Computing Achieved in Personal Computers. International Journal of Computer Science Issues (IJCSI). 12, 57, 2015.

[15] Tripathy, M., Tripathy, C., A Comparative Analysis of Some High Performance Computing Technologies. COMPUSOFT: International Journal of Advanced Computer Technology. 3, 2015.

[16] Milojicic, D., High Performance Computing (HPC) in the Cloud, Computing | Now, (Erişim tarihi: 16.05.2016)

[17] Anonim. TRUBA, http://www.truba.gov.tr/, (Erişim tarihi: 16.05.2016)

[18] Anonim. Parallel Computing, https://en.wikipedia.org/wiki/Parallel _computing (Erişim tarihi: 19.05.2016)

[19] Anonim. Understanding Node Roles in Microsoft Hpc Pack, Microsoft, https://technet.microsoft.com/en-us/library /ff919409 (v= ws.11).aspx, (Erişim tarihi: 22.07.2016)

[20] Minkenberg, C., Interconnection Network Architectures for High-Performance Computing, http://www.systems.ethz.ch/sites /default/files/file/Spring2013_Courses/AdvCompNetw_Spring2013/13-hpc.pdf, (Erişim tarihi: 24.07.2016)

[21] Anonim. Infiniband, Wikipedia, https://en.wikipedia.org/wiki/InfiniBand, (Erişim tarihi: 24.07.2016)

[22] Anonim. Fat Tree, Wikipedia, https://en.wikipedia.org/wiki/Fat_tree (Erişim tarihi: 24.07.2016)

[23] Anonim. 3d Torus, Wikipedia, https://en.wikipedia.org/wiki /Torus_ interconnect, (Erişim tarihi: 24.07.2016)

[24] Anonim. Cluster Topologies - Dragonfly, Hpc-Opinion, http://hpc-opinion.blogspot.com.tr/2014/08/cluster-topologies-dragonfly.html, (Erişim tarihi: 24.07.2016)

[25] Anonim. High Performance Computing Solutions, EMC, http://www.emc.com/storage/ high-performance-computing.htm, (Erişim tarihi: 24.07.2016)

[26] Pel, V., Energy Efficiency Aspects in Cray Supercomputers, ENA-HPC, http://www.ena-hpc.org/2010/talks/ EnA-HPC2010-Pel-Energy_ Efficiency_Ascpects_in_Cray_Supercomputers.pdf, (Erişim tarihi: 24.07.2016)

[27] Anonim. Parallel Computing, Wikipedia, http://en.wikipedia.org//wiki /Parallel_ computing, (Erişim tarihi: 21.07.2016)

[28] Anonim. Comparison O Cluster Software, Wikipedia, https://en.wikipedia.org/wiki/Comparison_of_cluster_software, (Erişim tarihi: 24.07.2016)

[29] Saeed Iqbal, R.G., Yung-Chin Fang, Planning Considerations for Job Scheduling in Hpc Clusters, Dell, http://www.dell.com/downloads/global/power/ps1q05-20040135-fang.pdf, (Erişim tarihi: 24.07.2016)

[30] Anonim. List of Job Scheduler Software, Wikipedia, https://en.wikipedia.org/wiki/List_of_job_scheduler_software, (Erişim tarihi: 28.07.2016)

[31] Erdal E., Erguzen A., Ozcan A., A Parallel Approach for Determining Region–of–Interest Area of Digital Medical Images, The IRES -28th International conferences on Engineering and Natural Science (ICENS), 2015.

[32] Anonim. Uygulama Yazılımları, İTU, http://www.uhem.itu.edu.tr /index.php/yazilim/, (Erişim tarihi: 03.08.2016)

[33] Anonim, Cluster Partner, Silicon Mechanics, http://www.siliconmechanics.com/i51427/microsoft-hpc-pack-2012, (Erişim tarihi: 30.07.2016)

[34] Anonim. Windows Cluster, ithome.com.tw, http://ithelp.ithome.com.tw/questions/10034417, (Erişim tarihi: 31.07.2016)

[35] Anonim. HPC, http://www.admin-magazine.com/HPC, (Erişim tarihi: 03.08.2016)

[36] Campbell C., Johnson R., Miller A., Toub S., Parallel Programming with Microsoft .NET, http://parallelpatterns.codeplex.com, (Erişim tarihi: 10.08.2016)

[37] Anonim. UoB-HPC-Examples-2015, https://github.com/ UoB-HPC/UoB-HPC-Examples-2015/blob/master/mpi/examples/ example3/dartboard_pi_send.c, (Erişim tarihi: 10.08.2016)

