T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

CERN VERİ AĞINA (WLCG) YÖNELİK ANALİZ VE HESAPLAMA ALTYAPISI (TIER-3G) KURULUMU

YÜKSEK LİSANS TEZİ

AGÂH ALICI
Y1013.010013

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Hasan SAYGIN

Ocak, 2018

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **Y1013.010013** numaralı öğrencisi **Agah ALICI** 'nın **"CERN VERİ AĞINA (WLCG)YÖNELİK ANALİZ VE HESAPLAMA ALT YAPISI (TİER_3G)KURULUMU "** adlı tez çalışması Enstitümüz Yönetim Kurulunun 26.12.2017 tarih ve 2017/31 sayılı kararıyla oluşturulan jüri tarafından oybirliği ile Tezli Yüksek Lisans tezi olarak kabul edilmiştir.

Öğretim Üyesi Adı Soyadı	İmzası
Tez Savunma Tarihi : 17.01.2018	
1)Tez Danışmanı: PROF. DR. HASAN SAYGIN	.....
2) Jüri Üyesi : YRD. DOÇ. DR. SİNAN KUDAY	.....
3) Jüri Üyesi : YRD. DOÇ. DR. FERDİ SÖNMEZ	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır.** Aksi halde geçersizdir.

YEMİN METNİ

Yüksek lisans tezi olarak sunduğum “CERN VERİ AĞINA (WLCG) YÖNELİK ANALİZ VE HESAPLAMA ALTYAPISI (TIER-3G) KURULUMU” adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliografya’da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (…/…/2018)

AGÂH ALICI

ÖNSÖZ

Yüksek lisans tez çalışmalarına başladığım günden bu yana yardımlarını ve engin bilgilerini paylaşmayı benden esirgemeyen her zaman en doğru yolu gösteren çok değerli hocam Prof. Dr. Hasan SAYGIN'a en içten teşekkürlerimi sunarım. Tüm çalışmalarım boyunca bana bilgi ve sevgileriyle destek olan arkadaşlarıma ve sevgili aileme teşekkürü bir borç bilirim.

Ocak 2018 **AGÂH ALICI**

İÇİNDEKİLER

Sayfa

KISALTMALAR

CERN : Avrupa Nükleer Araştırma Konseyi
DRS : Dağıtık Kaynak Planlayıcısı
EGI : Avrupa Grid Altyapısı
LDAP : Hafifletilmiş Dizin Erişim Protokolü
LHC, BHÇ : Büyük Hadron Çarpıştırıcısı
LUN : Mantıksal Birim Numarası
SAN : Depolama Alanı Ağı
VM : Sanal Makine
TÜBİTAK : Türkiye Bilimsel ve Teknolojik Araştırma Kurumu ve
TAEK : Türkiye Atom Enerjisi Kurumu
WLCG : Küresel LHC Hesaplama Gridi

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

CERN VERİ AĞINA (WLCG) YÖNELİK ANALİZ VE HESAPLAMA ALTYAPISI (TIER-3G) KURULUMU

ÖZET

Hesaplamaya dayalı küme bilgisayarlar; günümüzde özellikle deneysel fizik, astronomi ve tıbbi bilimler gibi disiplinlerde geniş çaplı olarak kullanılmaya başlanmıştır. Gelişen teknolojiyle birlikte hesaplama yazılımlarını daha verimli çalıştırabilen çok çekirdekli (multicore) sunucu sistemleri küme bilgisayarların yerini almakta veya birlikte kullanılması konusunda çalışmalar yapılmaktadır. Bu tür sistemlere ihtiyaç, yapılan deney ve analizlerde ortaya çıkan yüksek verinin doğru ve zamanında işlenmesi, farklı analizlerin aynı anda farklı araştırmacılar tarafından yürütülmesi ve verimliliğin arttırılması gibi kritik nedenlerden ortaya çıkmaktadır. Dünya'nın başlıca deney merkezlerinden Avrupa Nükleer Araştırmalar Merkezi (CERN)'de, yıllık petabyte mertebelerinde deneysel yüksek veri üretildiği göz önüne alındığında, ihtiyaca cevap verebilecek bir veri ağının işlerliği ile analiz merkezlerinin üretkenliği, sürdürülebilirliği ve gelişimi önem kazanmaktadır. CERN'de yürütülmekte olan deneylerden ATLAS (A Toroidal LHC Apparatus) (Aad ve diğerleri, 2008), veri alımından birkaç yıl önce (2004) hesaplama modelini açıklamış (Adams ve diğerleri, 2004), buna göre Tier-0 merkezinde alınan ham verinin hiyerarşik yapıdaki Tier-1 ve Tier-2 merkezleri arasında paylaştırılarak dağıtık veriye dönüştürülmesi, CERN analiz tesisinde işlenmesi ve araştırmacıların veri yönetimi ile tekrar-işlemleme aşamalarında etkin rol alması öngörülmüştü. Bu amaçla geliştirilen Büyük Hadron Çarpıştırıcısı (LHC) veri ağı (WLCG), aynı dönemde kurularak deney-dışı (offline) yapılan ilk testleri ve başlangıç verisi işlemlemeyi başarı ile tamamlamıştır (Shiers, 2007). Diğer taraftan, belli başlı enstitüler ham verinin depolanması, çeşitli algoritmalarla ayrıştırılması ve dağıtılması gibi merkezi işlemleri gerçekleştiren bir ağın araştırmacıların son verinin (Ntuple) analizi için gereksinimleri karşılayamayacağını öngördüler. Bu dönemde, araştırmacılara yönelik kurulan, merkezi olmayan ve tamamen kurulduğu enstitünün kontrolünde yer alan Tier-3g merkezleri geliştirilmiştir (Gonzalez de la Hoz ve diğerleri 2008; Haupt 2010; Villipana 2012). Bu merkezler WLCG'nin bir parçası olup son deney verisini depolayabilmekte ve enstitülerin öncelikleri doğrultusunda analizleri gerçekleştirebilmektedirler. Enstitüler, açık kaynak kodlu geliştirilen ve çok çekirdekli sunucuların üzerinde efektif çalışabilen analiz yazılımlarını Tier-3g üzerinde geliştirmekte ve kullanmaktadır.

İstanbul Aydın Üniversitesi'nde geliştirilecek Tier-3g merkezi çalışmaları 2023 yılına kadar çalışmaya devam etmesi beklenen Büyük Hadron Çarpıştırıcısı verileri için yüksek luminosity fazında yapılacak analizlerde önem kazanacaktır. Bu çalışmalarda, son deney verisi formatına uygun (D3PD, xAOD) dengeli olarak çok çekirdekli işlemcilerde analiz edilebilecektir.

Anahtar Kelimeler: *Hesaplama, Veri, Analiz, CERN, BHÇ, Ağ.*

IMPLEMENTATION OF ANALYSIS AND COMPUTING INFRASTURUCTURE (TIER-3G) FOR CERN DATA GRID (WLCG)

ABSTRACT

Today, cluster computers based on computation have been widely used especially in disciplines like experimental physics, astronomy and medical sciences. With the evolving technology, multicore server systems, which can run the calculation software more efficiently, replacing cluster computers or there are studies to make them work together. The main need for such systems arises from the critical reason such as to handle big data of the experiments and analyses accurately and in time, to ensure that different analyses are carried out by different researchers at the same time and to increase efficiency. When one consider that big data is produced in petabyte ranks annually at the Center for Nuclear Research (CERN), one of the world's major experiments, the operation of a data network responding to need and the productivity, sustainability and development of analysis centers becomes important. ATLAS (A Toroidal LHC Apparatus) (Aad et all, 2008) one of the experiments running at CERN, explained the calculation model a few years before data acquisition (2004). According to that explanation, transformation of the raw data received at the Tier-0 center to the distributed data by sharing among the Tier-1 and Tier-2 centers in the hierarchical structure, processing in CERN analysis facility and researchers taking an active role in data management and reprocessing stages were predicted. The Large Hadron Collider (LHC) data network (WLCG), which was developed for this purpose and established during the same period, has successfully completed initial testing and initial data processing (Shiers, 2007). On the other hand, Major institutes predicted that a network performing central processing such as storage of raw data, decomposition and distribution by various algorithms would not meet the requirements of the researchers to analysed the n-tuples. In this period, Tier-3g centers have been developed which are set up for researchers, decentralized and fully under the control of the institute that they are established (Gonzalez de la Hoz et all 2008; Haupt 2010; Villipana 2012). These centers are part of the WLCG and can store the latest experimental data and carry out analyses in line with the priorities of the institutes. Institutes develop and use the analysis software programs, which are open sourced developed and can work effectively on multicore servers, on Tier-3g centers. Tier-3g center studies, will be developed in Istanbul Aydin University, will gain importance for the high luminosity data of Large Hadron Collider which is expected to continue to run until 2023. In these studies, the latest experimental data, can be balanced and analysed in multi-core processors in the appropriate format (D3PD, xAOD).

Key words: *Computing, Data, Analysis, CERN, LHC, Network.*

1. GİRİŞ

Avrupa Nükleer Araştırma Konseyi (CERN: Conseil Européen pour la Recherche Nucléaire) hesaplama modeli; olay filtrasyonu, Tier-0, Tier-1, Tier-2 hiyerarşik modelini spesifik roller ve sorumluluklar yoluyla takip eder. Olay filtrasyonu ham verinin birleştirilmesini sağlarken, CERN sınırlarındaki bilgi işlem merkezi olarak görülebilecek olan Tier-0 merkezi; kalibrasyon, validasyon, depolama ve formatlama (AOD, ESD gibi) görevlerini gerçekleştirir. Tier-1 merkezleri dünyanın farklı enstitülerinde yer alan şebeke noktalarıdır ve veri replikasyonu, rekonstrüksiyonu görevlerini gerçekleştirirken yüksek miktarlardaki verinin ayıklanması ve depolanmasına da yardımcı olur. Tier-2 merkezleri, Monte-Carlo simulasyon üretimleri ve veri analizlerinin karşılaştırmalı gerçekleştirilebileceği kompakt veri formatına (DPD, AOD gibi) uyumlu yüksek performanslı ağ merkezleridir.

CERN kaynaklarından bağımsız olarak enstitülerin küme bilgisayarlar yoluyla daha hızlı işlemleme ve kendi kontrollerinde veri alma, simülasyon üretimi görevlerini yapabildiği ağ noktalarına Tier-3g ismi verilmiştir. Enstitüler için, CERN/LHC veri ağına dâhil olabilme kriterleri deney koordinatörlükleri tarafından belirlenmiştir: temelde en az 100 Mbps band genişliği, 25 çoklu çekirdekli işlemleme gücü ve 10 TB depolama alanı gözetilmektedir.

Bu çalışma, CERN/ATLAS test deney verisini daha verimli şekilde Tier-3g üzerinde çalıştırabilecek yeni yazılımların iyileştirilmesini ve kurulumların tamamlanarak sonuçların yayınlanmasını kapsamaktadır.

Yüksek verinin (big data) işlemlenmesi konusunda Dünya üzerinde büyük çaplı projeler geliştirilmektedir. Buna örnek olarak, internetin ilk protokollerinin keşfedildiği İsviçre'nin Cenevre kentindeki CERN merkezinin WLCG projesi verilebilir. Gelişen teknolojilerle birlikte, uluslararası işbirliği ile kurulmuş olan veri

ağında karmaşık LHC verisinin analizi petabyte derecesindeki analizlerin ilki olma niteliğindedir.

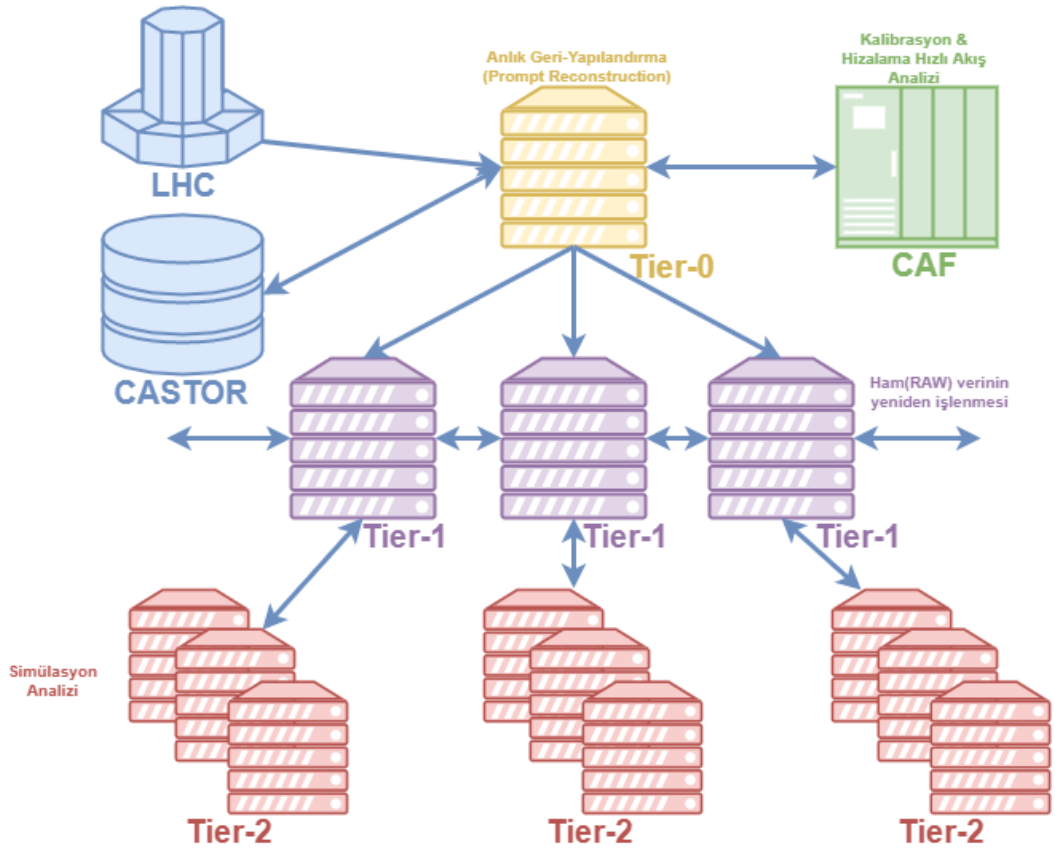
1.1 ATLAS Hesaplama Modeli

ATLAS hesaplama modeli, büyük hadron çarpıştırıcısı (Large Hadron Collider: LHC) tarafından üretilen verilerin işlenebilmesi için 2002 yılından itibaren grid yapıda oluşturulmaya başlamıştır. ATLAS hesaplama modeli ile (Jones & Barberis, 2010:239);

- Yıllık üretilen ham ve işlenmiş veri boyutunun petabyte'lar seviyesinde olması
- Veri formatlarının çeşitliliği
- Analizlerin tüm dünyadaki araştırmacılar tarafından uygulanabilir olması

sorunlarının aşılması hedeflenmiştir. Bu model 2005 yılında LHC komitesi tarafından değerlendirilmiş ve Computing Technical Design Report (C-TDR) oluşturulmuştur (ATLAS, 2005).

ATLAS, merkezi kontrol altında 3 katmanlı (Tier) yapısını korumaktadır. Bu 3 katmanın dışında üniversite ve araştırma grupları tarafından oluşturulan ve bu tezde kurulumu açıklanan "Tier-3" katmanı hesaplama merkezleri bulunmaktadır. ATLAS hesaplama modelinde yer alan katmanlara ait akış, Şekil 1.1'de sunulmaktadır.



Şekil 1.1: ATLAS İş Akışı

Tier-0: ATLAS hesaplama modelinin merkezinde bulunan ve CERN araştırma merkezinde bulunan Tier-0 katmanı, büyük hadron çarpıştırıcısı tarafından oluşturulan devasa hızlı akış analizi (express stream analysis), kalibrasyon, hizalama ham verilerini depolamaktadır.

Tier-1: Büyük hesaplama merkezlerinde konumlandırılan Tier-1 merkezler, Tier-0 katmanından aldıkları ham verileri farklı veri formatlarına dönüştürerek uzun süre depolamak ile yükümlüdür. Bunun yanı sıra, çeşitli hizmetler aracılığı ile Tier-2'lerin ihtiyaç duyduğu verilerin sağlanması görevini üstlenirler. Tier-1 merkezleri, sistem yöneticileri haricinde araştırmacılar ve diğer kullanıcılara hizmet vermemektedir.

Tier-2: Kullanıcıların veriler ile etkileşime geçebildiği katman olarak tanımlanır. Tier-2'deki veriler, ihtiyaca göre kısa süreli depolanmaktadır.

Tier-3: Enstitüler ve araştırma grupları tarafından kullanılan ve verilere Tier-2 tarafından çeşitli araçlar (rucio, gridFTP vb.) kullanarak erişip, hesaplamaları kendi bünyesinde yapabilen katmandır.

1.2 EGI ve CERN Gridi (WLCG)

EGI (European Grid Infrastructure, Avrupa Grid Altyapısı), araştırma ve yenilik için gelişmiş bilgi işlem hizmetleri sunmak için kurulmuş federe bir e-altyapıdır. Kamu tarafından finanse edilmekte olan EGI e-altyapısı, Avrupa'da ve dünyada yüzlerce veri merkezi ve bulut sağlayıcıdan oluşmaktadır.

EGI, hesaplama, depolama, veri ve destek için geniş bir hizmet yelpazesi sunmaktadır. EGI 730.000'den fazla mantıksal işlemci ve 650 PB'lik disk ve kaset erişimi sağlar.

LHC Küresel Hesaplama Grid (WLCG) projesi, 42 ülkede 170'in üzerinde bilgi merkezinin küresel bir işbirliği olup, ulusal ve uluslararası grid altyapılarını birbirine bağlar.

WLCG projesinin misyonu, büyük hadron çarpıştırıcısı tarafından üretilen (2017'de beklenen ~ 50 petabayt) veriyi depolamak, dağıtmak ve analiz etmek için küresel bilgi işlem kaynakları sağlamaktır.

1.3 Türkiye'deki Durum ve Tarihçesi

Türkiye'nin CERN ile çalışmaları ilk defa 1961 yılında gözlemci statüsü ile başlamıştır. Bu bilimsel çalışmalar, başlangıçta bireysel olarak yürütülmekte olsa da, daha sonra Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) ve Türkiye Atom Enerjisi Kurumu (TAEK) tarafından devam ettirilmiştir. Türkiye ile CERN arasındaki çalışmaların çerçevesini belirleyen TAEK-CERN işbirliği anlaşması 2008 yılında Cenevre'de imzalanmış, 2010 yılında ise Türkiye'nin tam üyelik başvurusu ile süreç farklı bir boyuta taşınmıştır. Ancak bu süreçte tam üyelik statü başvuruları, "ortak üyelik" ya da "tam üyelik yolunda ortak üyelik" statü başvuruları olarak güncellenmiştir. Aynı dönemde başvuru yapan Slovenya, Sırbistan, İsrail, Güney Kıbrıs Rum Yönetimi ülkelerinin başvuruları "tam üyelik yolunda ortak üyelik" statüsünde değerlendirilmiştir. 2014 yılında imzalanan ilgili kanunun, 2015 yılında yayınlanması ile Türkiye'nin CERN'e ortak üyelik süreci tamamlanmıştır.

1.4 Kullanıcı Veri Analizi

Bu bölümde, CERN tarafından üretilen verilere ait veri yapıları ve Tier-3g sisteminde analiz hesaplamalarında kullanılmakta olan katmanlı kod yapıları anlatılmaktadır.

1.4.1 Veri yapısı

Çizelge 1.1'de CERN tarafından üretilen çarpışma testlerine ait verilerin, ham ve dönüştürülen veri yapıları ile ilgili veri yapılarının açıklamaları ve ortalama veri boyutları listelenmiştir

Çizelge 1.1: Veri Yapıları

Veri Adı	Açıklama	Boyut
Monte Carlo	Benzetim ile elde edilmiş veri	-
RAW (Ham Veri)	Dedektör çıktısı analog veri	1.6 MB/olay
ESD (Event Summary Data)	Obje formatında olay dosyası	1.0 birim
AOD (Analysis Object Data)	Analiz objeleri veri dosyası	0.2 birim
TAG	Örneklenmiş olay ön izleme	0.01 birim
DPD (Derived Physics Data)	İnceltilmiş, fizik veri dosyası	0.01 birim
Flat n-tuple	Sonuç Histogram verisi	~KB
HITS	RAW veriden üretilen dijital veri	-

1.4.2 Kod yapısı

Kod yapısı içinde yer alan 4 önemli katman aşağıdaki gibidir:

- **Temel Kütüphaneler (non-HEP):** git, make, cmake, gcc-c++, gcc, gfortran, libX11-devel, libXpm-devel, libXft-devel, python gibi çalışma çerçevesini (framework) oluşturacak derleyici ve temel araçları içerir. Bu kütüphaneler genel amaçlı olup sadece deneysel analizler için hazırlanmamıştır.
- **Temel Kütüphaneler (HEP):** Cernlib, Geant4, Root gibi deneysel amaçlar için hazırlanmış kod kütüphaneleridir. Analizlerde yapılacak hesaplamaların ve analizlerin temel fonksiyonlarını (class yapısını) sağlarlar.
- **Deneysel Çalışma Çerçevesi (Framework):** Simülasyon veya gerçek deney verisi ile birlikte hazırlanmış deneye özgü temel özelliklerin sunulduğu katmandır. Örneğin deneydeki olay modellemesi, detektör özellikleri, kalibrasyon, veri alımı gibi veriyi etkileyen süreçlerin modellemesini kullanıcılara sunar. ATLAS deneyinde "Rootcore" ismiyle anılan çalışma çerçevesinin temel hali (base) buna örnek verilebilir.

- **Uygulamalar:** Kullanıcıların (deney üyelerinin) kendi analiz algoritmalarını yazarak uyguladığı kodları ifade eder. Örneğin ATLAS deneyinde "Top Quark Physics" çalışma grubunun oluşturduğu TTH analiz kodu, bir RootCore base kullanılarak oluşturulmuştur. Kullanıcılara yazacakları analiz kodlarına yardımcı olması amacıyla periyodik olarak güncellenen RootCore-Base sürümleri hakkında bilgilendirme ve eğitimler verilmektedir.

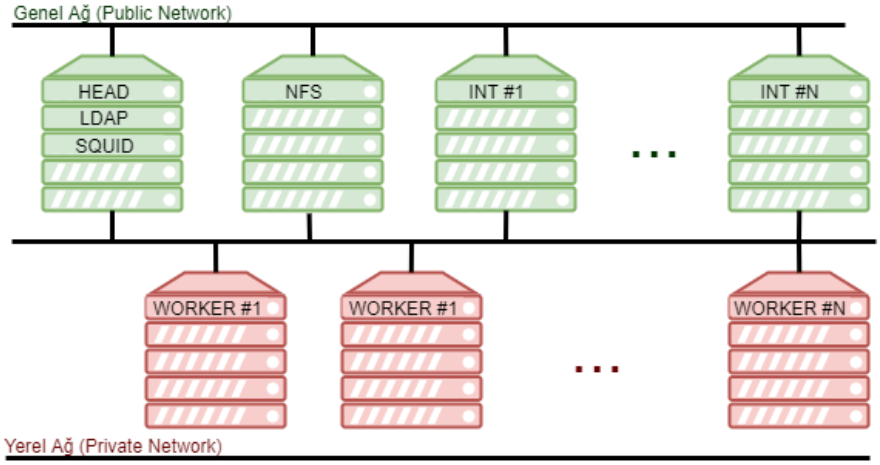
1.5 Kullanıcı Girişi ve Simülasyon İşlemleri

İstanbul Aydın Üniversitesi Tier-3g sistemi, (interactive sunucu için) linux komut satırından (console) çalıştırılmak üzere tasarlanmıştır. Sistem kullanımı için yetkilendirilen her kullanıcıya secure shell (ssh) erişimi için kullanıcı hesabı, /local/xrootd/ dizini içerisinde kullanıcı adı ile aynı isimde bir klasör otomatik olarak oluşturulmakta ve kullanıcının kendi analiz dosyalarını bu alanda barındırması sağlanmaktadır. İlgili klasör NFS sunucu ile Tier-3g sisteminde yer alan diğer sunucular ile paylaşılmakta olup, Condor batch sistemi ile worker sunucusuna gönderilen işler için veriler aynı kaynaktan okunabilmektedir. Üniversite içi bağlantılar için; güvenlik duvarı yetki tanımı yapılmakta, üniversite dışı bağlantılar için; aynı kullanıcı hesap bilgileri VPN hesabı oluşturularak ilgili sunuculara erişim yapılması sağlanmaktadır.

Simülasyon işlemleri komut satırı üzerinden interactive sunucudan ya da tercihen Condor batch sistemi ile worker sunucular üzerinde yapılabilmektedir. Rucio ve GridFTP uygulamaları ile analiz dosyalarının indirilip kullanılabilmesi için /local/xrootd/ dizini altında her bir kullanıcı için alan tahsis edilmiştir. Simülasyon işlemlerinin worker sunucular üzerinde çalıştırılmasının tercih edilmesi durumunda aynı fiziksel yol ile mount edilmiş tahsisli disk alanları kullanılarak gerçekleştirilebilmektedir.

2. MATERYAL VE YÖNTEM

Tier-3g sistemi temel olarak 5 farklı sunucu türü barındıran bir sunucu kümesidir. Kümeyi oluşturan sunucular işlevlerine göre yerel ağ (private network) ya da genel ağ (wan) içerisinde yer alır. Sistem içerisinde yer alan INTERACTIVE ve WORKER sunucu sayısı, beklenen işlem gücü kapasitesi ile orantılı olmakla birlikte N tane sunucunun birlikte çalışabilmesine olanak tanıyacak şekilde tasarlanmıştır.



Şekil 2.1: Tier-3g Sunucu Ağ Yapısı

2.1 Donanım Bileşenlerinin Kurulumu

Tier-3g kurulumu için Vmware Hypervisor 6.5 sunucu sanallaştırma sistemi ile yönetilen, Dell 820 (25 Core (4 Sockets) Xeon E5 CPU, 2133 Mhz 144GB RAM) ve EMC SAN Storage (12 TB LUN) kullanılmıştır. Vmware Hypervisor ile aşağıdaki sanal sunucu birimleri oluşturulmuş ve ilgili donanım yapılandırması yapılmıştır;

- **HEAD Sunucu**: 3 Core CPU, 24GB RAM, 1 TB Disk Alanı
- **INTERACTIVE Sunucu**: 4 Core CPU, 36 GB RAM, 4 TB Disk Alanı
- **NFS Sunucu**: 4 Core CPU, 16 GB RAM, 1.5 TB Disk Alanı
- **WORKER Sunucu**: 12 Core CPU, 48 GB RAM, 4 TB Disk Alanı
- **LDAP Sunucu**: 1 Core CPU, 8 GB RAM, 0.5 TB Disk Alanı

- **SQUID Sunucu**: 1 Core CPU, 12 GB RAM, 1 TB Disk Alanı.

Yukarıda belirtilen sunuculara ek olarak laboratuvar kullanım saatleri dışında kullanılmak üzere, laboratuvar bilgisayarlarını çalıştıran sanal sunucu üzerinde 2 adet WORKER sunucu kullanılmak üzere planlanmıştır.

2.2 Ağ (Network) ve Cluster Planlama

Sistem için network altyapısı ATLAS Computing dokümantasyonunda (https://twiki.cern.ch/twiki/bin/view/AtlasComputing/Tier3gNetworkConfig) belirtildiği üzere internet ağına açık (public) ve internet ağına kapalı (private) iki ağ grubu ile yapılandırılmıştır. Üniversitenin sahip olduğu IP havuzundan 91.239.204.41-48 aralığındaki IP adresleri rezerve edilmiş, 91.239.204.43, 91.239.204.44, 91.239.204.45 IP adresleri internet ağına açık (public), diğer IP adresleri ise internet ağına kapalı (private) olarak yapılandırılmıştır. Planlanan sunucu IP adresi grupları;

- **91.239.204.41** SQUID Proxy Sunucu (private network)
- **91.239.204.42** LDAP Sunucu (private network)
- **91.239.204.43** HEAD Sunucu (public network)
- **91.239.204.44** NFS Sunucu (public network)
- **91.239.204.45** INTERACTIVE Sunucu (public network)
- **91.239.204.46** WORKER Sunucu (private network)
- **91.239.204.47** WORKER Sunucu #2 (private network)
- **91.239.204.48** WORKER Sunucu #3 (private network)

2.3 Jenerik Servislerin Ayarlanması

Tier-3g sistemi için, işletim sistemi ve jenerik sistem yapılandırma dokümantasyonlarının 2014 yılında oluşturulması ve başta işletim sistemi (Scientific Linux 5 serisi) olmak üzere tanımlı servislerinin hizmet süresinin sona ermiş olması nedeniyle, sistem kurulumlarının temel yapılandırmaya sadık kalınarak güncel işletim sistemleri (Scientific Linux 6.9) ve bağımlı jenerik sistemler ile değiştirilmesine karar verilmiştir.

Oluşturulan sunucuların yapılandırmaları ve görev paylaşımları nedeniyle, işletim sistemi ve gömülü genel servisler kurulduktan sonra ilgili sunuculara özel hizmetler

kurulmuştur. Her bir sunucuya ait servis için CERN kurulum dokümantasyonunda bulunan betik scriptleri güncellenerek kullanılmıştır.

Kurulum ve bakım avantajları nedeniyle tüm sunuculara aynı işletim sistemi kurulmuştur. Aşağıda, sunucu işletim sistemi kurulum adımları ve her bir sunucuya ait jenerik yapılandırma bilgileri bulunmaktadır.

2.3.1 İşletim sistemi kurulumu

Sistem kurulumunda referans alınan ATLAS Computing dokümantasyonunda bulunan bazı uygulamaların Scientific Linux işletim sisteminin 7.0 ve üzeri sürümlerde kararlı çalışmaması nedeniyle, ortak işletim sistemi olarak yayındaki en üst sürüm olan Scientific Linux 6.9'un kullanılmasına karar verilmiştir. İşletim sistemine ait kurulum DVD'si https://www.scientificlinux.org/downloads/sl-mirrors/ adresinden indirilip, ilgili sanal sunucuya bağlanarak aşağıdaki anlatımı yapılan kurulumlar tamamlanmıştır.

DVD, ilgili sunucuya takılıp sunucu başlatıldığında Scientific Linux kurulum başlangıç ekranı görüntülenir. İlk kurulum için "Install or upgrade an existing system" seçeneği seçilerek devam edilir. İlgili seçenek kullanıldığında Scientific Linux kurulumu için gerekli kurulum dosyaları açılmaya başlar. Bu işlem sunucu donanım yapılandırması ve kullanılan donanım özelliklerine göre birkaç dakika sürebilir.

Kurulum dosyalarının yüklemesi tamamlandığında atanmış disk donanımı test onayı ekranı görüntülenir. Bu ekran yeni takılmış diskler için kurulum öncesi gerekli kararlılık testlerini gerçekleştirir. İşletim sisteminde daha sonra çıkabilecek sorunlara karşı disklerin taranması tavsiye edilmektedir. Disk taraması işlemini yapmak için ekranda görüntülenen "OK" düğmesine, disk taraması yapmadan kurulum işlemine devam etmek için ise "Skip" düğmesine basılır.

Bu ekrandan sonra cihaz, kurulum için hazırdır. Toplam 12 adımda tamamlanacak olan süreçte sunucular arası farklılık gösteren ayarlar ayrıca belirtilecektir. Bunlar dışında kalan ayarlar tüm sunucular için aynı yapılandırılabilir.

Disk taraması sonunda kurulum sihirbazı açılır. "Next" tuşuna basılarak kurulum işlemine devam edilir. Kurulum, dil seçeneğinin ayarlanması ile başlar. Karşılaşılan problemlerde çeviri hatalarını minimize etmek ve anahtar kelime kullanımı kolaylığı açısından kurulum İngilizce dilinde yapılmıştır.

Bir sonraki adımda işletim sistemi depolama türü seçimi yapılır. Kullanılan donanım özelliklerine göre ilgili seçenek işaretlenerek gerekli ayarlamalar yapılır. Bu projede sanal sunucu ve SAN yapılandırması kullanıldığı için "Basic Storage Devices" seçeneği ile kuruluma devam edilmiştir.

Bir sonraki adımda cihaza ait sunucu (host) ve ağ (network) bağlantı ayarları yapılmaktadır. Sunucunun görevine göre isimlendirilmesi, daha sonraki yapılandırmalarda kolaylık sağlayacaktır. Sunucunun hem yerel ağ (local network) hem de genel ağ (wan) için gerekli ayarlar ilgili ekrandaki "Configure Network" düğmesi yardımı ile tamamlanabilir.

Bir sonraki adım sunucuya bağlanmış olan fiziksel disk (ya da VM ile sağlanan LUN) alanı, uygulamaların kullanabileceği şekilde ayrılmasını sağlayacak mantıksal sürücü alanlarının oluşturulmasını sağlamaktadır. Bu projede her bir sunucunun mantıksal sürücü birimlerinin birbirinden farklılık göstermesi nedeniyle mantıksal sürücü ayarları her bir sunucu başlığının altında ayrıca belirtilecektir.

Mantıksal sürücülerin oluşturulmasında fiziksel disklerin işletim sistemi tarafından farklı olarak boyutlandırılmış alt sanal depolama birim gruplarına bölünmesi hedeflenir. Böylece işletim sistemi tek bir fiziksel diski birden fazla mantıksal sürücüye bölebilir ya da birden fazla fiziksel diski aynı mantıksal sürücü içerisinde birleştirerek işletim sistemi içerisindeki gereksinimleri karşılayabilir.

Bir sonraki adımda, işletim sisteminin sunucu açıldığında başlamasını (boot) sağlayan "boot loader" kurulumu ile ilgili ayarlar mevcuttur. İşletim sistemi kurulum sihirbazı "boot loader"ı fiziksel disk üzerindeki gerekli mantıksal sürücüye yerleştirmek için varsayılan ayarları otomatik olarak oluşturmaktadır. Bu kurulumda bakım kolaylığı açısından varsayılan "boot loader" yolunun kullanımına karar verilmiştir.

İsteğe bağlı olarak sunucu başlatma şifresi (burada bahsedilen işletim sistemi açılış şifresi değildir) bu ekrandan oluşturulabilir.

Bir sonraki adımda, işletim sistemi tipi ve yüklenecek ek özellikler belirlenmektedir. Proje kapsamında her bir sunucu farklı bir amaçla kullanılacağı için, "Minimal" seçeneği kullanılmıştır. Bu seçenek ile işletim sistemi için gerekli çekirdek yapılar kurularak, sunucu temelli özel yapılandırma sonraki ekranlarda yapılacaktır.

Bir sonraki adımda, ilgili işletim sistemi için gerekli paketler (her bir sunucu için ayrıca belirtilmiştir) kurulmak üzere işaretlenir.

Önceki adımlar tamamlandığında Scientific Linux kurulumu başlar. Sunucu donanım özellikleri ve önceki ekranlardaki yapılandırmalara göre (seçilen paketler, mantıksal sürücü birimleri gibi) kurulum birkaç dakika alabilir. Gerekli sistem ve paketlerin kurulumu ardından "boot loader" kurulumu yapılarak işlem tamamlanacaktır.

Tüm mantıksal sürücüler, kurulum için gerekli ve/veya kullanıcı tarafından kurulmak üzere ayarlanmış dosyaların kurulmasının ardından kurulumun sorunsuz tamamlandığına ait ekran görüntülenir. Bu ekrandan sonra işletim sistemi ilk kez açılmaya hazırdır.

Tüm kurulum adımları tamamlanıp sunucu yeniden başlatıldığında teorik olarak kullanıma hazırdır. Sunucu kişiselleştirme işlemleri yapılmaya başlamadan önce var olan kurulumun bir kopyasının oluşturulması daha sonra karşılaşılacak problemlerde işletim sisteminin tekrar kurulması yerine, kurulumu tamamlanmış halinin daha hızlı devreye alınmasını sağlayacaktır.

İşletim sistemi kurulumu sonrası yüklü uygulamaların en güncel ve kararlı sürümleri ile güncellenebilmesi için tüm sistemin güncellemelerinin yapılması önem arz etmektedir. Bu güncellemeler sistemin kararlı şekilde çalışmasına katkıda bulunduğu gibi olası güvenlik risklerini de ortadan kaldırmaktadır. "yum update all" komutu ile gerekli tüm sistem güncellemeleri yapılarak sistem yeniden başlatılabilir. Bu işlemden sonra sunucuya özel uygulamaların kurulumuna başlanabilir. Zamanlanmış görev yöneticisi (crontab) üzerinden sistem güncelleme işleminin tanımlanması, elle müdahale etmeden sistemin belirli periyotlar ile kendini güncellemesini sağlayacaktır.

2.3.2 SQUID Proxy sunucusu

SQUID Proxy sunucusu Tier-3g sistemi içerisindeki sunucuların internet üzerinden indireceği veriler için arabirim oluşturmakta, gerekli dosyaları önbelleğinde tutarak aynı verinin yeniden indirilmesini engellemektedir. Proxy sunucusuna ait mantıksal sürücü birimi bilgileri aşağıda belirtilmektedir. Kurulum kapsamında CERN için özelleştirilmiş Frontier-Squid 1.1.1 (http://frontier.cern.ch/) sürümü Proxy hizmeti kullanılmıştır.

```
part /boot --asprimary --fstype="ext3" --size=300 --ondisk=vda

part pv.01 --size=1 --grow --ondisk=vda
# LVM configuration so that we can resize /var, /opt , /tmp
volgroup sysvg pv.01
logvol swap --fstype="swap" --vgname=sysvg --size=4096   --name=lvswap
logvol /    --fstype="ext3" --vgname=sysvg --size=10000  --name=lvroot         #  10 GB for root partition
logvol /tmp --fstype="ext3" --vgname=sysvg  --size=2000      --name=tmp            #   2 GB for tmp
logvol /var --fstype="ext3" --vgname=sysvg  --size=2000      --name=var            #   2 GB for var
logvol /opt --fstype="ext3" --vgname=sysvg  --size=2000      --name=opt            #   2 GB for opt

logvol /local/scratch --fstype="ext3"  --vgname=sysvg  --size=1 --grow  --name=scratch        #   rest of space for scratch
```

```
part pv.02 --size=1 --grow --ondisk=vdb
volgroup squidvg pv.02
#  LVM for /var/log/squid
logvol /var/log/squid --fstype="ext3"   --vgname=squidvg   --size=50000  --name=squidlog   # squid log space (Org. 50 GB)
#  LVM for /var/cache/squid
logvol /var/cache/squid --fstype="ext3"  --vgname=squidvg  --size=225000 --grow  --name=squid    # squid cache space (Org. 225 GB)
```

Kurulan Paketler: base, Emacs

Kurulan Temel Uygulamalar: subversion, sysstat, xinetd, ntp, ruby-rdoc, puppet, frontier-squid.x86_64 (http://frontier.cern.ch/dist/rpms/RPMS/noarch/frontier-release-1.1-1.noarch.rpm)

2.3.3 LDAP sunucusu

Lightweight Directory Access Protocol (LDAP) sunucusu, Tier-3g sistemi içerisindeki yetkilendirme mekanizmasını oluşturmaktadır. Komut satırı üzerinden çalıştırılan Tier-3g sistemi için gerekli konsol yetkilendirmeleri LDAP sunucusu sayesinde tek bir noktadan yapılmaktadır. Oluşturulan sistem içerisindeki sunucu sayısı artsa bile, yetkilendirme tek noktadan yapıldığı için sunucu yönetimi için harcanan süreden tasarruf edilmektedir. LDAP sunucusu için oluşturulan mantıksal sürücü birimi yapılandırması aşağıda belirtilmiştir.

```
part /boot --asprimary --fstype="ext3" --size=300 --ondisk=vda
part pv.01 --size=1 --grow --ondisk=vda
# LVM configuration so that we can resize /var, /opt , /tmp
volgroup sysvg pv.01
logvol swap --fstype="swap" --vgname=sysvg --size=4096   --name=lvswap
logvol /    --fstype="ext3" --vgname=sysvg --size=10000  --name=lvroot         #  10 GB for root partition
logvol /tmp --fstype="ext3" --vgname=sysvg  --size=2000      --name=tmp            #   2 GB for tmp
logvol /var --fstype="ext3" --vgname=sysvg  --size=2000      --name=var            #   2 GB for var
logvol /opt --fstype="ext3" --vgname=sysvg  --size=2000      --name=opt            #   2 GB for opt
```

```
part pv.02 --size=1 --grow --ondisk=vdb
volgroup squidvg pv.02
#  LVM for /var/log/squid
logvol /var/log/squid --fstype="ext3"   --vgname=squidvg   --size=50000  --name=squidlog   # squid log space (O-50 GB)
#  LVM for /var/cache/squid
logvol /var/cache/squid --fstype="ext3"  --vgname=squidvg  --size=225000 --grow  --name=squid    # squid cache space (O-225 GB)
```

Kurulan Paketler: base, Emacs

Kurulan Temel Uygulamalar: openssl-perl, openldap*, openldap-clients, openldap-servers, subversion, sysstat, xinetd, ntp, ruby-rdoc, puppet

CERN Tier-3g sistemi için OpenLDAP paketi kullanılmaktadır.

2.3.3.1 OpenLDAP kurulumu

Komut satırından “slappasswd” komutu çalıştırılarak ayar dosyalarında kullanılabilecek kriptolanmış şifre anahtarı oluşturulur.

```
[root@ldap ~]# slappasswd
New password:
Re-enter new password:
{SSHA}bHSiwuPJEypHS6zHSE2Uy7M69sQjmkPL
```

Bu anahtar OpenLDAP yapılandırmalarında gerektiğinde kullanılacağı için saklanır.