EKLER

EK-1. YBH Kümesini test için kullanılan **BasicParallelLoops** programı c++ kodları

```
//====================================================================
// Microsoft patterns & practices
// Parallel Programming Guide
//====================================================================
// Copyright © Microsoft Corporation.  All rights reserved.
// This code released under the terms of the
// Microsoft patterns & practices license
// (http://parallelpatterns.codeplex.com/license).
//====================================================================

#include <exception>
#include <string>
#include <iosfwd>
#include <ppl.h>
#include <concrt.h>
#include <concrtrm.h>
#include <math.h>
#include <vector>
#include <algorithm>

#include "SampleUtilities.h"

volatile bool g_SimulateInternalError;

using namespace ::std;
using namespace ::Concurrency;
using namespace ::SampleUtilities;

class ParallelForExampleException : exception {};
class InvalidValueFoundException : exception {};

#pragma region Worker methods

double round(double value, int decimals)
{
    double exp = pow(double(10.0), decimals);
    return floor(value * exp) / exp;
}

double DoWork(int i, int workLoad)
{
    double result = 0;
    for (int j = 1; j < workLoad + 1; ++j)
    {
        double j2 = (double)j;
        double i2 = (double)i;
```

```
        result += sqrt(((double)9.0 * i2 * i2 + (double)16.0 * i * i) * j2
* j2);
    }


// Simulate unexpected condition in loop body

    if ((i % 402030 == 2029) && g_SimulateInternalError)
        throw ParallelForExampleException();

    return round(result, 1);
}

double ExpectedResult(int i, int workLoad)
{
    return (double)2.5 * (workLoad + 1) * workLoad * i;
}

void VerifyResult(const vector<double>& values, int workLoad)
{
    for (unsigned int i = 0; i < values.size(); ++i)
    {
        if (values[i] != ExpectedResult(i, workLoad))
            throw InvalidValueFoundException();
    }
}

#pragma endregion


// Sequential for loop

void Example01(vector<double>& results, int workLoad)
{
    size_t n = results.size();
    for (size_t i = 0; i < n; ++i)
    {
        results[i] = DoWork(i, workLoad);
    }
}


// Parallel for loop

void Example02(vector<double>& results, int workLoad)
{
    size_t n = results.size();
    parallel_for(0u, n, [&results, workLoad](size_t i)
    {
        results[i] = DoWork(i, workLoad);
    });
}


// Sequential for each loop

void Example03(size_t size, int workLoad)
{
```

```
// Create input values

    vector<size_t> inputs(size);
    for (size_t i = 0; i < size; ++i)
        inputs[i] = i;

    for_each(inputs.cbegin(), inputs.cend(), [workLoad](size_t i){
        DoWork(i, workLoad);
    });
}


// Parallel for each loop

void Example04(size_t size, int workLoad)
{


// Create input values

    vector<size_t> inputs(size);
    for (size_t i = 0; i < size; ++i)
        inputs[i] = i;

    parallel_for_each(inputs.cbegin(), inputs.cend(), [workLoad](size_t i){
        DoWork(i, workLoad);
    });
}



// Breaking out of loops early (with task group cancellation)
// Use default capture mode for nested lambdas.
// http://connect.microsoft.com/VisualStudio/feedback/details/560907/
// capturing-  variables-in-nested-lambdas


void Example05(vector<double>& results, int workLoad)
{
    task_group tg;
    size_t fillTo = results.size() - 5 ;
    fill(results.begin(), results.end(), -1.0);

    task_group_status status = tg.run_and_wait([&]{
        parallel_for(0u, results.size(), [&](size_t i){
            if (i > fillTo)
                tg.cancel();
            else
                results[i] = DoWork(i, workLoad);
        });
    });

    if (status != canceled)
        throw new InvalidValueFoundException();
```