“/etc/openldap/slapd.d/cn=config/olcDatabase={2}bdb.ldif” dosyasında bulunan “olcSuffix” değeri, oluşturulacak domain adı ile (ör: “olcSuffix: dc=ct3,dc=aydin,dc=edu,dc=tr”), “olcRootPW” değeri slappasswd komutu sonucunda oluşturulan kripto anahtarı ile (ör: “olcRootPW: {SSHA}bHSiwuPJEypH56…QjmkPL”), “olcRootDN” alanı seçilmiş olan domain adına ait “Manager” CN alanı ile değiştirilir (ör: olcRootDN: cn=Manager,dc=ct3,dc=aydin,dc=edu,dc=tr).

Aynı klasörde bulunan “olcDatabase={1}monitor.ldif” yapılandırma dosyasında yer alan “olcAccess” alanındaki değer, daha önce belirlediğimiz “olcRootDN” ile değiştirilir. (ör: olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" read by dn.base="cn=manager,dc=ct3,dc=aydin,dc=edu,dc=tr" read by * none”).

Yapılan işlemlerin doğruluğu “slaptest –u” ile kontrol edilir.

```
[root@ldap ~]# slaptest -u
config file testing succeeded
```

Bu yapılandırma dosyaları OpenLDAP sunucusunun çalışabilmesi için gerekli olduğundan ayarların test edilmesi önem taşımaktadır.

OpenLDAP sunucusuna ait yapılandırma dosyaları tamamlandıktan sonra temel kullanıcı kataloğunun oluşturulması için EK A’da yer alan Şekil A.1’deki betik dosyası çalıştırılır.

Betik çalıştırıldığında, /tmp/ klasöründe database0.XXXXXXXXXX adı ile LDAP temel kataloğunu oluşturacaktır. Bu katalog “ldapadd” komutu ile LDAP sunucusuna tanıtılır.

```
ldapadd -x -v -D 'cn=Manager,dc=ct3,dc=aydin,dc=edu,dc=tr' -w 'xxxyyyzzztt' -H ldap://ldap.ct3.aydin.edu.tr -f /tmp/database0.XXXXXXXX
```

2.3.4 HEAD sunucu

Head sunucusu başta CVMFS (CernVM File System - https://cernvm.cern.ch/portal/filesystem) olmak üzere birçok sistemin çalıştığı ve verilere ortak ulaşım ve denetim yapılan sunucudur. Head sunucusu aynı zamanda XRootD gibi dağıtık depolama sistemlerinin yönetim merkezi olarak konumlandırılmıştır.

Sunucu mantıksal depolama yapılandırması aşağıda verilmiştir.

```
part /boot --asprimary --bytes-per-inode=4096 --fstype="ext3" --ondisk=sda --size=300
part pv.01 --size=1 --grow  --ondisk=sda

# LVM configuration
volgroup sysvg pv.01
logvol /                --vgname=sysvg  --size=20000     --name=sys
logvol /var             --vgname=sysvg  --size=10000     --name=var
logvol /tmp             --vgname=sysvg  --size=10000     --name=tmp
logvol /opt             --vgname=sysvg  --size=10000     --name=opt
logvol /var/log/condor  --vgname=sysvg  --size=10000     --name=log_condor
logvol /var/lib/condor  --vgname=sysvg  --size=15000     --name=lib_condor
```

```
part swap  --asprimary --bytes-per-inode=4096 --fstype="swap" --ondisk=sdb --size=20000
part pv.02 --size=300000  --ondisk=sdb
volgroup squidvg pv.02
```

Kurulan Paketler: Yum Utils, OpenAfs Client, base-x, Emacs, editors, gnome-desktop, Developtment

Kurulan Temel Uygulamalar: fuse, fuse-libs, screen, vnc, sendmail, ntp, dnsmasq, dhcp, openssl-perl, openldap, openldap-clients, subversion, sysstat, xinetd, syslinux, httpd, ruby-rdoc, puppet

CVMFS kurulumu için Cern YUM repository verisinin sisteme eklenmesi gerekmektedir.

```
/etc/yum.repos.d/cern.repo
[cern]
name=Scientific Linux $releasever - $basearch
baseurl=http://linuxsoft.cern.ch/wlcg/sl6/$basearch/
enabled=1
gpgcheck=0
```

Repository verisinin eklenmesinin ardından HEP_OSLibs, cvmfs ve cvmfs-auto-setup uygulamaları kurulur.

```
yum install HEP_OSlibs_SL6.x86_64
yum     install     https://ecsft.cern.ch/dist/cvmfs/cvmfs-release/cvmfs-release-
latest.noarch.rpm
yum install cvmfs-2.3.5-1.el6 cvmfs-config-default-1.3-1 cvmfs-auto-setup-1.5-2
```

CVMFS için gerekli yapılandırma (/etc/cvmfs/default.local**)** dosyası aşağıdaki şekildedir.

```
CVMFS_REPOSITORIES=atlas.cern.ch,atlas-condb.cern.ch,atlas-
nightlies.cern.ch,sft.cern.ch
CVMFS_QUOTA_LIMIT=8000
CVMFS_HTTP_PROXY="http://squid.ct3.aydin.edu.tr:3128"
```

Yapılandırma dosyası tamamlandığında "cvmfs_config chksetup" komutu çalıştırılarak uygulama yapılandırması tamamlanır.

2.3.5 NFS sunucusu

NFS sunucu, Tier-3g sisteminde, dosyaların sunucular arasında veri paylaşımını ve büyük boyutlu dosyaların sunucu üzerinde tutulmadan erişilerek kullanımını sağlamaktadır. Aşağıda NFS sunucu için sunucu mantıksal depolama yapılandırması verilmiştir.

```
part /boot --asprimary --bytes-per-inode=4096 --fstype="ext3" --ondisk=sda --size=300

part pv.01 --size=1 --grow  --ondisk=sda
# LVM configuration so that we can resize / /var, /opt , /tmp
volgroup sysvg pv.01
logvol /                --vgname=sysvg  --size=20000     --name=sys             #   20 GB for system
logvol /tmp             --vgname=sysvg  --size=10000     --name=tmp             #   10 GB for tmp
logvol /var             --vgname=sysvg  --size=5000      --name=var             #   5 GB for var
logvol /opt             --vgname=sysvg  --size=5000      --name=opt             #   5 GB for opt
```

```
part swap  --asprimary --bytes-per-inode=4096 --fstype="swap" --ondisk=sdb --size=2000000
part pv.02 --size=1  --grow --ondisk=sdb
volgroup exportvg pv.02
logvol /export/home     --vgname=exportvg  --size=100000     --name=home
logvol /export/share/osg    --vgname=exportvg  --size=500000     --name=osg
logvol /export/share/atlas  --vgname=exportvg  --size=2000000     --name=atlas
logvol /export/share/condor-etc    --vgname=exportvg  --size=50000      --name=condor-etc
logvol /export/share/pilot    --vgname=exportvg  --size=2000000     --name=pilot
logvol /export/share/pandat3-output    --vgname=exportvg  --size=2000000     --name=pandat3
```

NFS sunucusu aynı zamanda "xrootd" (http://xrootd.org/) gibi ölçeklenebilir ve dağıtık yapıda çalışabilen dosya sistemleri uygulamaları ve "gridftp" (http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/) gibi geniş alan ağ sistemleri veri transfer protokol uygulamalarını da barındıracaktır.

Kurulan Paketler: base-x, Emacs, editors, gnome-desktop

Kurulan Temel Uygulamalar: sysstat, screen, vnc, sendmail, ntp, httpd, xfsdump, xfsprogs, openssl-perl, openldap, nss_ldap, openldap-clients, compat-readline43, subversion, gcc, glibc-devel, ruby-rdoc, puppet-server, puppet

2.3.6 INTERACTIVE sunucu

Interactive sunucu en basit ifadeyle; akademisyenlerin secure shell (ssh) yoluyla erişim sağlayarak konsol üzerinden INTERACTIVE sunucu üzerinde ya da (condor

ile) WORKER sunucu üzerinde analizlerini çalıştırmalarını sağladıkları sunucudur. Gerek sunucunun internet ağına açık (public network) üzerinde bulunması, gerek se sunucuya konsol üzerinden diğer kullanıcıların bağlanması nedeniyle bu sunucu üzerinde yapılacak yapılandırmaların doğruluğu büyük önem taşımaktadır. Sunucu yapılandırmasındaki hatalar sunucunun saldırılara açık hale gelmesine neden olabilecektir. Sunucu mantıksal sürücü yapılandırması aşağıda belirtilmiştir.

```
part /boot --asprimary --bytes-per-inode=4096 --fstype="ext3" --ondisk=sda --size=300
part swap  --asprimary --bytes-per-inode=4096 --fstype="swap" --ondisk=sda --size=36000

part pv.01 --size=1 --grow  --ondisk=sda
# LVM configuration so that we can resize /var, /opt , /tmp
volgroup sysvg pv.01
logvol /                --vgname=sysvg  --size=20000      --name=sys
logvol /tmp             --vgname=sysvg  --size=10000      --name=tmp            #   10 GB for tmp
logvol /var             --vgname=sysvg  --size=10000      --name=var            #   10 GB for var
logvol /opt             --vgname=sysvg  --size=10000      --name=opt            #   10 GB for opt
logvol /var/cache/cvmfs --vgname=sysvg  --size=25000      --name=cvmfs          #   25 GB for cvmfs mount
logvol /var/lib/condor   --vgname=sysvg --size=25000      --name=lib
logvol /var/log/condor   --vgname=sysvg --size=25000      --name=log
```

```
part pv.02 --size=1 --grow --ondisk=sdb
volgroup xrootdvg pv.02

# 0-550GB
logvol /local/xrootd/a  --vgname=xrootdvg --size=550000   --name=xrootd

# 0-4TB
#part /local/xrootd --asprimary  --bytes-per-inode=4096 --fstype="xfs"  --ondisk=sdb --size=4000000
```

Kurulan Paketler: Base-x, gnome-desktop, editors, engineering-and-scientific, graphical-internet, development-tools, x-software-development, gnome-software-development, legacy-software-development, Emacs

Kurulan Temel Uygulamalar: Alpine, anaconda, anaconda-runtime, bzip2-devel, cairo-devel.i386, compat-db, compat-gcc-34, compat-gcc-34-c++, compat-gcc-34-g77, compat-glibc, compat-glibc-headers, compat-libf2c-34, compat-libgcc-296, compat-libstdc++-296, compat-libstdc++-33, compat-openldap, compat-openldap.i386, compat-readline43, curl, curl-devel, cyrus-sasl-devel.i386, cyrus-sasl-gssapi, cyrus-sasl-md5, device-mapper-multipath, e2fsprogs-devel.i386, elfutils-libs.i386, expect, fipscheck, fuse, fuse-libs, fuse-devel, gcc44, gcc44-c++, gcc44-gfortran, giflib, giflib-devel, giflib.i386, gv, imake, kexec-tools, krb5-devel, krb5-libs, krb5-server, krb5-workstation, ksh, libgfortran, libsane-hpaio, libstdc++44-devel, libstdc++-devel, libXft.i386, mesa-libGLU, mesa-libGLU-devel, nss-devel, nss-devel.i386, nss_ldap, ntp, openldap, openldap-clients, openldap-devel, openldap-servers, openssh, openssl097a.x86_64, openssl-devel, openssl-devel.i386, pam_krb5, perl-Config-General, psutils, pycairo-devel.i386, python-devel, python-devel.i386,

python-numeric, qt, readline-devel, redhat-artwork.i386, rpm-devel, scrollkeeper.i386, sqlite-devel.i386, subversion, swig, system-config-netboot, system-config-netboot-cmd, tetex, tetex-dvips, tetex-fonts, tetex-latex, texinfo, tk.i386, vim-X11, xfig, xfsdump, xfsprogs, xinetd, xorg-x11-server-Xnest, xorg-x11-server-Xvfb, xorg-x11-utils, xorg-x11-xbitmaps, zlib-devel, zsh, libgfortran44.i386, libXp.i386, openmotif22.i386, openmotif22.x86_64, openmotif.i386, openssl097a.i386, ruby-rdoc, puppet

2.3.7 WORKER sunucu

WORKER sunucusu/sunucuları; INTERACTIVE sunucu üzerinden (condor ile) gönderilen analizleri yapıp, sonuçları INTERACTIVE sunucu üzerine geri gönderen sunuculardır. WORKER sunucu, bir sunucudan oluşabileceği gibi, birim zamanda yapılan analiz sayısına göre birden fazla sunucudan da oluşabilmektedir. Bu çalışmada, yapılandırma 07:00-22:59 saatleri arasında bir WORKER sunucu, 23:00-06:59 saatleri arasında üç WORKER sunucu hizmet verecek şekilde planlanmıştır. Aşağıda WORKER sunucu için oluşturulmuş örnek mantıksal sürücü yapılandırması bulunmaktadır.

```
part /boot --asprimary --bytes-per-inode=4096 --fstype="ext3" --ondisk=sda --size=300
part swap  --asprimary --bytes-per-inode=4096 --fstype="swap" --ondisk=sda --size=35000
part /     --asprimary --bytes-per-inode=4096 --fstype="ext3" --ondisk=sda --size=20000
part pv.01 --size=1 --grow  --ondisk=sda
# LVM configuration so that we can resize /var, /opt , /tmp
volgroup sysvg pv.01
logvol /tmp             --vgname=sysvg  --size=10000      --name=tmp             #   10 GB for tmp
logvol /var             --vgname=sysvg  --size=10000      --name=var             #   10 GB for var
logvol /opt             --vgname=sysvg  --size=10000      --name=opt             #   10 GB for opt
# area for cvmfs cache
logvol /var/cache/cvmfs --vgname=sysvg  --size=25000      --name=cvmfs           #   25 GB for cvmfs cache area

# LVM Configuration for areas required by condor
logvol /var/log/condor --vgname=sysvg --size=25000  --name=condor_log         #  25 GB for Condor logs
logvol /var/lib/condor --vgname=sysvg --size=35000 --name=condor_lib         # 350 GB for Condor exec areas
```

```
part pv.02 --size=1 --grow --ondisk=sdb
volgroup xrootdvg pv.02

#logvol /local/xrootd/a  --vgname=xrootdvg --size=5700000 --name=xrootd       # 5.7 TB xrootd
logvol /local/xrootd/a  --vgname=xrootdvg --size=1000000 --name=xrootd       # 1.0 TB xrootd
```

Kurulan Paketler: Base-x, gnome-desktop, editors, engineering-and-scientific, graphical-internet, development-tools, x-software-development, gnome-software-development, legacy-software-development,

Kurulan Temel Uygulamalar: Alpine, anaconda, anaconda-runtime, bzip2-devel, cairo-devel.i386, compat-db, compat-gcc-34, compat-gcc-34-c++, compat-gcc-34-g77, compat-glibc, compat-glibc-headers, compat-libf2c-34, compat-libgcc-296,

compat-libstdc++-296, compat-libstdc++-33, compat-openldap, compat-openldap.i386, compat-readline43, curl, curl-devel, cyrus-sasl-devel.i386, cyrus-sasl-gssapi, cyrus-sasl-md5, device-mapper-multipath, e2fsprogs-devel.i386, elfutils-libs.i386, expect, fipscheck, fuse, fuse-libs, fuse-devel, gcc44, gcc44-c++, gcc44-gfortran, giflib, giflib-devel, giflib.i386, gv, imake, kexec-tools, krb5-devel, krb5-libs, krb5-server, krb5-workstation, ksh, libgfortran, libsane-hpaio, libstdc++44-devel, libstdc++-devel, libXft.i386, mesa-libGLU, mesa-libGLU-devel, nss-devel, nss-devel.i386, nss_ldap, ntp, openldap, openldap-clients, openldap-devel, openldap-servers, openssh, openssl097a.x86_64, openssl-devel, openssl-devel.i386, pam_krb5, perl-Config-General, psutils, pycairo-devel.i386, python-devel, python-devel.i386, python-numeric, qt, readline-devel, redhat-artwork.i386, rpm-devel, scrollkeeper.i386, sqlite-devel.i386, subversion, swig, sysstat, system-config-netboot, system-config-netboot-cmd, tetex, tetex-dvips, tetex-fonts, tetex-latex, texinfo, tk.i386, vim-X11, xfig, xfsdump, xfsprogs, xinetd, xorg-x11-server-Xnest, xorg-x11-server-Xvfb, xorg-x11-utils, xorg-x11-xbitmaps, zlib-devel, zsh, , libgfortran44.i386, libXp.i386, openmotif22.i386, openmotif22.x86_64, openmotif.i386, openssl097a.i386, ruby-rdoc, puppet

2.4 Batch Servislerin Ayarlanması

Tier-3g sistemine ait sunucu kurulumları sonrası her bir sunucu için gerekli batch servisi yapılandırılmaları gerekir. CERN dokümantasyonunda her bir sunucu için özel olarak hazırlanmış yapılandırma betik dosyası bulunmaktadır. İlgili betik dosyaları http://svnweb.cern.ch/world/wsvn/atustier3 adresinden alınabilir. Her bir sunucu için gerekli betik dosyası çalıştırılmadan önce ilgili yapılandırma dosyasının güncellenmesi gerekir. Yapılandırma dosyaları, başta güvenlik duvarı, yerel kullanıcı ayarları, nfsv4 bağlantısı, ldap yetkilendirme, condor yapılandırması, cvmfs bağlantısı, kerberos ve gerekli kütüphanelerin kurulması gibi işlemlerin otomatik olarak yapılmasını sağlarlar.

2.4.1 Genel yapılandırma ayarları

Batch servislerin kurulumda kullanılmak üzere SVN sunucusundan alınan dosyalar, yapacakları işlere göre kategorilendirilmiştir. Tüm kurulum betikleri temel yapılandırma bilgilerinin bulunduğu ve EK A.'da yer alan Şekil A2'deki "node-environment.config" dosyasını kullaranarak işlemlerini yapar. "node-

environment.config" dosyası içerisinde; Ganglia Cluster ismi (GANGLIA_CLUSTER), ağ domain adresi (NETWORK_DOMAIN_NAME), Condor Headnode adresi (CONDOR_HEADNODE), nfs server kısa adı (NFS_SERVER_SHORT_NAME), ldap sunucu adresi (LDAP_SERVER_NAME), root e-posta adresi (ROOT_EMAIL_ADDR), Squid Proxy adresi (SQUID_NAME), sunucuya ait ip adresi (GMETAD_IP) değişkenleri ile, hata durumunda kurulumu durduran "pause_script_here", svn sunucudan istenen kurulum dosyasını indiren "fetch_shell_script", svn sunucudan istenen yapılandırma dosyasını indiren "fetch_config_file", sunucuya ait ip adresini bulan "get_private_ip" ve sunucu ethernet cihaz bilgilerini alan "get_private_ethernet_device" shell fonksiyonları bulunmaktadır.

Tüm sunucuların birbirleri ile haberleşebilmeleri için gerekli ip yapılandırma bilgileri EK A.'da yer alan Şekil A3'teki "private-network.config" dosyasında yer almaktadır. "private-network.config" dosyası içeriği temel olarak sunucu sisteminde yer alan sunucuların domain adları ve ip adres verilerini içerir (ör: head head 91.239.204.43).

EK A.-Şekil A4'te bulunan "create_local_users.sh" betik dosyası; XRootD dağıtık depolama sistemi ve Condor için gerekli yerel kullanıcıların oluşturulmasını, "condor" ve "xrootd" yerel kullanıcılarının otomatik olarak açılmasını ve gerekli yetkilendirme ve kısıtlama işlemlerinin yapılmasını sağlar.

Tier-3g sisteminde yer alan sunuculardaki saat farkı oluşmaması için her sunucuda ntp hizmeti çalışmaktadır. EK A.-Şekil A5'te bulunan "mail_alias.sh" betik dosyası ile ntp hizmeti, zamanlanmış görev hizmeti (crond) ve e-posta gönderim hizmeti (sendmail) sunucu başlangıcında otomatik çalışacak şekilde ayarlanır. Ayrıca "node-environment.config" dosyasında ROOT_EMAIL_ADDR değişkeni ile tanımlanan sunucu yöneticisi e-posta adresi her sunucu için "/etc/aliases" içerisine eklenerek tüm sistem mesajlarının sunucu yöneticisine e-posta olarak gönderilmesini sağlar.

Tier-3g sisteminde konsol bağlantılarında kullanılacak LDAP yetkilendirme yapılandırması EK A.-Şekil A6'da bulunan "ldap_client_setup.sh" betik dosyası ile oluşturulur. İlgili betik dosyası ile konsol kullanıcı bilgileri Tier-3g sisteminde yer alan LDAP sunucusundan sorgulanır. Böylece tüm sunucular için kullanıcı yönetimi tek bir sunucu üzerinden kontrol edilebilmektedir. Kullanıcı etkileşimli betik dosyası ile gerekli parametreler (ldap sunucu adresi gibi) konsoldan girilerek kurulum tamamlanır.

2.4.2 SQUID Proxy sunucusu

SQUID Proxy sunucusu için kurulum sonrası batch servis ayar betik dosyası "post-install-squidvm-configuration.sh", EK A.'da yer alan Şekil A8'de verilmiştir. İlgili betik dosyası ile sırasıyla aşağıdaki işlemler yapılır;

- 2.4.1'de yapısı anlatılan "private-network.config" dosyası sunucuya tanıtılarak sunucular arası haberleşmenin sunucu isimleri ile yapılması sağlanır.
- Proxy hizmeti sunacak sunucunun ilgili portlardan yayın yapabilmesi, bağlantı kurabilmesi ve bağlantıları kabul edebilmesi için güvenlik duvarında gerekli istisnalar tanımlanır.
- 2.4.1'de anlatıldığı şekilde, XRootD ve Condor için gerekli yerel kullanıcı hesapları oluşturulur.
- EK A. Şekil A9'da verilen Ganglia (http://agm.cern.ch/) kurulumu ve EK A. Şekil A10'da verilen gmond-istemci yapılandırması yapılır.
- SQUID Proxy sunucusu için gerekli yerel kullanıcı ve gruplar (squid:squid) oluşturularak Frontier Squid hizmeti kurulur. Hizmet için gerekli önbellek klasör (/var/cache/squid) yetkileri tanımlanır.
- NssSwitch yapılandırması, Tier-3g için oluşturulmuş şablon yapılandırma ile yeniden oluşturulur.
- İşletim sistemindeki tüm uygulamalar için güncellemeler kontrol edilerek kurulum tamamlanır.

2.4.3 LDAP sunucusu

LDAP sunucusu için kurulum sonrası batch servis ayar betik dosyası EK A.'da yer alan Şekil A11'da verilmiştir. İlgili betik dosyası sırasıyla;

- 2.4.1'de yapısı anlatılan "private-network.config" dosyası sunucuya tanıtılarak sunucular arası haberleşmenin sunucu isimleri ile yapılması sağlanır.
- LDAP hizmeti verecek sunucunun ilgili portlardan yayın yapabilmesi, bağlantı kurabilmesi ve bağlantıları kabul edebilmesi için güvenlik duvarında gerekli istisnalar tanımlanır.

- 2.4.1'de anlatıldığı şekilde, XRootD ve Condor için gerekli yerel kullanıcı hesapları oluşturulur.
- EK A. Şekil A9'da verilen Ganglia (http://agm.cern.ch/) kurulumu ve EK A. Şekil A10'da verilen gmond-istemci yapılandırması yapılır.
- EK A. Şekil A14'de verilen ldap_server_setup.sh betik dosyası LDAP sunucu hizmeti kurulumu yapılır.
- LDAP hizmetinde kullanılacak gerekli kullanıcı hesaplarının oluşturulabilmesi için EK A. Şekil A13'de verilen ldap_adduser.sh betik dosyası çalıştırılır.
- LDAP sunucuna ait slapd ayar dosyası ve kullanıcı veritabanının yedeği oluşturularak, "/root/ldap/backup" klasörüne gzip sıkıştırılmış dosyası olarak kaydedilir.
- NssSwitch yapılandırması, Tier-3g için oluşturulmuş şablon yapılandırma ile yeniden oluşturulur.
- İşletim sistemindeki tüm uygulamalar için güncellemeler kontrol edilerek kurulum tamamlanır.

2.4.4 HEAD sunucusu

HEAD sunucusu için kurulum sonrası batch servis ayar betik dosyası EK A.'da yer alan Şekil A15'de verilmiştir. İlgili betik dosyası sırasıyla;

- 2.4.1'de yapısı anlatılan "private-network.config" dosyası sunucuya tanıtılarak sunucular arası haberleşmenin sunucu isimleri ile yapılması sağlanır.
- HEAD sunucusu için gerekli güvenlik duvarı istisnaları tanımlanır.
- 2.4.1'de anlatıldığı şekilde, XRootD ve Condor için gerekli yerel kullanıcı hesapları oluşturulur.
- EK A. Şekil A16'da verilen "increase_file_limits.sh" betik dosyası ile XRootD için işletim sistemi dosya limitleri arttırılır (soft: 16384, hard:63536).
- Genel Ağ'a açık olan HEAD sunucusunda "root", "atlasadmin" ve "xrootd" yerel kullanıcıları için erişim kısıtları tanımlanır.
- Ağ paylaşım sistemi bağlantısı için "NfsV4 istemcisi" kurulumu yapılır.

- EK A. Şekil A9'da verilen Ganglia (http://agm.cern.ch/) kurulumu ve EK A. Şekil A10'da verilen gmond-istemci yapılandırması yapılır.
- EK A. Şekil A17'de verilen "headnode-condor.sh" betik dosyası ile YUM indirme kütüphanesine Condor kütüphanesi eklenerek Condor kurulumu yapılır.
- EK A. Şekil A18'de verilen "create-condor-credential-file.sh" betik dosyası ile condor kümesindeki tüm sunuculara gerekli yapılandırma yüklenir.
- EK A.-Şekil A6'da bulunan "ldap_client_setup.sh" betik dosyası LDAP istemcisi yüklenir.
- EK A.-Şekil A19'da bulunan "make-xrootd_ file_mounts_ headnode.sh" betik dosyası ile XRootD için gerekli dosya bağları oluşturulur.

2.4.5 NFS sunucusu

NFS sunucusu için kurulum sonrası batch servis ayar betik dosyası EK A.'da yer alan Şekil A20'de verilmiştir. İlgili betik dosyası sırasıyla;

- 2.4.1'de yapısı anlatılan "private-network.config" dosyası sunucuya tanıtılarak sunucular arası haberleşmenin sunucu isimleri ile yapılması sağlanır.
- NFS sunucusu için gerekli güvenlik duvarı istisnaları tanımlanır.
- 2.4.1'de anlatıldığı şekilde, XRootD ve Condor için gerekli yerel kullanıcı hesapları oluşturulur.
- İşletim sisteminde yer alan kullanıcılar için kullanıcı klasörleri oluşturulur.
- EK A. Şekil 21'de bulunan "configure-nfsv4-public.sh" betik dosyası ile NfsV4 hizmeti kurulumu ve gerekli yapılandırma işlemleri yapılır.
- EK A. Şekil A9'da verilen Ganglia (http://agm.cern.ch/) kurulumu ve EK A. Şekil A10'da verilen gmond-istemci yapılandırması yapılır.
- Genel Ağ'a açık olan NFS sunucusunda "root", "atlasadmin" ve "xrootd" yerel kullanıcıları için erişim kısıtları tanımlanır.
- EK A. Şekil 22'de bulunan "nfsnode-condor-public.sh" betik dosyası ile NFS sunucusu için gerekli Condor küme yapılandırmaları yapılır.

- Opsiyonel olarak DDM betikleri çalıştırılır.

2.4.6 INTERACTIVE sunucu

INTERACTIVE sunucu için kurulum sonrası batch servis ayar betik dosyası EK A.'da yer alan Şekil A23'de verilmiştir. İlgili betik dosyası sırasıyla;

- 2.4.1'de yapısı anlatılan "private-network.config" dosyası sunucuya tanıtılarak sunucular arası haberleşmenin sunucu isimleri ile yapılması sağlanır.
- INTERACTIVE sunucu için gerekli güvenlik duvarı istisnaları tanımlanır.
- 2.3.6'da belirtilen paketlerin yanında analizler için gerekli kütüphaneler indirilerek kurulur.
- Kerberos yapılandırma ayarları Tier-3g için gerekli şekilde güncellenir.
- EK A. Şekil 26'da bulunan "configure-cvmfs.sh" betik dosyası ile CVMFS(https://cernvm.cern.ch/) kurulum ve yapılandırması tamamlanır.
- EK A. Şekil A9'da verilen Ganglia (http://agm.cern.ch/) kurulumu ve EK A. Şekil A10'da verilen gmond-istemci yapılandırması yapılır.
- 2.4.1'de anlatıldığı şekilde, XRootD ve Condor için gerekli yerel kullanıcı hesapları oluşturulur.
- EK A. Şekil 7'de bulunan "nfsv4-client.sh" betik dosyası ile NfsV4 istemcisi kurulur.
- EK A. Şekil 24'de bulunan "interactive-condor.sh" betik dosyası ile INTERACTIVE sunucu için gerekli Condor küme yapılandırmaları yapılır.

2.4.7 WORKER sunucu

WORKER sunucu için kurulum sonrası batch servis ayar betik dosyası EK A.'da yer alan Şekil A27'de verilmiştir. İlgili betik dosyası sırasıyla;

- 2.4.1'de yapısı anlatılan "private-network.config" dosyası sunucuya tanıtılarak sunucular arası haberleşmenin sunucu isimleri ile yapılması sağlanır.
- WORKER sunucu için gerekli güvenlik duvarı istisnaları tanımlanır.
- 2.3.7'de belirtilen paketlerin yanında analizler için gerekli kütüphaneler indirilerek kurulur.

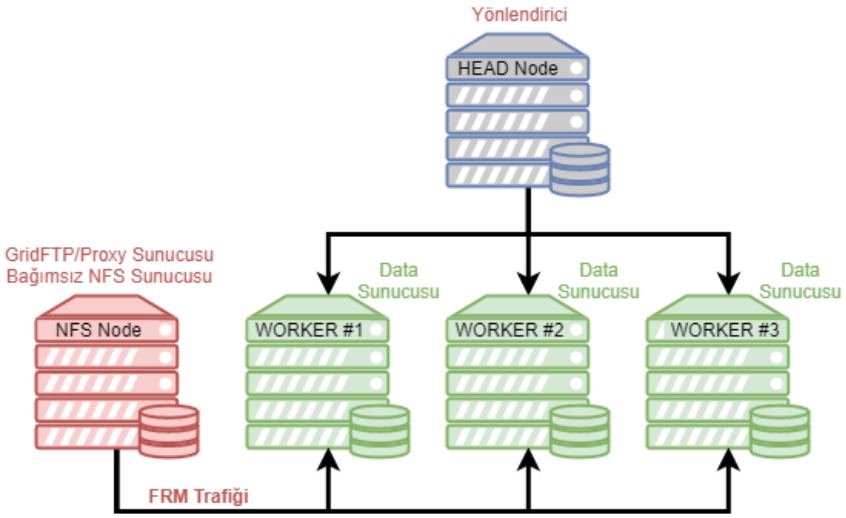
- Kerberos yapılandırma ayarları Tier-3g için gerekli şekilde güncellenir.
- EK A. Şekil A9'da verilen Ganglia (http://agm.cern.ch/) kurulumu ve EK A. Şekil A10'da verilen gmond-istemci yapılandırması yapılır.
- EK A. Şekil A16'da verilen "increase_file_limits.sh" betik dosyası ile XRootD için işletim sistemi dosya limitleri arttırılır (soft: 16384, hard:63536).
- WORKER sunucusunda "root", "atlasadmin" ve "xrootd" yerel kullanıcıları için erişim kısıtları tanımlanır.
- EK A. Şekil 7'de bulunan "nfsv4-client.sh" betik dosyası ile NfsV4 istemcisi kurulur.
- WORKER sunucu için gerekli Condor küme yapılandırmaları yapılır.

2.5 Dağıtık Depolama Servisleri Kurulumu

Tier-3g sistemi, standart grid sistemleri gibi dağıtık depolama hizmeti kullanmak üzere yapılandırılmıştır. CERN Tier-3g sisteminde dağıtık depolama servisi olarak xrootd kullanılmaktadır (https://twiki.cern.ch/twiki/bin/view/AtlasComputing/Tier3gXrootdSetup).

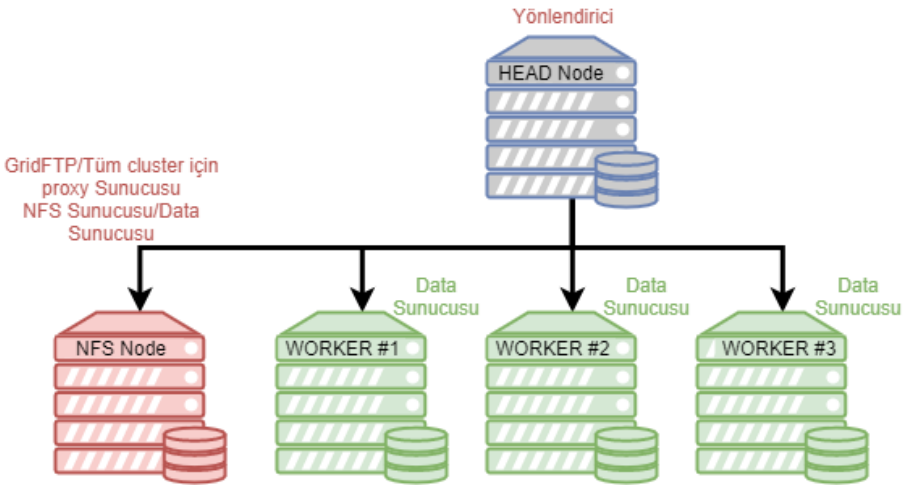
XRootD hizmeti HEAD sunucu üzerinde yönlendirici olarak, WORKER sunucular üzerinde ise veri sunucusu olarak hizmet verir. Böylece sistem üzerinde yer alan her bir WORKER sunucu, XRootD hizmetine ait bir veri sunucusu olarak tanımlanabilir.

XRootD Tier-3g sisteminde iki farklı şekilde yapılandırılabilir;



Şekil 2.2: GridFTP Yapılandırması #1

WORKER sunucu üzerindeki depolama alanı, NFS sunucu üzerindeki depolama alanından önemli ölçüde küçük ise, WORKER sunucu disk alanları XRootD sisteminde önbellek alanı olarak kullanılabilirler. Bu yapılandırma seçeneğinde dosyaların bulunduğu asıl sunucu NFS sunucusudur. Dosyaların NFS sunucudan WORKER sunuculara kopyalanma işlemi XRootD dosya konum yöneticisi(XRootD File Residency Manager-FRM) tarafından yapılır. GridFTP üzerinden cluster'a ulaşan veri NFS sunucusu üzerine kaydedilir. Eğer veri depolama sunucusu global XRootD birleşik depolama ağının bir parçası ise, verinin depolama ağı üzerinden sunulması işlemi proxy üzerinden gerçekleştirilir.



Şekil 2.3: GridFTP Yapılandırması #2

Tüm XRootD veri sunucularını bir arada kullanmak bir önceki yapılandırmaya göre daha kolaydır. Bu yapılandırma WORKER sunucular üzerinde yeterince alan bulunması durumunda kullanım açısından daha doğru bir seçimdir. Bu yapılandırmada; veri GridFTP proxy sunucusu üzerinden sisteme girer ve her bir node üzerinde dağıtılır. Eğer veri depolama sunucusu global XRootD birleşik depolama ağının bir parçası ise, verinin depolama ağı üzerinden sunulması işlemi proxy üzerinden gerçekleştirilir.

2.6 ATLAS Yazılımı ve Kaydı

ATLAS yazılımının sunucu sisteminde çalıştırılabilmesi için, kullanılan uygulamaların grid üzerinde yetkilendirilmesi ve doğrulamasının yapılması gerekmektedir. GridFTP ve Rucio uygulamalarının sertifikasyonu için gerekli olan sunucu sertifikası (host certificate), Ulusal Akademik Ağ ve Bilim Merkezi Türk Ulusal e-Bilim e-Altyapısı (TRUBA:http://www.grid.org.tr) tarafından sağlanmakta

olup, bu çalışmada "C=TR, O=TRGrid, OU=Istanbul Aydin, CN=gftp.ct3.aydin.edu.tr" adına "C=TR, O=TRGrid, CN=TR-Grid CA" sertifika otoritesi tarafından imzalanmıştır.

ATLAS yazılımının ATLAS Grid'i kayıt kontrolleri, ATLAS Grid Information System (AGIS: http://atlas-agis.cern.ch) üzerinden gerçekleştirilmektedir.

Bu çalışmada, "ATLAS yazılımları" ifadesi ile sadece grid üzerinde kayıtlı araştırmacılara kısıtlı veya tam yetkili olarak sunulmuş açık kaynak kodlu kodlamalar kastedilmektedir. Grid üzerinde yer almayan, detektör kalibrasyon ve kontrol sistem yazılımları ve arayüzleri gibi yazılımlar da ATLAS yazılımları dahilinde erişilebilse dahi, aynı tanıma dahil edilmemiştir. Buna göre, ATLAS yazılımları;

- **Dağıtık Veri Yönetim Sistemi:** (Data Distributed Management System – DDM/DQ2)
- **Üretim / Analiz Sistemleri:** (Production And Data Analysis System - PANDA)
- **Veri Analiz Kodları:** Bkz 1.4.2. Kod Yapısı

olmak üzere 3 kategoriye ayrılmaktadır.

2.7 Analiz Kodunun Test Edilmesi ve Yük Dengeleme

Bu bölümde Tier-3 sisteminin kurulumu sonrasında kurulu hesaplama gücü ve performans ölçümü için basic event loop, benchmark full cut stream event loop ve eos disk performans testlerinin uygulamaları anlatılmış olup, sanal sunucu sistemi ile kurulan Tier-3g sisteminde uygulanan yük denheleme sistemi açıklanmıştır.

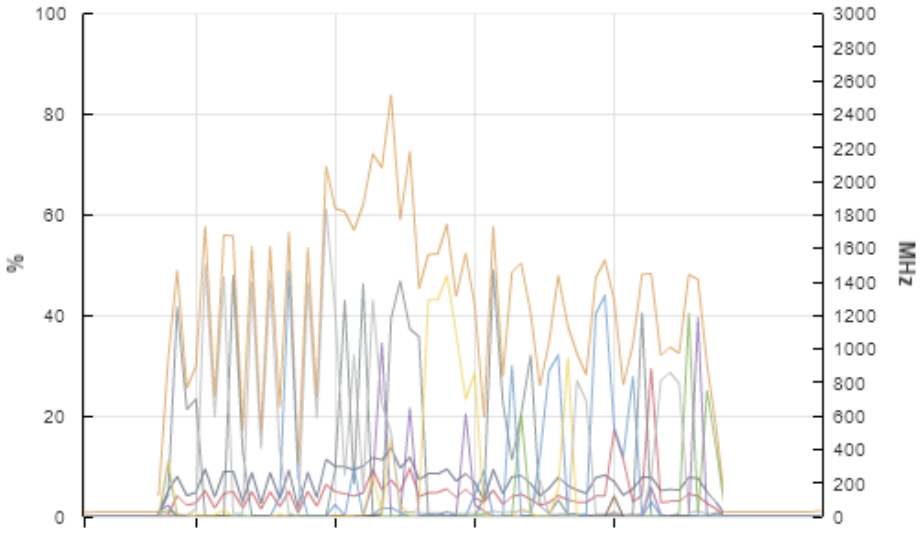
2.7.1 Analiz kodunun test edilmesi

Aşağıda oluşturulan Tier-3g sisteminin performansı ve teknik gücünün test edilebilmesi amacıyla yapılan testler yer almaktadır. Testlere ait ölçülebilir sonuçlara ait grafikler ilgili testin altında gösterilmiştir. Test amaçlı da olsa hiçbir şekilde CERN deney verisi kullanılmamış, simülasyon ve test verileri üzerinden çalışılmıştır.

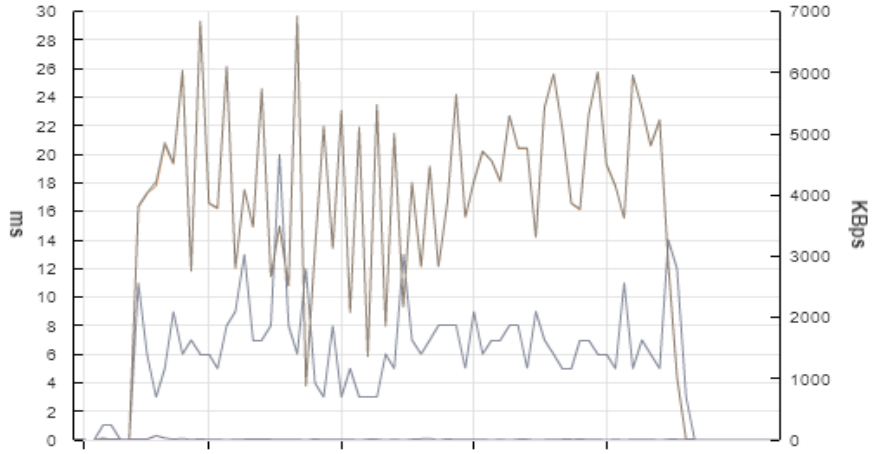
- **Basic Event Loop Testi:** RootCore Base-2.5.1 koduyla, AOD formatında veri dosyasına erişilerek, antikt-jet algoritması ve kinematik histogramların root

kütüğüne flat n-tuple olarak yazdırılmasından oluşan kod interactive sunucu sisteminde test edilmiştir.