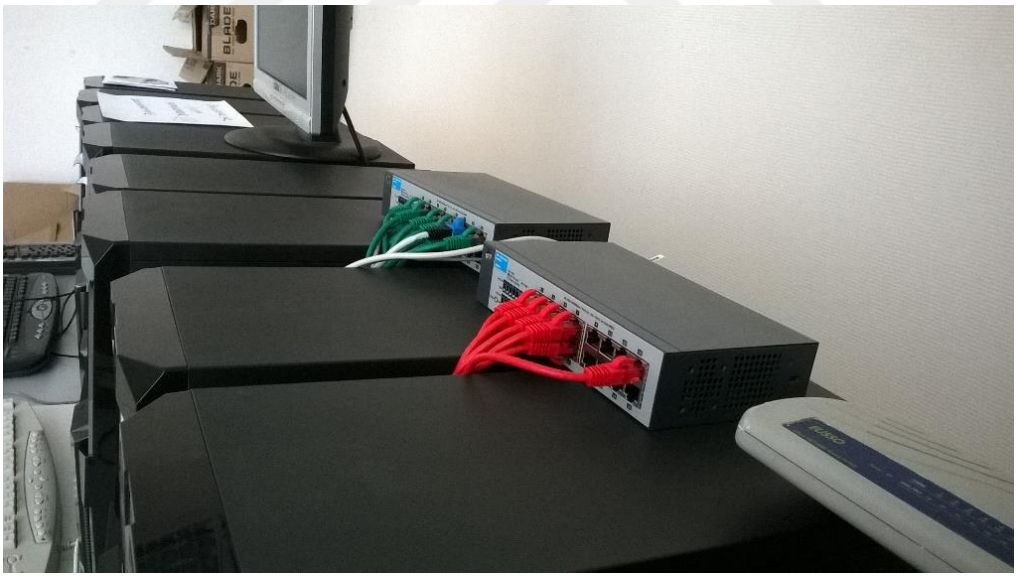
```
// No results in the last five elements of the array will be set.
// Some values  in the remaining array will be set but not all.

    for (size_t i = 0; i < results.size(); ++i)
    {
        if ((i > fillTo) && (results[i] != -1.0))
            throw new InvalidValueFoundException();
    }
}

// Handling exceptions

void Example06()
{
    bool foundException = false;
    g_SimulateInternalError = true;

    vector<double> results(100000);
    try
    {
        size_t n = results.size();
        parallel_for(0u, n, [&results](size_t i)
        {
            results[i] = DoWork(i, 10); // throws exception
        });
    }
    catch (ParallelForExampleException e)
    {
        printf( "Exception caught as expected.\n");
        foundException = true;
    }
    if (!foundException)
        throw InvalidValueFoundException();

    g_SimulateInternalError = false;
}


// Special Handling of Small Loop Bodies


// Note: The PPL supports specification of a range size but not custom range
partitioners

void Example07(vector<double>& results, int workLoad)
{
    size_t size = results.size();
    size_t rangeSize = size / (GetProcessorCount() * 10);
    rangeSize = max(1, rangeSize);

    parallel_for(0u,   size,   rangeSize,   [&results,   size,   rangeSize,
workLoad](size_t i)
    {
        for (size_t j = 0; (j < rangeSize) && (i + j < size); ++j)
            results[i + j] = DoWork(i + j, workLoad);
```

```
    });
}

void ParallelForExample(int workLoad, int numberOfSteps, bool verifyResult)
{
    vector<double> results(numberOfSteps);

    printf("Parallel For Examples (workLoad=%d, NumberOfSteps=%d)\n",
        workLoad, numberOfSteps);
    try

        {
        TimedRun([&results, workLoad](){ Example01(results, workLoad); },
            "Sequential for          ");
        VerifyResult(results, workLoad);

        TimedRun([&results, workLoad](){ Example02(results, workLoad); },
            "Simple parallel_for     ");
        VerifyResult(results, workLoad);

        TimedRun([numberOfSteps,   workLoad](){   Example03(numberOfSteps,
workLoad); },
            "Sequential for each     ");

        TimedRun([numberOfSteps,   workLoad](){   Example04(numberOfSteps,
workLoad); },
            "Simple parallel_for_each");

        TimedRun([&results, workLoad]() { Example05(results, workLoad); },
            "Canceling parallel_for  ");

        TimedRun([&results, workLoad]() { Example07(results, workLoad); },
            "Ranged parallel_for_each");
        VerifyResult(results, workLoad);
    }
    catch (InvalidValueFoundException e)
    {
        printf( "Error: Verification Failed\n");
    }
    printf("\n");
}

int main()
{
    printf("Basic Parallel Loops Samples\n\n");
#if _DEBUG
    printf("For most accurate timing results, use Release build.\n\n");
#endif
```

```
// Parameters: workLoad, NumberOfSteps, VerifyResult

    ParallelForExample( 10000000, 10, true );
    ParallelForExample( 1000000, 100, true );
    ParallelForExample( 10000, 10000, true );
    ParallelForExample( 100, 1000000, true );
    ParallelForExample( 10, 10000000, true );

    printf("parallel_for handling exceptions\n");
    Example06();

    printf("\nRun complete... press enter to finish.");
    getchar();
}
```

EK-2. Çalışılan Microsoft YBH kümesine ait resimler


NODE 7
Campus Net IP :
Cluster Net IP : 172.16.10.17
Subnet Mask : 255.255.255.0
Gateway : 172.16.10.1
DARK