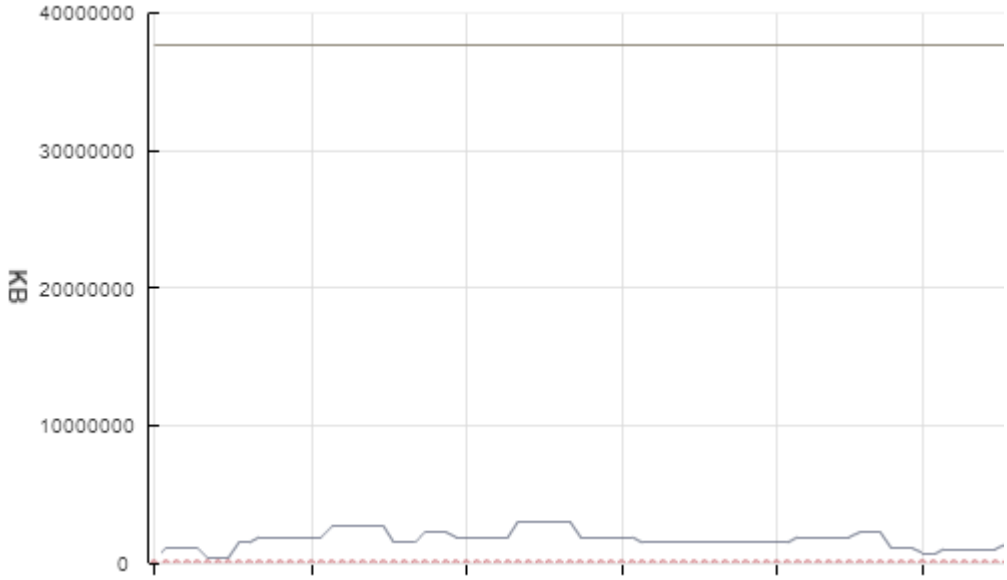
RootCore Base-2.5.1 koduyla yapılan single core testi 593,74 saniye sürmüş ve test sırasındaki CPU, RAM, Disk ve Network yoğunluk grafikleri aşağıdaki gibi gözlemlenmiştir. Multi core (48 CPU) test 127 saniye sürmüş olup orantılı benzer CPU, RAM, Disk ve Network yoğunlukları gözlemlenmiştir.



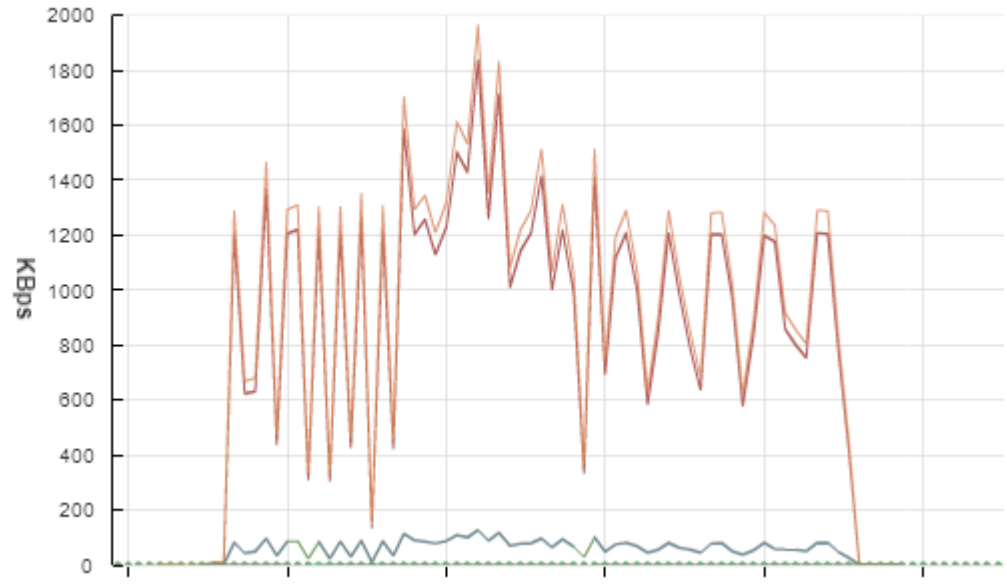
Şekil 2.4: Basic Event Loop Testi CPU Cost ve Frekans Grafiği (Single Core)



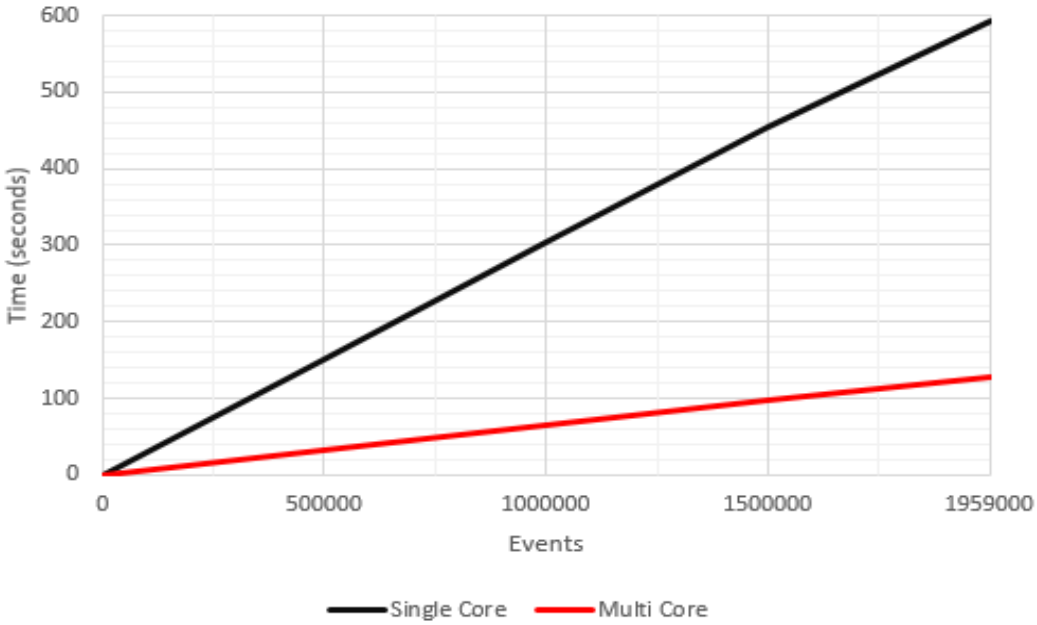
Şekil 2.5: Basic Event Loop Testi Disk Kullanımı ve Gecikme Grafiği (Single Core)



Şekil 2.6: Basic Event Loop Testi Memory Kullanım Grafiği (Single Core)



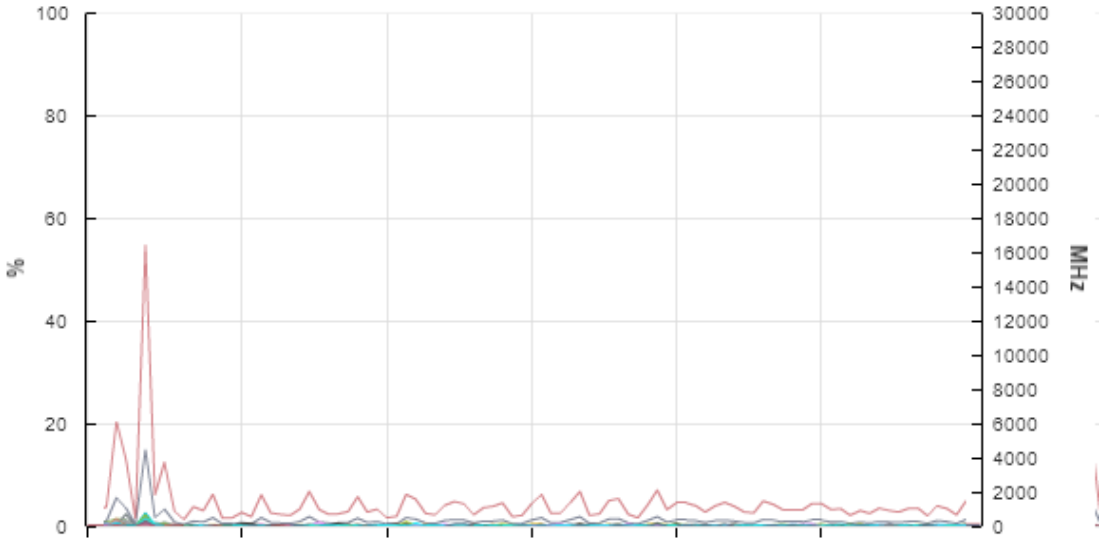
Şekil 2.7: Basic Event Loop Testi Network Kullanım Grafiği (Single Core)



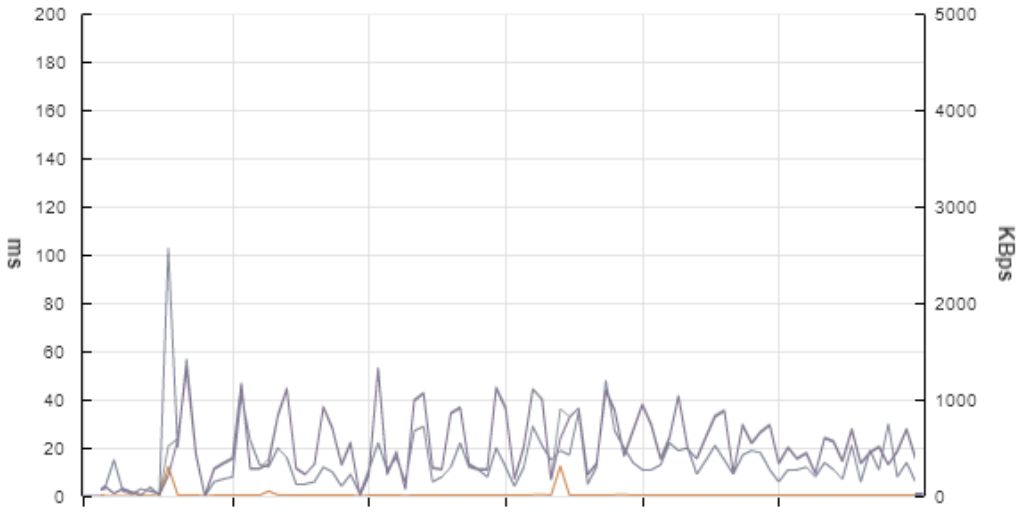
Şekil 2.8: Basic Event Loop Testi Single-Multi Core Çalışma Süreleri

- **Benchmark Full Cut Stream Event Loop Testi:** Rootcore Base-2.4.28a kullanılarak ATLAS analizlerinden "Single Lepton Vector-like Quark" araştırması için hazırlanan kod interactive sisteminde test edilmiştir.

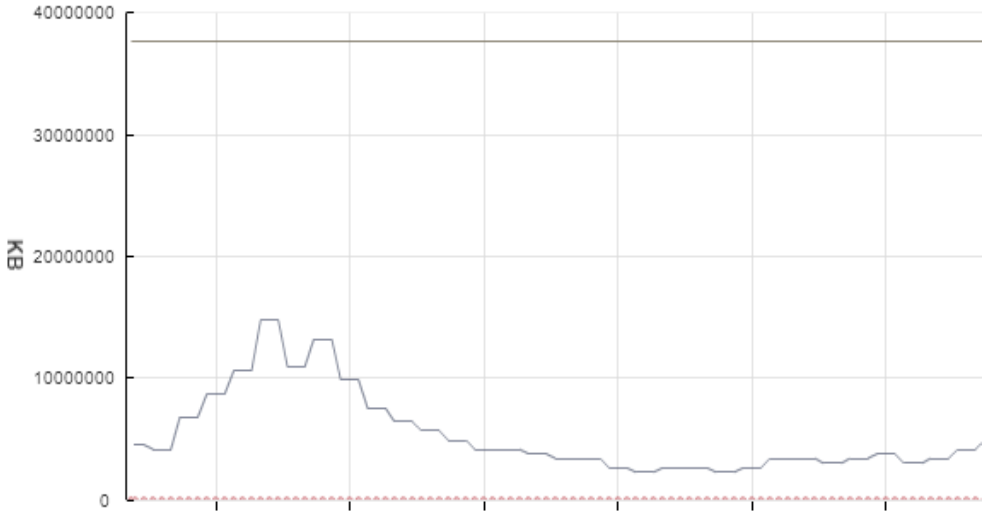
- **EOS Disk Performans Testi:** Daha önce RootCore Base 2.5.1 kullanılarak yapılmış olan Basic Event-Loop Testinin benzeri tekrarlanmış, veri dosyasına erişim CERN/EOS disk sistemi üzerinden gerçekleştirilmiştir. EOS disk sistemi ve test verisi, sunucularımıza mount edilerek test sırasında; 742.7 GB hacmindeki veri 2006 sn'lik sürede 48 çekirdek kullanılarak (multicore) işlenmiştir.



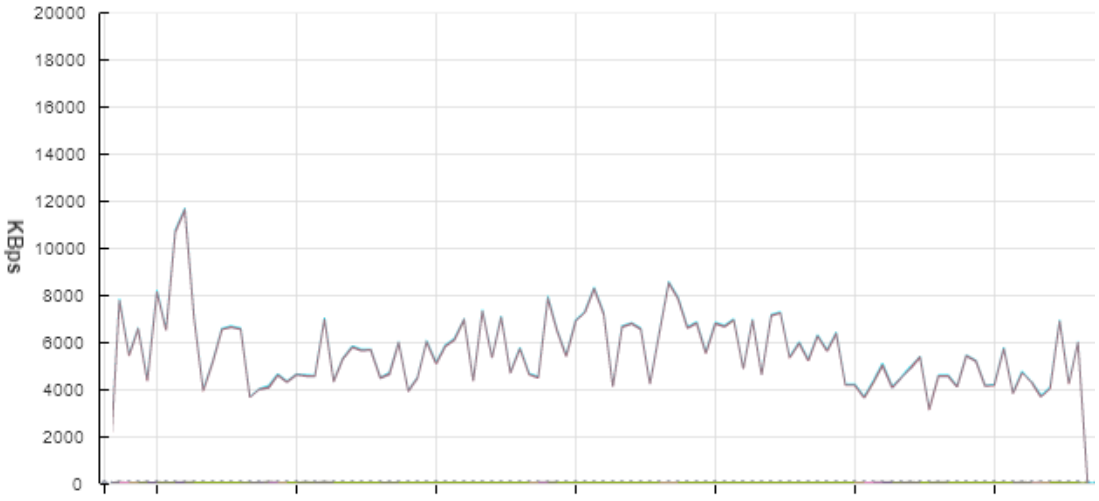
Şekil 2.9: EOS Disk Performans Testi CPU Cost ve Frekans Grafiği



Şekil 2.10: EOS Disk Performans Testi Disk Kullanımı ve Gecikme Grafiği



Şekil 2.11: EOS Disk Performans Testi Memory Kullanım Grafiği



Şekil 2.12: EOS Disk Performans Testi Network Kullanım Grafiği

2.7.2 Yük dengeleme

İstanbul Aydın Üniversitesi CERN Tier-3g sisteminde yük dengeleme sistemi olarak VMWare vMotion ve DRS (Distributed Resource Scheduler- Dağıtık Kaynak Planlayıcısı) teknolojileri kullanılmıştır.

vMotion bir sanal makinenin, iki hostun ortak paylaşımlı bir hafızası olmasa bile, veri deposu ve hostunu eş zamanlı olarak değiştirebilmesine olanak sağlamaktadır. İki host, network bağlantısına sahip olduğu sürece, canlı olarak, sanal makineler otomatik ya da elle taşınabilmektedir.

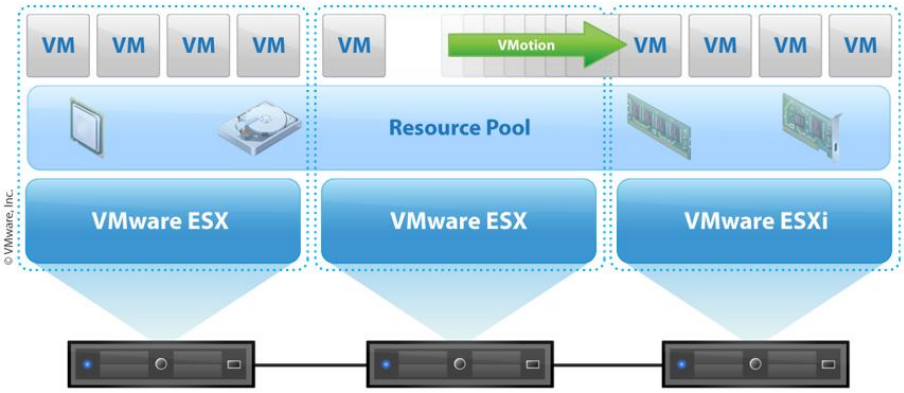
Büyük veri merkezleri aralarında ortak paylaşımlı bir bellek olmasa bile cluster arasında sanal makineleri taşıyabiliyorlarken, sunucuların yerel disklerini geliştirmelerine ve daha basit ayarlamalar kullanmalarına da olanak tanımaktadır.

Aslında bu teknoloji vMotion ve Storage vMotion'ı kombine etmektedir. Ama hem hesap durumunu bir başka hosta hem de diskleri diğer veri deposuna kopyalamak yerine hesap durumu ve diskin başka host ve veri deposuna transferi işlemi birleştirilmiş bir taşımadır.

vMotion aşağıdaki adımları izleyerek bir sanal makinayı bir fiziksel sunucudan diğerine taşır;

- Kaynak sanal sunucunun hedef sunucuda çalıştırılabilir olduğu kontrol edilir.
- Hedef sistemde kaynak sunucu için kullanılacak fiziksel kaynaklar rezerve edilmek üzere yeni bir VM işlemi başlatılır.
- Kaynak sunucuda bellek noktası oluşturulur. Bu andan itibaren kaynak sunucudaki tüm veri değişimleri ayrı bir alana yazılmaya başlar.
- Oluşturulan bellek noktasına ait veri hedef sunucuya gönderilmeye başlanır.
- Değişim alanındaki veriler minimize edilene kadar bu işlem tekrarlanır.
- Kaynak sunucudaki CPU durdurulur.
- Değişim alanında kalan veri hedef sunucuya gönderilir. Bu işlem milisaniyeler içerisinde tamamlanır.
- Kaynak sunucudaki vMotion işlemi fiziksel switch'e ters ARP paketlerinin gönderimi ile sona erer. Hard disk erişimi hedef ESX üzerine alınır.

- Kaynak sanal makine kapatılır ve kaynak sunucudaki veriler silinerek hedef sunucu devreye alınır.



Şekil 2.13: DRS vMotion

DRS sistemi vMotion'ı kullanarak fiziksel host(lar) üzerinde yapılandırılan sunucular (anlatılan sistemdeki INT#2 ve INT#3 hariç tüm sunucular) arasında CPU ve RAM dengesi sağlar. Böylece CPU ve RAM kaynağına ihtiyaç duymayan sunucular için ayrılmış kaynaklar ihtiyacı olan diğer sunucuların hizmetine sunulur.

Bu çalışmada ayrıca, bilgisayar laboratuvarlarının kullanılmadığı saatlerde devreye alınan yeni host sayesinde ilgili sunucular gerektiğinde DRS ve vMotion tarafından diğer fiziksel sunucuya taşınarak ek fiziksel CPU ve RAM kaynağına ulaşması planlanmaktadır.

3. SONUÇ VE TARTIŞMA

Tier-3g sistemleri, WLCG ağına bağlı olması sayesinde ATLAS verilerine ulaşılabilmesini, yoğun işlemci gücü gerektiren deney verisi bazlı araştırma hesaplamalarında enstitülerin veya araştırma merkezlerinin kendi kaynaklarını kullanabilmesini, dolayısıyla araştırmacıların veri ağı işlemci gücü ve yoğunluğundan bağımsız olarak çalışmalarını yapmasına olanak sağlamaktadır.

Bu çalışmada; dünyanın başlıca deney merkezlerinden Avrupa Nükleer Araştırmalar Merkezi (CERN) tarafından yürütülen ATLAS deneyleri sonucu oluşan verilerin, enstitüler veya araştırma grupları özelinde kullanılarak araştırmacıların hesaplamalarını yapmasını sağlayan ve WLCG ağının özerk bir parçası olan Tier-3g sistemini oluşturan HEAD, SQUID, LDAP, NFS, INERACTIVE ve WORKER sunucularının planlamadan kullanıma kadar olan kurulum süreçleri işlenmiş, kurulum ve bakım gibi yazılım/donanım kaynaklı süreçlerinin tanımlanmaları ile ilgili bilgilere yer verilmiştir.

CERN merkezli hazırlanan ve Tier-3g sistemlerin oluşturulmasını kolaylaştırmayı hedefleyen kurulum dosyalarının güncelliğini yitirmiş olması nedeniyle, başta işletim sistemi ve bağlı jenerik sistemler olmak üzere kurulum ile ilgili yapılandırma dosyaları ve değişkenlerinin görevleri tanıtılmış, sistemin kararlı çalışabilmesi için gerekli işletim sistemi ve jenerik sistem sürümleri belirlenerek kurulum için yapılması gereken güncellemeler anlatılmıştır. Tier-3g sisteminde kullanılan uygulamalar, genel itibariyle CERN için oluşturulmuş ya da CERN için özelleştirilmiş olmaları nedeniyle varolan en yeni işletim sistemleri ile kararlı çalışamayabilmektedir. Bu güncellemeler ile CERN sistemine uyarlanmış uygulamaların tamamının kararlı şekilde çalışabilmesi sağlanmıştır.

Sistem güvenliği için, CERN tarafından belirlenmiş olan “root kullanıcılarının erişimlerinin kısıtlanması”, “servislerin, oluşturulan yerel kullanıcılar ile çalıştırılması” ve “servis kullanıcılarının konsol erişimlerinin kısıtlanması” gibi merkezi güvenlik önlemlerinin yanısıra, her bir sunucuya ait erişim şablonlarının

oluşturulması ve gerekli erişim kısıtlamalarının güvenlik duvarı ile oluşturulması gibi ek önlemler alınmıştır. Bu sayede sistem üzerinde öngörülmeyen erişim açıklarının oluşmasının engellenmesi hedeflenmiştir.

Sistem içerisinde bulunan sunuculara ait yedekler, sanal sunucu sistemi üzerinde bulunan VM Snapshot özelliği kullanılarak, sunucu çalışır durumdayken alınmaktadır. Arttırımlı yedekleme teknolojisi ile sunuculara ait yedekler, günlük olarak oluşturulmakta ve istenilen tarihteki çalışır sunucu durumuna geri dönülebilmektedir.

Kurulumda, İstanbul Aydın Üniversitesi Bilgi İşlem Daire Başkanlığı'nın sunucu yönetimi için hali hazırda kullanmakta olduğu VMware sunucu sanallaştırma sistemi kullanılmıştır. İşletim sistemi katmanında temel farklılıkları bulunmayan sanal sunucu sistemlerinin, fiziksel sistemlere göre bir çok avantajı mevcuttur. Aşağıdaki özellikler sanal sunucu sistemlerinin, klasik fiziksel sunucu sistemlerine göre başlıca avantajları arasında yer almaktadır.

- Sunucu oluşturma işlemleri, donanım-yazılım bakım ve yönetimi, sunucu taşıma,
- Yedekleme, yedekten geri dönme ve felaket kurtarma işlemleri,
- Gelişmiş yük dengeleme ve kaynak paylaşımı.

Sunuculararası kaynak paylaşımını sağlayan yük dengeleme teknolojisi ile kurulumu yapılan Tier-3g sistemi içerisinde yer alan sunucuların kaynak kullanım verimliliklerinin arttırılması hedeflenmiştir. Bu teknoloji ile sistem içerisinde yeralan sunuculardan aktif CPU, RAM ve Disk kaynağı gibi ihtiyaçlar anlık olarak belirlenmekte, aktif işlem yapmayan sunuculara tahsis edilmiş kaynaklar, aktif işlem yapan ve daha fazla kaynak ihtiyacı duyan sunucuların kullanımına sunulmaktadır. İşletim sistemi katmanında farkedilemeyecek derecede hızla gerçekleşen bu işlem ile kullanılmayan sunucu kaynakları, ihtiyaç duyan sunucuların hizmetine sunulmakta, böylece sistem için tahsis edilmiş fiziksel teknolojik altyapının, tüm sistem genelinde maksimum verim ile kullanılmasının sağlanması hedeflenmektedir.

Ayrıca ileriki çalışmalarda; DRS teknolojisi kullanılarak, Tier-3g sistemi içerisinde yer almayan ve farklı amaçlar için kullanılan sunuculara ait boşta duran (idle) kaynakların, Tier-3g sisteminde kullanılmak üzere tahsis edilmesi sağlanabilir. Bunun yanı sıra belirli zaman aralıklarında farklı amaçlar için kullanılan sunucuların

kullanılmadıkları zaman dilimlerinde, Tier-3g sistemi içerisine dahil edilecek şekilde konfigüre edilmesi yoluyla analiz hesaplamalarına destek olmaları sağlanabilir. Bu ve benzeri yöntemlerin kullanıma sunulması, birim zamanda yapılan hesaplama miktarını, Tier-3g sistemine eklenen dış sunucu sayısı ile doğru orantılı olarak arttırır ve aynı zamanda kullanılmayan bilişim kaynaklarının da efektif bir şekilde kullanılarak bilişim altyapısı için harcanan mali kaynakların fiyat/performans oranının azaltılması sağlanabilir.

KAYNAKLAR

Adams, D. Barberis, D. Bee, C. P. Hawkings, R. Jarp, S. Jones, R. Malon, D. Poggioli, L. Poulard, G. Quarrie, D. Wenaus, T. (2004). The ATLAS Computing Model, Computing in High Energy Physics and Nuclear Physics, Interlaken, Switzerland, P.1007 (CERN-2005-002).

ATLAS Collaboration (Ueda, I. for the collaboration). (2012). ATLAS distributed computing operations in the first two years of data taking, PoS ISGC2012, 013.

ATLAS Collaboration Aad, G. et al. (2008). The ATLAS Experiment at the CERN Large Hadron Collider" - JINST 3 S08003.

Belov, S. et al. (2012). VM-based infrastructure for simulating different cluster and storage solutions used on ATLAS Tier-3 sites, J. Phys.: Conf. Ser. 396 042036.

G. Bell; J. Gray; A. Szalay. (2006). Petascale Computational Systems, Computer, Vol.39, Issue 1, PP.110-112.

Gonzalez de la Hoz, S. *et al.* (2008). Analysis Facility Infrastructure (Tier-3) for ATLAS Experiment, Eur. Phys. J. C. 54, PP.691–697.

Haupt, A., Kemp, Y. (2010). The NAF: National Analysis Facility at DESY, Journal of Physics Conference Series, Vol. 214, P.5.

Jones, R. W. L., & Barberis, D. (2010). The evolution of the ATLAS computing model. In Journal of Physics: Conference Series (Vol. 219, No. 7, p. 072037). IOP Publishing.

Villapana, M. *et al.* (2012). ATLAS Tier-3 within IFIC-Valencia analysis facility, Journal of Physics Conference Series, Vol. 396, p 4.

The ATLAS Computing Technical Design Report. (2005). ATLAS-TDR-017; CERN-LHCC-2005- 022.

Shiers, J. (2007). The Worldwide LHC Computing Grid, Computer Physics Communications, Vol. 177, Issues 1-2, PP. 219-223.

Internet Kaynakları:

Url-1 <https://home.cern/about/computing>, alındığı tarih: 25.04.2017

Url-2 <https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/WebHome>, alındığı tarih: 25.04.2017

Url-3 <https://www.egi.eu/about/>, alındığı tarih: 25.04.2017

Url-4 <http://www.taek.gov.tr/cern-sanal/741-cern-arastirmalar.html>, alındığı tarih: 02.05.2017

Url-5 <http://wlcg.web.cern.ch/>, alındığı tarih: 02.05.2017

Url-6 <http://svnweb.cern.ch/world/wsvn/atustier3>, alındığı tarih: 01.09.2017

Url-7 <http://www.vmware.pro/vsphere-5-1-vmotion-hakkinda>, alındığı tarih: 19.09.2017

Url-8 <https://pubs.vmware.com/vsphere-50/index.jsp?topic=%2Fcom.vmware.wssdk.pg.doc_50%2FPG_Ch13_Resources.15.6.html>, alındığı tarih: 19.09.2017

Url-9 <http://linux.web.cern.ch/linux/scientific6/>, alındığı tarih: 21.09.2017

Url-10 <https://www.scientificlinux.org/>, alındığı tarih: 21.09.2017

Url-11 <https://www.openldap.org/>, alındığı tarih: 28.09.2017

Url-12 <http://frontier.cern.ch/>, alındığı tarih: 01.11.2017

Url-13 <https://cernvm.cern.ch/portal/filesystem>, alındığı tarih: 01.11.2017

Url-14 <https://rucio.cern.ch/>, alındığı tarih: 01.11.2017

Url-15 <https://root.cern.ch/installing-xrootd>, alındığı tarih: 01.11.2017

EKLER

EK A: Kurulum betik dosyaları

```
addr="ldap.ct3.aydin.edu.tr"
base="dc=ct3,dc=aydin,dc=edu,dc=tr"
domaincomponent="ct3"

TMPFILE=`mktemp /tmp/database0.XXXXXXXXXX` || exit 1
cat - <<HereLdifEnds |sed 's?MYSERVERHERE?'$addr'?'|sed 's?MYBASEHERE?'$base'?'  |
sed 's?MYDOMAINCOMPONENT?'$domaincomponent'?' >$TMPFILE
dn: MYBASEHERE
dc: MYDOMAINCOMPONENT
objectClass: top
objectClass: domain

dn: ou=Hosts,MYBASEHERE
ou: Hosts
objectClass: top
objectClass: organizationalUnit

dn: ou=Rpc,MYBASEHERE
ou: Rpc
objectClass: top
objectClass: organizationalUnit

dn: ou=Services,MYBASEHERE
ou: Services
objectClass: top
objectClass: organizationalUnit

dn: nisMapName=netgroup.byuser,MYBASEHERE
nismapname: netgroup.byuser
objectClass: top
objectClass: nisMap

dn: ou=Mounts,MYBASEHERE
ou: Mounts
objectClass: top
objectClass: organizationalUnit

dn: ou=Networks,MYBASEHERE
ou: Networks
objectClass: top
objectClass: organizationalUnit

dn: ou=People,MYBASEHERE
ou: People
objectClass: top
objectClass: organizationalUnit

dn: ou=Group,MYBASEHERE
ou: Group
objectClass: top
objectClass: organizationalUnit

dn: ou=Netgroup,MYBASEHERE
ou: Netgroup
objectClass: top
objectClass: organizationalUnit

dn: ou=Protocols,MYBASEHERE
ou: Protocols
objectClass: top
objectClass: organizationalUnit

dn: ou=Aliases,MYBASEHERE
ou: Aliases
objectClass: top
objectClass: organizationalUnit

dn: nisMapName=netgroup.byhost,MYBASEHERE
nismapname: netgroup.byhost
objectClass: top
objectClass: nisMap

dn: uid=root,ou=People,MYBASEHERE
uid: root
cn: root
objectClass: account
objectClass: posixAccount
```

```
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}!!
shadowLastChange: 14568
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 0
gidNumber: 0
homeDirectory: $HOME

dn: uid=xrootd,ou=People,MYBASEHERE
uid: xrootd
cn: xrootd
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}!!
shadowLastChange: 14568
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1050
gidNumber: 1050
homeDirectory: /local/home/xrootd

dn: uid=condor,ou=People,MYBASEHERE
uid: condor
cn: condor
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}!!
shadowLastChange: 14568
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1055
gidNumber: 1055
homeDirectory: /local/home/condor

HereLdifEnds
```

Şekil A.1: OpenLDAP Temel Kullanıcı Kataloğu Betik Dosyası

```
#!/bin/bash
#  the system administrator for various site paramters save in a configuration
file
#-------------- Cluster specific information -----------

export GANGLIA CLUSTER="IAUASC"
export NETWORK DOMAIN NAME="ct3.aydin.edu.tr"
export CONDOR HEADNODE="head.ct3.aydin.edu.tr"
export NFS_SERVER_SHORT_NAME="nfs"export
LDAP_SERVER_NAME="ldap.ct3.aydin.edu.tr"export
ROOT EMAIL ADDR="agahalici@aydin.edu.tr"
export SQUID NAME="squid.ct3.aydin.edu.tr"
export GMETAD IP="91.239.204.44"      # NFS server


export PUBLIC INTERFACE="eth0"

if [ -e /etc/sysconfig/network-scripts/ifcfg-${PUBLIC INTERFACE} ] ; then
    export PUBLIC HW ADDR=`/sbin/ifconfig -a $PUBLIC INTERFACE | grep HWaddr | awk
'{print $5}'`
fi

function pause script here {
  file name=$1
  echo -n " Press enter after fixing $file name : "
  read ans
  return 0
}
```

```
function fetch shell script {
  file name=$1
  ret_code=0
  return 0

  if [ ! -e $HOME/post install/scripts/$file name ] ; then
     svn export
http://svnweb.cern.ch/guest/atustier3/post install/trunk/scripts/$file name
     ret_code=$?
     if [ "$ret_code" -ne 0 ]; then return $ret_code ; fi
     mv $file name $HOME/post install/scripts/
     ret code=$?
     if [ "$ret code" -ne 0 ]; then return $ret code ; fi

     chmod +x $HOME/post_install/scripts/$file_name
     ret_code=$?
     if [ "$ret code" -ne 0 ]; then return $ret code ; fi
  fi

  #pause script here $file name
  return $ret_code
}

function fetch config file {
file_name=$1
ret_code=0

if [ ! -e $HOME/post install/config files/$file name ] ; then
   svn export
http://svnweb.cern.ch/guest/atustier3/post install/trunk/config files/$file name
   ret_code=$?
   if [ "$ret_code" -ne 0 ]; then return $ret_code ; fi

   mv $file name $HOME/post install/config files/
   ret code=$?
   if [ "$ret code" -ne 0 ]; then return $ret code ; fi
fi

  #pause_script_here $file_name
  return $ret code
}

function get_private_ip {
echo -n " Please enter the private network short name (for example headprv) - "
read private name
grep $private name $HOME/post install/private-network.config >& /dev/null
ret code=RC$?
if [ $ret_code != "RC0" ] ; then
    echo
    echo "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   Error
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
    echo
    echo "   could not find IP address for $private name -  check the name that
you entered"
    echo
    return 1
fi

export PRIVATE IP=`grep  $private name  $HOME/post install/private-network.config
| awk '{print $3}'`

return 0
}

function get private ethernet device {
echo -n " Please enter the private network interface (for example eth1) - "
read ans
export PRIVATE INTERFACE=$ans
if [ ! -e /etc/sysconfig/network-scripts/ifcfg-${PRIVATE INTERFACE} ] ; then
    echo
    echo "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!   Error
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
    echo
    echo " Could not find the ethernet interface specified - $PRIVATE INTERFACE"
    echo
    return 1
fi
```

```
if [ -e /etc/sysconfig/network-scripts/ifcfg-${PRIVATE_INTERFACE} ] ; then
    export PRIVATE_HW_ADDR=`/sbin/ifconfig -a $PRIVATE_INTERFACE | grep HWaddr |
awk '{print $5}'`
fi
return 0
}
```

Şekil A.2: "node-environment.config" Dosyası

```
#
#   This configuration file contains network information and names
#

head            head   91.239.204.43
nfs             nfs    91.239.204.44
ldap            ldap    91.239.204.42
squid           squid   91.239.204.41
interactive     int   91.239.204.45
worker          wrk   91.239.204.46
```

Şekil A.3: "private-network.config" Dosyası

```
function create_local_user () {
name=$1
num=$2
/usr/sbin/groupadd -g $num $name
/usr/sbin/adduser -u $num -g $name -d /local/home/$name  $name
/bin/chown $name:$name /local/home/$name
/bin/chmod 755 /local/home/$name
}
if [ ! -e /local/home ] ; then mkdir -pv /local/home ; fi

create local user  condor 1055
create_local_user  xrootd 1050
```

Şekil A.4: "create_local_users.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post install/scripts:$PATH

# define the environmental variables for installation
source $HOME/post_install/node-environment.config

# Turn needed stuff on
/sbin/chkconfig ntpd --level 345 on
/sbin/chkconfig crond --level 345 on
/sbin/chkconfig sendmail --level 345 on
/bin/echo "root: "$ROOT_EMAIL_ADDR >> /etc/aliases
/usr/bin/newaliases
exit 0
```

Şekil A.5: "mail_alias.sh" Dosyası

```
#!/bin/bash

echo Backup copies of the old files will be saved in /var/tier3g
mkdir /var/tier3g
cp /etc/ldap.conf /var/tier3g/ldap nss.conf
echo Changing /etc/ldap.conf to use your server for
echo cluster information
echo

addr=$1

HOSTWAIT=10         # Specify up to 10 seconds for host query reply.
ipaddr=$(host -W $HOSTWAIT $addr | awk '{print $4}')  # Doing host lookup to get
IP address.
ipname=$(host -W $HOSTWAIT $ipaddr | awk '{print $5}')  # Doing host lookup to get
IP address.

full hostname=""
if [ $ipname == $addr. ]; then
```

```
   full hostname=$addr
else
   echo " Error - $addr is not a full qualified server name  $addr .ne. $ipname "
   exit 1
fi

echo -n "Enter your ldap server's fully qualified domain name (in lower case):  "
read addr
echo -n " The fully qualified server is $addr (y/n)? "
read ans
if [ $ans != "y" ]; then
  echo Come back soon\!
  exit 0
fi

host=`echo $addr |sed 's?\..*$??'`
base=`echo $addr |sed 's?^'$host'??'|sed 's?\.?,dc=?g'|sed 's?^,??'`
echo ldap base is being set to $base

if [ ! -e /etc/openldap/cacerts ] ; then mkdir -p /etc/openldap/cacerts ; fi

cat - <<HereLdapConfEnds | sed 's?MYSERVERHERE?'$addr'?' | sed
's?MYBASEHERE?'$base'?' >/etc/ldap.conf
# config generated for ATLAS Tier3
nss_initgroups_ignoreusers
root,ldap,named,avahi,haldaemon,dbus,radvd,tomcat,radiusd,news,mailman,nscd,gdm
tls_cacertdir /etc/openldap/cacerts
uri ldap://MYSERVERHERE/
ssl start tls
tls cacertdir /etc/openldap/cacerts
pam_password md5
base MYBASEHERE
HereLdapConfEnds

echo Changing openldap ldap.conf file
cp /etc/openldap/ldap.conf /var/tier3g/openldap ldap.conf
#echo -n "Is this okay (y/n)? "
#read ans
#if [ $ans != "y" ]; then
#       echo enter another choice for the base (no spaces)
#         read base
#fi
cat - <<HereOLdapConfEnds |sed 's?MYSERVERHERE?'$addr'?'|sed
's?MYBASEHERE?'$base'?' >/etc/openldap/ldap.conf
# config for ATLAS tier 3
URI ldap://MYSERVERHERE/
BASE MYBASEHERE
TLS_CACERTDIR /etc/openldap/cacerts
TLS_REQCERT allow
HereOLdapConfEnds

echo updating /etc/nsswitch.conf
cp /etc/nsswitch.conf /var/tier3g/
cat - <<HereNsswitchConfEnds >/etc/nsswitch.conf

passwd:     files ldap
shadow:     files ldap
group:      files ldap
hosts:      files dns

ethers:     files
netmasks:   files
networks:   files
protocols:  files
rpc:        files
services:   files

netgroup:   files ldap

#publickey:  nisplus

automount:  files ldap
aliases:    files
#aliases:    files nisplus

HereNsswitchConfEnds
```

```
echo Setting /etc/pam.d/system-auth to use ldap authentication
cp /etc/pam.d/system-auth /var/tier3g/

cat - >/etc/pam.d/system-auth <<HereItEnds
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        sufficient    pam ldap.so use first pass
auth        required      pam deny.so

account     required      pam unix.so broken shadow
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     [default=bad success=ok user_unknown=ignore] pam_ldap.so
account     required      pam permit.so

password    requisite     pam cracklib.so try first pass retry=3
password    sufficient    pam unix.so md5 shadow nullok try first pass use authtok
password    sufficient    pam_ldap.so use_authtok
password    required      pam_deny.so

session     optional      pam keyinit.so revoke
session     required      pam_limits.so
session     optional      pam_mkhomedir.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet
use uid
session     required      pam unix.so
session     optional      pam ldap.so
HereItEnds

echo Ldap authentication enabled

echo "Congratulations! "
echo "This client is almost configured to use LDAP"

exit 0
```

Şekil A.6: "ldap_client_setup.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post install/scripts:$PATH
source $HOME/post_install/node-environment.config
#---------------------------------  NFS section ------------------------------
echo "---------------------------------  NFS section --------------------------"

echo "[`/bin/basename $0`] fetch configure-nfsv4.sh "
fetch shell script configure-nfsv4.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
    echo "Failed to fetch configure-nfsv4.sh"
    exit 1
fi

echo "[`/bin/basename $0`] configure-nfsv4.sh "
$HOME/post_install/scripts/configure-nfsv4.sh -S $NFS_SERVER_SHORT_NAME
ret code=RC$?
if [ $ret code == "RC0" ] ; then
       echo "Finished configuring nfs client"
else
    exit 1
fi
exit 0
```