EK-3. YBH Kümesini test için kullanılan **MonteCarlo Pi Calculator** programı c++ kodları

```
// PiCalculator.cpp : Defines the entry point for the console application.
//
/*
** An MPI program to estimate pi using the dartboard
** (monte-carlo) algorithm.
**
** Imagine a circle inscribed inside a square.
** The area of the circle is, of course: A-circ = pi * sqr(r).
** But we're trying to find pi, so we re-arrange to get:
** pi = A-circ / sqr(r)  -- (eqn 1).
** We know that sqr(r) is the area of one quarter of the square,
** so A-sq = 4 * sqr(r).
** Re-arranging again, we get:
** sqr(r) = A-sq / 4  -- (eqn 2).
** We can subsitute (eqn 2) into (eqn 1) to get:
** pi = 4 * A-circ / A-sq.
**
** Lastly, if we assume the darts land randomly somewhere inside
** the square, i.e. sometimes within the cicle, and sometimes
** outside the circle, then we can substitute the ratio of areas
** with a ratio of dart counts, i.e. a count of the darts which
** fell inside the cicrle over a count of those which fell inside
** the square (all of them).
*/

#include "stdafx.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "mpi.h"

#define NDARTS 1    /* number of throws at dartboard */
#define ROUNDS 10000000       /* number of times we throw NDARTS */
#define MASTER 0        /* task ID of master task */

double throw_darts (int nthrows);

#define sqr(x)((x)*(x))

int main(int argc, char **argv)
{

  double local_pi;        /* value of pi calculated by current task */
  double pi;              /* average of pi after "darts" are thrown */
  double avepi;           /* average pi value for all iterations */
  double pirecv;          /* pi received from worker */
  double pisum;           /* sum of workers pi values */
  double PI25DT = 3.14159265358979323846264643;
```

```
  int rank;               /* task ID - also used as seed number */
  int nproc;              /* number of tasks */
  int tag;                /* message tag */
  int i;
  int n;

  MPI_Status  status;



  /* Obtain number of tasks and task ID */
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &nproc);

  /* Set seed for random number generator equal to rank */
  srand (rank);

  avepi = 0;

  for (i = 0; i < ROUNDS; i++) {
    /* All tasks calculate pi using dartboard algorithm */
    local_pi = throw_darts(NDARTS);

    /* Workers send local_pi to master */
    /* - Message type will be set to the iteration count */
    if (rank != MASTER) {
      tag = i;
      MPI_Send(&local_pi, 1, MPI_DOUBLE, MASTER, tag, MPI_COMM_WORLD);
    }

    /* Master receives messages from all workers */
    /* - Message type will be set to the iteration count */
    /*   a message can be received from any task, as long as the */
    /*   message types match */
    /* - The return code will be checked, and a message displayed */
    /*   if a problem occurred */
    else {
      tag = i;
      pisum = 0;
      for (n = 1; n < nproc; n++) {
        MPI_Recv(&pirecv, 1, MPI_DOUBLE, MPI_ANY_SOURCE, tag,
                 MPI_COMM_WORLD, &status);
        /* keep running total of pi */
        pisum += pirecv;
      }
      /*
      ** Master calculates the average value of pi for this iteration
      ** not forgetting the Master's contribution
      */
      pi = (pisum + local_pi)/nproc;
      /* Master calculates the average value of pi over all iterations */
      avepi = ((avepi * i) + pi)/(i + 1);
      printf("   After %3d throws, average value of pi = %10.8f (error is
%.16f)\n",
             (NDARTS * (i + 1)),avepi, fabs(avepi - PI25DT));
    }
```

```
  }
  MPI_Finalize();

  return EXIT_SUCCESS;
}



double throw_darts(int nthrows)
{
  double x_coord;       /* x coordinate, between -1 and 1  */
  double y_coord;       /* y coordinate, between -1 and 1  */
  double pi;            /* pi  */
  double r;             /* random number between 0 and 1  */
  int score = 0;        /* number of darts that hit circle */
  int n;

  /* "throw darts at board" */
  for (n = 1; n <= nthrows; n++) {
    /* generate random numbers for x and y coordinates */
    r = (double)rand()/RAND_MAX;
    x_coord = (2.0 * r) - 1.0;
    r = (double)rand()/RAND_MAX;
    y_coord = (2.0 * r) - 1.0;

    /* if dart lands in circle, increment score */
    if ((sqr(x_coord) + sqr(y_coord)) <= 1.0)
      score++;
  }

  /* calculate pi */
  pi = 4.0 * (double)score/(double)nthrows;
  return(pi);
}
```