Şekil A.7: "nfsv4-client.sh" Dosyası

```
#!/bin/bash
#  This script is used by the system administrator to setup system after kick
start installation
# create area for log files if needed

if [ ! -e $HOME/post install/logs/ ] ; then mkdir -p $HOME/post install/logs ; fi;

#  get date stamp for the log files
date stamp=`date +%d%m%Y %H%M%S`
```

```
# Define all status flags for individual steps

config_etc_hosts="PENDING"
config_firewall="PENDING"
config local users="PENDING"
config ganglia="PENDING"
config squid="PENDING"
config nsswitch="PENDING"
config_mail_alias="PENDING"

export PATH=$HOME/bin:$HOME/post install/scripts:$PATH
source $HOME/post install/node-environment.config

#--------------------  record what was done -----------------
echo " " >> $HOME/post_install/logs/squidvm-configuration.log
echo "  Configuring head node -"$date_stamp >> $HOME/post_install/logs/squidvm-
configuration.log

#
#   fetch the scripts needed for configuration
#

#-------------- configure /etc/hosts file ------------
echo
echo "----------------------  create new etc hosts ---------------------"
echo

fetch shell script configure etc hosts.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config_etc_hosts="FAILED"
else
  $HOME/post_install/scripts/configure_etc_hosts.sh >&
$HOME/post install/logs/squidvm configure etc hosts-$date stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config_etc_hosts="PASSED"
  else
     config_etc_hosts="FAILED"
  fi
fi

echo " Configure /etc/hosts - "$config_etc_hosts >>
$HOME/post_install/logs/squidvm-configuration.log


#------------ ganglia ------------
echo
echo "---------------------- ganglia setup --------------------"
echo
fetch shell script gmond-client.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config_ganglia="FAILED"
else
  $HOME/post_install/scripts/gmond-client.sh >& $HOME/post_install/logs/squidvm-
ganglia-$date stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config_ganglia="PASSED"
  else
     config ganglia="FAILED"
  fi
fi

echo " Configure ganglia - "$config_ganglia >> $HOME/post_install/logs/squidvm-
configuration.log

#------------- squid -------------
echo
echo "---------------------- squid setup --------------------"
echo
fetch shell script squidvm-squid.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config squid="FAILED"
else
```

```
  $HOME/post install/scripts/squidvm-squid.sh >& $HOME/post install/logs/squidvm-
squid-$date stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config squid="PASSED"
  else
     config squid="FAILED"
  fi
fi

echo " Configure squid - "$config squid >> $HOME/post install/logs/squidvm-
configuration.log

#----------------- new version of nsswitch.conf ----------------
echo
echo "---------------------- nsswitch.conf setup --------------------"
echo
fetch shell script nsswitch-vm.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config_nsswitch="FAILED"
else
  $HOME/post install/scripts/nsswitch-vm.sh >& $HOME/post install/logs/squidvm-
nsswitch-$date stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config_nsswitch="PASSED"
  else
     config nsswitch="FAILED"
  fi
fi

echo " Configure /etc/nsswitch - "$config_nsswitch >>
$HOME/post install/logs/squidvm-configuration.log


#----------------- mail alias ----------
echo
echo "---------------------- mail alias setup --------------------"
echo

fetch shell script mail alias.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config mail alias="FAILED"
else
  $HOME/post install/scripts/mail alias.sh >& $HOME/post install/logs/squidvm-
mail_alias-$date_stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config mail alias="PASSED"
  else
     config mail alias="FAILED"
  fi
fi

echo " Configure mail alias - "$config mail alias >>
$HOME/post install/logs/squidvm-configuration.log

echo " " >> $HOME/post_install/logs/squidvm-configuration.log


echo "starting yum update "
#
 /usr/bin/yum -y update
```

Şekil A.8: “post-install-squidvm-configuration.sh” Dosyası

```
#!/bin/bash
#  This script is used by the system administrator to setup system after kick
start installation

export PATH=$HOME/bin:$HOME/post_install/scripts:$PATH

# define the environmental variables for installation
```

```
source $HOME/post install/node-environment.config

#---------------------------- Ganglia Section --------------------------------

# define the environmental variables for installation
source $HOME/post install/node-environment.config

echo "---------------------------- Ganglia Section --------------------------------
--"
#
# fetch and setup the ganglia client
#
echo " Installing epel repo skipped"

echo "yum install ganglia ganglia-gmond"
yum -y install ganglia ganglia-gmond
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "Error installing epel repo"
    exit 1
fi

# put in proper address for gmetad server
echo "$HOME/post install/scripts/modify client gmond sl6 conf.sh  $GANGLIA CLUSTER
$GMETAD_IP"
$HOME/post_install/scripts/modify_client_gmond_sl6_conf.sh  $GANGLIA_CLUSTER
$GMETAD_IP
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "Error modifying gmond configuration files"
    exit 1
fi

/bin/cp -v $HOME/post install/config files/gmond.conf.new /etc/gmond.conf
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "Error installing /etc/gmond.conf"
    exit 1
fi

/sbin/chkconfig gmond --level 345 on
/sbin/service gmond restart
ret_code=RC$?
if [ $ret_code != "RC0" ] ; then
    echo "Error restarting gmond"
    exit 1
fi

exit 0
```

Şekil A.9: "gmond-client.sh" Dosyası

```
#!/bin/bash

cluster name=$1
gmeta_server_ip=$2

/bin/sed "s/MYCLUSTERNAMEHERE/$cluster name/g; "
$HOME/post install/config files/gmond.conf.template sl5 | /bin/sed
"s/GMETASERVERIPHERE/$gmeta server ip/g;" >
$HOME/post_install/config_files/gmond.conf.new
```

Şekil A.10: "modify_client-gmond_sl6_conf_sh" Dosyası

```
#!/bin/bash
#  This script is used by the system administrator to setup system after kick
start installation

if [ ! -e $HOME/post_install/logs/ ] ; then mkdir -p $HOME/post_install/logs ; fi;

date stamp=`date +%d%m%Y %H%M%S`

config etc hosts="PENDING"
config_firewall="PENDING"
config local users="PENDING"
config ganglia="PENDING"
```

```
config ldap="PENDING"
config ldap backup="PENDING"
config_nsswitch="PENDING"
config_mail_alias="PENDING"


export PATH=$HOME/bin:$HOME/post install/scripts:$PATH

source $HOME/post_install/node-environment.config

echo " " >> $HOME/post install/logs/ldapvm-configuration.log
echo "  Configuring head node -"$date stamp >> $HOME/post install/logs/ldapvm-
configuration.log

#-------------- configure /etc/hosts file ------------
echo
echo "---------------------- /etc/hosts setup --------------------"
echo

fetch shell script configure etc hosts.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config etc hosts="FAILED"
else
  $HOME/post_install/scripts/configure_etc_hosts.sh >&
$HOME/post_install/logs/ldapvm_configure_etc_hosts-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config etc hosts="PASSED"
  else
     config_etc_hosts="FAILED"
  fi
fi

echo " Configure /etc/hosts - "$config etc hosts >>
$HOME/post install/logs/ldapvm-configuration.log


#------------ ganglia ------------
echo
echo "---------------------- ganglia setup --------------------"
echo
fetch_shell_script gmond-client.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config ganglia="FAILED"
else
  $HOME/post_install/scripts/gmond-client.sh >& $HOME/post_install/logs/ldapvm-
ganglia-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config ganglia="PASSED"
  else
     config_ganglia="FAILED"
  fi
fi

echo " Configure ganglia - "$config ganglia >> $HOME/post install/logs/ldapvm-
configuration.log

#------------- ldap -------------
echo
echo "---------------------- ldap server setup --------------------"
echo
fetch shell script ldapvm-ldap.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config ldap="FAILED"
else
  $HOME/post install/scripts/ldapvm-ldap.sh 2>&1 | tee
$HOME/post install/logs/ldapvm-ldap-$date stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config ldap="PASSED"
  else
     config ldap="FAILED"
  fi
```

```
fi

echo " Configure ldap - "$config_ldap >> $HOME/post_install/logs/ldapvm-
configuration.log

#------------- ldap database backup script -------------
echo
echo "---------------------- ldap backup script setup ---------------------"
echo
fetch_shell_script ldapvm-ldap-backup.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config ldap backup="FAILED"
else
  $HOME/post_install/scripts/ldapvm-ldap-backup.sh 2>&1 | tee
$HOME/post_install/logs/ldapvm-ldap-backup-$date_stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config ldap backup="PASSED"
  else
     config_ldap_backup="FAILED"
  fi
fi

echo " Configure ldap backup script - "$config_ldap_backup >>
$HOME/post_install/logs/ldapvm-configuration.log

#----------------- new version of nsswitch.conf ----------------
echo
echo "---------------------- nsswitch setup ---------------------"
echo
fetch_shell_script nsswitch-vm.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config nsswitch="FAILED"
else
  $HOME/post_install/scripts/nsswitch-vm.sh >& $HOME/post_install/logs/ldapvm-
nsswitch-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config nsswitch="PASSED"
  else
     config_nsswitch="FAILED"
  fi
fi

echo " Configure /etc/nsswitch - "$config nsswitch >>
$HOME/post_install/logs/ldapvm-configuration.log


#----------------- mail alias ----------
echo
echo "---------------------- mail alias setup ---------------------"
echo
fetch_shell_script mail_alias.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config mail alias="FAILED"
else
  $HOME/post_install/scripts/mail_alias.sh >& $HOME/post_install/logs/ldapvm-
mail_alias-$date_stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config mail alias="PASSED"
  else
     config_mail_alias="FAILED"
  fi
fi

echo " Configure mail alias - "$config mail alias >>
$HOME/post install/logs/ldapvm-configuration.log


echo " " >> $HOME/post install/logs/ldapvm-configuration.log


echo "starting yum update "
```

```
 /usr/bin/yum -y update
```

Şekil A.11: “post-install-ldapvm-configuration.sh” Dosyası

```
#!/bin/bash
#  This script is used by the system administrator to setup system after kick
start installation

export PATH=$HOME/bin:$HOME/post install/scripts:$PATH

source $HOME/post_install/node-environment.config

#
#----------------------------- Ldap server configuration -------------------------
#
#   Scripts for setting up ldap
#
mkdir -v $HOME/bin

echo "Fetch scripts for ldap server configuration"
fetch_shell_script ldap_adduser.sh
ret_code=$?
if [ $ret code != "0" ] ; then
    echo "Failed to fetch the ldap adduser.sh script"
    exit 1
fi

fetch_shell_script ldap_server_setup.sh
ret code=$?
if [ $ret code != "0" ] ; then
    echo "Failed to fetch the ldap server setup.sh script"
    exit 1
fi

echo "  !!!!!!!!!!!!  start process of setting up ldapd server !!!!!!!!!!!!!!!!!!
"
$HOME/post install/scripts/ldap server setup.sh $LDAP SERVER NAME
ret_code=RC$?
if [ $ret_code != "RC0" ] ; then
    echo "  Failed setting up the ldap server"
    exit 1
fi

exit 0
```

Şekil A.12: “ldapvm-ldap.sh” Dosyası

```
#!/bin/bash
# you should customize this script to your own location
# by hardcoding the BASE (e.g. BASE=" dc=ct3,dc=aydin,dc=edu,dc=tr ")
BASE="dc=ct3,dc=aydin,dc=edu,dc=tr"
echo the base is set to $BASE

# create a temporary file
tempfoo=`basename $0`
TMPFILE=`mktemp -q /tmp/${tempfoo}.XXXXXX`
if [ $? -ne 0 ]; then
    echo "$0: Can't create temp file, exiting..."
    exit 1
fi


echo
echo "  you will running the ldapsearch command to determine the largest User id
used (UID) in the LDAP DB"
echo

ldapsearch -x -W -D "cn=root,$BASE" objectClass=account uidNumber | grep uidNumber
| awk '{print $2}' | sort -n | tail -1 > $TMPFILE

largest userid=`cat $TMPFILE`

if (( largest userid == 1055 || largest userid == 1050 ))
then
  (( largest userid = 1056 ))
fi
```

```
echo "Please choose a UID > $largest userid"

echo -n "please enter the username: "
read NAME
echo -n "please enter the UID: "
read NEW UID
#echo -n "please enter the GID: "
#read NEW_GID
echo "All users put into the users group with GID = 100"
NEW GID=100
GROUP=users

echo -n "please enter the inital password (it will be stored encrypted: "
read XXX
HASH_FROM_SLAPPASSWD=`slappasswd -s $XXX`
echo The following will be added:
echo user $NAME, group $NAME, UID $NEW UID, GID $NEW GID, initial password $XXX
echo -n "Is this correct (y/n)? "
read ans
if [ "$ans" != "y" ] ; then
    echo Come back soon\!
    exit 0
fi
TMPF=`mktemp /tmp/ldif_${NAME}.XXXXXXXXX || exit 1`
cat - <<EOF  >$TMPF
dn: uid=$NAME,ou=People,$BASE
uid: $NAME
cn: $NAME
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: $HASH FROM SLAPPASSWD
shadowLastChange: 14568
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: $NEW_UID
gidNumber: $NEW GID
homeDirectory: /export/home/$NAME
gecos: $NAME

EOF
# add the user
echo " Calling command to add user - ldapadd -x -W -D "cn=root,$BASE" -f $TMPF"
ldapadd -x -W -D "cn=root,$BASE" -f $TMPF
rm $TMPF
#add the user to the users netgroup
#cat - <<EOF >$TMPF
#dn: cn=users,ou=Netgroup,$BASE
#changetype: modify
#add: memberNisNetgroup
#memberNisNetgroup: $NAME
#EOF
#echo "Calling command to and netgroup info - ldapadd -x -W -D "cn=root,$BASE" -f
$TMPF"
#ldapadd -x -W -D "cn=root,$BASE" -f $TMPF
#rm $TMPDBF
exit 0
```

Şekil A.13: "ldap_adduser.sh" Dosyası

```
#!/bin/bash
# server setup script for ldap server REB 2/18/10

IFCONFIG=/sbin/ifconfig
AWK=/bin/awk
IPCALC=/bin/ipcalc
GREP=/bin/grep

network netmask="0.0.0.0"
network_ip="0.0.0.0"

function get network info {
```

```
NETIF=$1

network_netmask="none"
network_ip="none"

#  check if this network interface as a ip address

$IFCONFIG $NETIF | $GREP -q "inet addr"
ret_code=RC$?

if [ $ret code == "RC0" ]
then
    IP="`$IFCONFIG $NETIF | $AWK  /$NETIF/'{next}//{split($0,a,":");split(a[2],a,"
");print a[1];exit}'`"

    NETMASK="`$IFCONFIG $NETIF | $AWK
/$NETIF/'{next}//{split($0,a,":");split(a[4],a," ");print a[1];exit}'`"

    PREFIX=`$IPCALC -p $IP $NETMASK | $AWK --field-separator "=" '{print
$2;exit}'`

    network_ip=$IP
    network netmask=$NETMASK
fi

export network_ip
export network_netmask
}

echo " The following packages need to be installed:  openssl-perl openldap
openldap-servers "
echo

echo -n "Enter your ldap server's fully qualified address (lower case):
<nodename.myuniv.edu> "
read addr
HOSTWAIT=10  # Specify up to 10 seconds for host query reply.

ipaddr=$(host -W $HOSTWAIT $addr | head -n 1 | awk '{print $4}')  # host lookup to
get IP address.
ipname=$(host -W $HOSTWAIT $ipaddr | head -n 1 | awk '{print $5}')  # host lookup
to get IP address.

full_hostname=""
if [ $ipname == $addr. ]; then
   full hostname=$addr
else
   echo " Error - $addr is not a full qualified server name  $addr .ne. $ipname "
   exit 1
fi

echo -n "The fully qualified server name is $addr with ip $ipaddr (y/n)? "
read ans
if [ $ans != "y" ]; then
       echo Come back soon\!
       exit 1
fi

#   Now make a self signed certificates

echo
echo " ++++   Now Make a self signed certificates ++++ "
echo
echo "  Making a self signed certificate for use with the ldap server "; echo

dirfile=/etc/openldap/cacerts
if [[ ! -e $dirfile ]]; then  mkdir -p /etc/openldap/cacerts ;  fi

echo " Note - you will be asked a series of questions - it is critical that the
Common Name (CN) "
echo " is the fully qualified hostname of the server the certificates created here
are good for 10 years "
echo

openssl req -new -x509 -nodes -out /etc/openldap/cacerts/${full hostname}.pem -
keyout /etc/openldap/cacerts/${full hostname}.pem -days 3650
chgrp ldap /etc/openldap/cacerts/${full_hostname}.pem
```

```
chmod 640 /etc/openldap/cacerts/${full hostname}.pem
#(cd /etc/openldap/cacerts;c rehash)

#
#  Extract the information needed for the client machines  #
grep -A 50 CERT /etc/openldap/cacerts/${full hostname}.pem >
/etc/openldap/cacerts/${full hostname}-cl.pem

echo
echo Backup copies of the old files will be saved in /var/asc

mkdir /var/asc
cp /etc/openldap/ldap.conf /var/asc/ldap openldap.conf

echo customizing /etc/openldap/ldap.conf
echo

host=`echo $addr |sed 's?\..*$??'`
base=`echo $addr |sed 's?^'$host'??'|sed 's?\.?,dc=?g'|sed 's?^,??'`
echo ldap base is being set to $base
echo -n Is this okay \(y/n\)\?
read ans
if [ $ans != "y" ]; then
        echo enter another choice for the base \(no spaces\)
         read base
fi

domaincomponent=`echo $base | awk 'BEGIN{FS=","}{print $1}' | sed 's?dc=??g' | awk
'{print $1}'`

cat - <<HereLdapConfEnds |sed 's?MYSERVERHERE?'$addr'?'|sed
's?MYBASEHERE?'$base'?'|sed 's?MYSERVERIPHERE?'$ipaddr'?' >/etc/openldap/ldap.conf

# LDAP Defaults

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

URI ldap://MYSERVERHERE/
BASE MYBASEHERE
TLS CACERTDIR /etc/openldap/cacerts
TLS REQCERT allow
HereLdapConfEnds
echo
echo " You will be asked for root password for ldap database"
echo
HASH FROM SLAPPASSWD=`slappasswd`

echo "Modifying ldap_adduser.sh script to add proper ldap base"
cp $HOME/post_install/scripts/ldap_adduser.sh /tmp/ldap_adduser.sh.sed

cat /tmp/ldap adduser.sh.sed | sed 's?MYBASEHERE?'$base'?'>
$HOME/post install/scripts/ldap adduser.sh
cp -v  $HOME/post_install/scripts/ldap_adduser.sh $HOME/bin/ldap_adduser.sh

echo "Modifying ldap_removeuser.sh script to add proper ldap base"
cp $HOME/post install/scripts/ldap removeuser.sh /tmp/ldap removeuser.sh.sed

cat /tmp/ldap removeuser.sh.sed | sed 's?MYBASEHERE?'$base'?'>
$HOME/post_install/scripts/ldap_removeuser.sh
cp -v  $HOME/post_install/scripts/ldap_removeuser.sh $HOME/bin/ldap_removeuser.sh


echo Modifying /etc/openldap/slapd.conf
cp /etc/openldap/slapd.conf /var/asc
cat - <<HereSlapdConfEnds |sed 's?MYSERVERHERE?'$addr'?'|sed
's?MYBASEHERE?'$base'?' |sed 's?FULLHOST?'$full hostname'?' |  sed 's?^# SCRIPT
LOCATOR FOR ROOTPW.*$?rootpw '$HASH FROM SLAPPASSWD'?' >/etc/openldap/slapd.conf
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema
```

```
# Allow LDAPv2 client connections.  This is NOT the default.
allow bind v2

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral       ldap:/$HOME.openldap.org

pidfile         /var/run/openldap/slapd.pid
argsfile        /var/run/openldap/slapd.args

TLSCACertificateFile /etc/openldap/cacerts/FULLHOST-cl.pem
TLSCertificateFile /etc/openldap/cacerts/FULLHOST.pem
TLSCertificateKeyFile /etc/openldap/cacerts/FULLHOST.pem

# the below is addded to protect the password hash from prying eyes
access to attrs=userPassword
       by self write
       by anonymous auth
       by * none

access to * by * read
########################################################################
# ldbm and/or bdb database definitions
########################################################################

database        bdb
suffix          "MYBASEHERE"
rootdn          "cn=root,MYBASEHERE"
directory       /var/lib/ldap

# Indices to maintain for this database
index objectClass                       eq,pres
index ou,cn,mail,surname,givenname      eq,pres,sub
index uidNumber,gidNumber,loginShell    eq,pres
index uid,memberUid                     eq,pres,sub
index nisMapName,nisMapEntry            eq,pres,sub

# Replicas of this database
#replogfile /var/lib/ldap/openldap-master-replog
#replica host=ldap-1.example.com:389 starttls=critical
#     bindmethod=sasl saslmech=GSSAPI
#     authcId=host/ldap-master.example.com@EXAMPLE.COM
HereSlapdConfEnds

echo
echo Modifying the database
TMPFILE=`mktemp /tmp/database0.XXXXXXXXXX` || exit 1
cat - <<HereLdifEnds |sed 's?MYSERVERHERE?'$addr'?'|sed 's?MYBASEHERE?'$base'?'  |
sed 's?MYDOMAINCOMPONENT?'$domaincomponent'?' >$TMPFILE
dn: MYBASEHERE
dc: MYDOMAINCOMPONENT
objectClass: top
objectClass: domain

dn: ou=Hosts,MYBASEHERE
ou: Hosts
objectClass: top
objectClass: organizationalUnit

dn: ou=Rpc,MYBASEHERE
ou: Rpc
objectClass: top
objectClass: organizationalUnit

dn: ou=Services,MYBASEHERE
ou: Services
objectClass: top
objectClass: organizationalUnit

dn: nisMapName=netgroup.byuser,MYBASEHERE
nismapname: netgroup.byuser
objectClass: top
objectClass: nisMap

dn: ou=Mounts,MYBASEHERE
ou: Mounts
objectClass: top
```

```
objectClass: organizationalUnit

dn: ou=Networks,MYBASEHERE
ou: Networks
objectClass: top
objectClass: organizationalUnit

dn: ou=People,MYBASEHERE
ou: People
objectClass: top
objectClass: organizationalUnit

dn: ou=Group,MYBASEHERE
ou: Group
objectClass: top
objectClass: organizationalUnit

dn: ou=Netgroup,MYBASEHERE
ou: Netgroup
objectClass: top
objectClass: organizationalUnit

dn: ou=Protocols,MYBASEHERE
ou: Protocols
objectClass: top
objectClass: organizationalUnit

dn: ou=Aliases,MYBASEHERE
ou: Aliases
objectClass: top
objectClass: organizationalUnit

dn: nisMapName=netgroup.byhost,MYBASEHERE
nismapname: netgroup.byhost
objectClass: top
objectClass: nisMap

dn: uid=root,ou=People,MYBASEHERE
uid: root
cn: root
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}!!
shadowLastChange: 14568
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 0
gidNumber: 0
homeDirectory: $HOME

dn: uid=xrootd,ou=People,MYBASEHERE
uid: xrootd
cn: xrootd
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}!!
shadowLastChange: 14568
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1050
gidNumber: 1050
homeDirectory: /local/home/xrootd

dn: uid=condor,ou=People,MYBASEHERE
uid: condor
cn: condor
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}!!
```

```
shadowLastChange: 14568
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1055
gidNumber: 1055
homeDirectory: /local/home/condor

HereLdifEnds

cp /etc/openldap/DB CONFIG.example /var/lib/ldap/DB CONFIG
slapadd -l $TMPFILE
chown -R ldap:ldap /var/lib/ldap/*
/bin/rm $TMPFILE


echo
echo starting and configuring the ldap server
chkconfig --add ldap
chkconfig ldap on
service ldap start
echo you are almost done

# external network interface
EXTIF=$PUBLIC_INTERFACE

get_network_info $EXTIF

XXX=$network ip"/"$network netmask

echo
echo "now add the following two lines to your /etc/system/iptables file just"
echo "before the line that reads"
echo "-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited COMMIT"
echo "here are the four lines:"
echo "-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp -s "$XXX" --dport
389 -j ACCEPT"
echo "-A RH-Firewall-1-INPUT -m state --state NEW -m udp -p udp -s "$XXX" --dport
389 -j ACCEPT"
echo "-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp -s "$XXX" --dport
636 -j ACCEPT"
echo "-A RH-Firewall-1-INPUT -m state --state NEW -m udp -p udp -s "$XXX" --dport
636 -j ACCEPT"
echo
echo "Once you have change the file"
echo
echo "issue this command: /sbin/service iptables restart"
echo "issue this command: /sbin/service iptables save"
```

Şekil A.14: "ldap_server_setup.sh" Dosyası

```
#!/bin/bash
#  This script is used by the system administrator to setup system after kick
start installation

if [ ! -e $HOME/post_install/logs/ ] ; then mkdir -p $HOME/post_install/logs ; fi;

date stamp=`date +%d%m%Y %H%M%S`

# Define all status flags for individual steps

config_firewall="PENDING"
config local users="PENDING"
config nfsv4="PENDING"
config ganglia="PENDING"
config squid="PENDING"
config_ldap_server="PENDING"
config_ldap_backup="PENDING"
config ldap client="PENDING"
config condor="PENDING"
config squid="PENDING"
config file limits="PENDING"
config_restrict_users="PENDING"
config_mail_alias="PENDING"
config xrootd file mounts="PENDING"
```

```
export PATH=$HOME/bin:$HOME/post install/scripts:$PATH

source $HOME/post_install/node-environment.config

#------------------  record what was done -----------------

echo " " >> $HOME/post install/logs/headnode-configuration.log
echo "$date stamp >> $HOME/post install/logs/headnode-configuration.log

echo " Configure firewall - "$config_firewall >> $HOME/post_install/logs/headnode-
configuration.log

#----------- local users ----------
echo
echo "---------------------- setup local users --------------------"
echo
fetch shell script local users.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config local users="FAILED"
else
  $HOME/post_install/scripts/local_users.sh >& $HOME/post_install/logs/headnode-
local users-$date stamp.log
  ret code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config_local_users="PASSED"
  else
     config local users="FAILED"
  fi
fi

echo " Configure local users - "$config_local_users >>
$HOME/post_install/logs/headnode-configuration.log

#----------- configure for xrootd file limits  ----------
echo
echo "----------------------  xrootd file limits --------------------"
echo
fetch_shell_script increase_file_limits.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config file limits="FAILED"
else
  $HOME/post_install/scripts/increase_file_limits.sh >&
$HOME/post install/logs/headnode-local users-$date stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config_file_limits="PASSED"
  else
     config_file_limits="FAILED"
  fi
fi

echo " Configure xrootd file limits - "$config_file_limits >>
$HOME/post_install/logs/headnode-configuration.log

#----------- Restrict users ----------
echo
echo "----------------------  fetch scripts to restrict login access --------------
------"
echo
fetch shell script restrict login access.sh
ret code=$?
if [ $ret code != "0" ] ; then
    echo "Failed to fetch the restrict login access.sh"
    echo " Fetch restrict_login_access.sh  - FAILED" >>
$HOME/post install/logs/headnode-configuration.log
   config local users="FAILED"
else
  echo " You will need to execute the script later by hand :"
  echo
  echo "$HOME/post_install/scripts/restrict_login_access.sh"
  echo
  echo " Fetch restrict login access.sh  - PASSED" >>
$HOME/post install/logs/headnode-configuration.log
fi
```

```
echo " Configure restrict login access  - "$config restrict users >>
$HOME/post install/logs/headnode-configuration.log

#------------ nfs v4 ------------
echo
echo "---------------------  nfs client setup --------------------"
echo

fetch_shell_script nfsv4-client.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config nfsv4="FAILED"
else
  $HOME/post install/scripts/nfsv4-client.sh >& $HOME/post install/logs/headnode-
nfsv4-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config nfsv4="PASSED"
  else
     config nfsv4="FAILED"
  fi
fi

echo " Configure nfsv4 - "$config nfsv4 >> $HOME/post install/logs/headnode-
configuration.log

#------------ ganglia ------------
echo
echo "--------------------- ganglia setup --------------------"
echo

fetch_shell_script gmond-client.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config ganglia="FAILED"
else
  $HOME/post_install/scripts/gmond-client.sh >& $HOME/post_install/logs/headnode-
ganglia-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config ganglia="PASSED"
  else
     config_ganglia="FAILED"
  fi
fi

echo " Configure ganglia - "$config ganglia >> $HOME/post install/logs/headnode-
configuration.log


#-------------- condor ----------
echo
echo "---------------------  condor setup --------------------"
echo

fetch_shell_script headnode-condor.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config condor="FAILED"
else
  $HOME/post_install/scripts/headnode-condor.sh >&
$HOME/post install/logs/headnode-condor-$date stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config condor="PASSED"
  else
     config condor="FAILED"
  fi
fi

echo " Configure Condor - "$config condor >> $HOME/post install/logs/headnode-
configuration.log

# --------------------------------  Ldap client ----------------------------------
-----------
# Fetch the script for setting up the ldap client
echo
```

```
echo "----------------------  ldap client setup --------------------"
echo
fetch_shell_script ldap_client_setup.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config ldap client="FAILED"
else
    $HOME/post install/scripts/ldap client setup.sh $LDAP SERVER NAME  >&
HOME/post_install/logs/headnode-ldap-client-$date_stamp.log
    ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config ldap client="PASSED"
  else
     config ldap client="FAILED"
  fi
fi

#echo " Configure ldap client - "$config ldap client >>
$HOME/post install/logs/headnode-configuration.log
#----------------- mail alias ----------
echo
echo "----------------------  mail alias --------------------"
echo
fetch shell script mail alias.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config_mail_alias="FAILED"
else
  $HOME/post install/scripts/mail alias.sh >& $HOME/post install/logs/headnode-
mail alias-$date stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config_mail_alias="PASSED"
  else
     config mail alias="FAILED"
  fi
fi

echo " Configure mail alias - "$config_mail_alias >>
$HOME/post install/logs/headnode-configuration.log

#------------------------  xrootd file mounts ------------------------

echo
echo "-----------------------   xrootd file mounts -------------- "
echo

fetch_shell_script make_xrootd_file_mounts_headnode.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
    config xrootd file mounts="FAILED"
else
    echo " execute $HOME/post install/scripts/make xrootd file mounts headnode.sh
after xrootd account is created"  >> $HOME/post_install/logs/headnode-
configuration.log
fi
echo " Configure xrootd file mounts - "$config xrootd file mounts >>
$HOME/post install/logs/headnode-configuration.log

echo " " >> $HOME/post_install/logs/headnode-configuration.log

echo "starting yum update "
 /usr/bin/yum -y update
```

Şekil A.15: "post-install-headnode-configuration-public-network.sh" Dosyası

```
#!/bin/bash

EXPECTED ARGS=1
default xrootd user=xrootd
username=""

if [ $# -ne $EXPECTED_ARGS ] ; then
    echo " Using default xrootd user name - "$default_xrootd_user
    username=$default xrootd user
else
```

```
    username=$1
fi

echo "modify /etc/security/limits.conf for $username account for xrootd"

echo "# added automatically" >> /etc/security/limits.conf
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "failed to modify /etc/security/limits.conf"
    exit 1
fi

echo $username"      soft    nofile          16384" >> /etc/security/limits.conf
ret code=RC$?
if [ $ret_code != "RC0" ] ; then
    echo "failed to modify /etc/security/limits.conf"
    exit 1
fi

echo $username"      hard    nofile          63536" >> /etc/security/limits.conf
ret_code=RC$?
if [ $ret_code != "RC0" ] ; then
    echo "failed to modify /etc/security/limits.conf"
    exit 1
fi
```

Şekil A.16: "increase_file_limits.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post install/scripts:$PATH
source $HOME/post install/node-environment.config

echo "[$0] /usr/bin/yum -y install condor"
/usr/bin/yum -y install condor
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "Could not install condor with yum"
    exit 1
fi

echo "[$0] fetch shell script create-condor-credential-file.sh"
fetch shell script create-condor-credential-file.sh
ret code=$?
if [ $ret_code != "0" ] ; then
    echo "could not fetch script to create condor-credential-file.sh"
    exit 1
fi

echo "[$0] $HOME/post install/scripts/create-condor-credential-file.sh"
$HOME/post_install/scripts/create-condor-credential-file.sh
ret_code=RC$?
if [ $ret code != "RC0" ] ; then
    echo " Could not create condor credential file "
    exit 1
fi

echo "[$0] modify /etc/condor/condor_config.local"
#   add information at the end of the condor config.local file to replace
information defined earlier
/bin/echo "##  Next configuration to be read is for the T3 cluster setup" >>
/etc/condor/condor_config.local
ret_code=RC$?
if [ $ret code != "RC0" ] ; then
    echo " Could not modify /etc/condor/condor config.local"
    exit 1
fi

/bin/echo "LOCAL_CONFIG_FILE = /export/share/condor-etc/condor_config.cluster
/export/share/condor-etc/condor config.head.local" >>
/etc/condor/condor config.local
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo " Could not modify /etc/condor/condor_config.local"
    exit 1
fi
```

```
#  check that the cluster wide config files exist
echo "[$0] check if can read the condor configuration files "
if [! -e /export/share/condor-etc/condor_config.cluster ] ; then
   if [ ! -e /export/share/condor-etc/condor_config.head.local ] ; then
       echo "Could not find remote config files"
       exit 1
   fi
fi

/sbin/chkconfig --level 235 condor on
exit 0
```

Şekil A.17: "headnode-condor.sh" Dosyası

```
#!/bin/bash
#  Create the condor credential same for all nodes in condor cluster

/usr/sbin/condor store cred -f /var/lib/condor/condor_credential -p atlast3
```

Şekil A.18: "create-condor-credential-file.sh" Dosyası

```
#!/bin/bash

xrootd_account=xrootd

/usr/bin/getent passwd | /bin/grep $xrootd account >& /dev/null
ret code=RC$?
if [ ! $ret_code == "RC0" ] ; then  # no xrootd account
    echo " Error !!!! $xrootd_account account does not exist - Exiting early" >&2
    exit 1
fi

if [ ! -e /atlas/inventory ] ; then /bin/mkdir -pv /atlas/inventory
/bin/chown -R $xrootd_account /atlas

exit 0
```

Şekil A.19: "make-xrootd_file_mounts_headnode.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post_install/scripts:$PATH
source $HOME/post_install/node-environment.config

if [ ! -e $HOME/post install/logs/ ] ; then mkdir -p $HOME/post install/logs ; fi;
date stamp=`date +%d%m%Y %H%M%S`

config_firewall="PENDING"
config_local_users="PENDING"
config nfsv4="PENDING"
config ganglia="PENDING"
config ldap="PENDING"
config condor="PENDING"
config_httpd="PENDING"
config_mail_alias="PENDING"
config user dirs="PENDING"
config restrict users="PENDING"
config ddm scripts="PENDING"
config dir ownership="PENDING"

#-------------------  record what was done -----------
echo " " >> $HOME/post install/logs/nfsnode-configuration.log
echo " Configuring nfs node -"$date stamp >> $HOME/post install/logs/nfsnode-
configuration.log


#---------- firewall --------
echo
echo "---------------------  firewall configuration --------------------"
echo
fetch_shell_script nfsnode-firewall-public.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config firewall="FAILED"
else
  $HOME/post_install/scripts/nfsnode-firewall-public.sh >&
$HOME/post_install/logs/nfsnode-firewall-$date_stamp.log
```

```
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config_firewall="PASSED"
  else
     config firewall="FAILED"
  fi
fi

echo " Configure firewall - "$config_firewall >> $HOME/post_install/logs/nfsnode-
configuration.log
#----------- local users ----------
echo
echo "--------------------- setup local users --------------------"
echo
fetch_shell_script local_users.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config local users="FAILED"
else
  $HOME/post install/scripts/local users.sh >& $HOME/post install/logs/nfsnode-
local_users-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config local users="PASSED"
  else
     config_local_users="FAILED"
  fi
fi

echo " Configure local users - "$config local users >>
$HOME/post_install/logs/nfsnode-configuration.log

#----------- script to create user directories ---------
echo
echo "-------------------- create user directories ------- "
echo

fetch_shell_script create_user_dir.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config user dirs="FAILED"
else
  mkdir $HOME/bin
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
      echo " Create $HOME/bin directory " > $HOME/post install/logs/nfsnode-user-
dirs-$date stamp.log
      chmod +x $HOME/post_install/scripts/create_user_dir.sh
      cp -v $HOME/post_install/scripts/create_user_dir.sh $HOME/bin/
      ret_code=RC$?
      if [ $ret code == "RC0" ] ; then
         config user dirs="PASSED"
         echo "Copied create user dir.sh " >> $HOME/post install/logs/nfsnode-
user-dirs-$date_stamp.log
      else
         echo " Failed to copy create_user_dir.sh " >>
$HOME/post install/logs/nfsnode-user-dirs-$date stamp.log
         config user dirs="FAILED"
      fi
   else
     echo " Failed to fetch create_user_dir.sh " >
$HOME/post install/logs/nfsnode-user-dirs-$date stamp.log
     config user dirs="FAILED"
  fi
fi

echo " Create user directories  - "$config user dirs >>
$HOME/post install/logs/nfsnode-configuration.log

#---------------------------------  NFS section ---------------------------------
--
#
#  get the routines for configuring nfs and configuring iptables
#
echo
echo "---------------------  nfs server setup --------------------"
echo
```

```
fetch shell script nfsv4-server-public.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config nfsv4="FAILED"
else
    $HOME/post install/scripts/nfsv4-server-public.sh >&
$HOME/post install/logs/nfsnode-nfsv4-$date stamp.log
    ret_code=RC$?
    if [ $ret_code == "RC0" ] ; then
       config nfsv4="PASSED"
    else
       config nfsv4="FAILED"
    fi
fi

echo " Configure nfsv4 - "$config nfsv4 >> $HOME/post install/logs/nfsnode-
configuration.log

#------------ ganglia ------------
echo
echo "---------------------- ganglia setup --------------------"
echo
fetch shell script ganglia-server.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config_ganglia="FAILED"
else
  $HOME/post install/scripts/ganglia-server.sh >& $HOME/post install/logs/nfsnode-
ganglia-$date stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config_ganglia="PASSED"
  else
     config ganglia="FAILED"
  fi
fi

echo " Configure ganglia - "$config_ganglia >> $HOME/post_install/logs/nfsnode-
configuration.log

#----------- Restrict users ----------
echo
echo "----------------------  fetch scripts to restrict login access --------------
------"
echo
fetch shell script restrict login access.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
    echo "Failed to fetch the restrict_login_access.sh"
    echo " Fetch restrict login access.sh  - FAILED" >>
$HOME/post install/logs/nfsnode-configuration.log
   config local users="FAILED"
else
  echo " You will need to execute the script later by hand :"
  echo
  echo "$HOME/post install/scripts/restrict login access.sh"
  echo
  echo " Fetch restrict login access.sh  - PASSED" >>
$HOME/post_install/logs/nfsnode-configuration.log
fi

echo " Configure restrict login access  - "$config restrict users >>
$HOME/post install/logs/nfsnode-configuration.log


#-------------- condor ----------
echo
echo "----------------------  condor setup --------------------"
echo
fetch shell script nfsnode-condor-public.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config condor="FAILED"
else
  $HOME/post install/scripts/nfsnode-condor-public.sh >&
$HOME/post_install/logs/nfsnode-condor-$date_stamp.log
```

```
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config_condor="PASSED"
  else
     config condor="FAILED"
  fi
fi

echo " Configure Condor - "$config_condor >> $HOME/post_install/logs/nfsnode-
configuration.log

echo
echo "---------------------  fetch the DDM Scripts --------------------"
echo
fetch_shell_script ddm_scripts_config.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config ddm scripts="FAILED"
else
    $HOME/post install/scripts/ddm scripts config.sh >&
$HOME/post_install/logs/nfsnode-ddm-scripts-$date_stamp.log
    ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config ddm scripts="PASSED"
  else
     config_ddm_scripts="FAILED"
  fi
fi

fetch shell script change export share ownership.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
    config_dir_ownership="FAILED"
else
    $HOME/post install/scripts/change export share ownership.sh >&
$HOME/post install/logs/nfsnode-export share ownership-$date stamp.log
    ret_code=RC$?
    if [ $ret_code == "RC0" ] ; then
       config_dir_ownership="PASSED"
    else
       config dir ownership="FAILED"
    fi
fi

echo " set ownership of /export/share/atlas - "$config dir ownership >>
$HOME/post install/logs/nfsnode-configuration.log

echo "---------------------  mail alias --------------------"
fetch_shell_script mail_alias.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config mail alias="FAILED"
else
  $HOME/post_install/scripts/mail_alias.sh >& $HOME/post_install/logs/nfsnode-
mail_alias-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config mail alias="PASSED"
  else
     config_mail_alias="FAILED"
  fi
fi

echo " Configure mail alias - "$config mail alias >>
$HOME/post install/logs/nfsnode-configuration.log

echo " " >> $HOME/post install/logs/nfsnode-configuration.log

 /usr/bin/yum -y update
```

Şekil A.20: "post-install-nfsnode-configuration-public.sh" Dosyası

```
#!/usr/bin/env bash

IFCONFIG=/sbin/ifconfig
AWK=/bin/awk
```

```
IPCALC=/bin/ipcalc
GREP=/bin/grep

source $HOME/post_install/node-environment.config


# external network interface
EXTIF=$PUBLIC INTERFACE

# internal network interface
INTIF=$PRIVATE INTERFACE

function usage {
    echo "[configure-nfsv4]:  Error !!! "
    echo "[configure-nfsv4]: call format: configure-nfsv4-public.sh [-V
debug_mode] -N network_domain [-S server_short_name]"
    echo "[configure-nfsv4]:  Exiting early "
    exit 1
}

network_netmask="0.0.0.0"
network_ip="0.0.0.0"

function get network info {

NETIF=$1
network_netmask="none"
network ip="none"

#  check if this network interface as a ip address

$IFCONFIG $NETIF | $GREP -q "inet addr"
ret_code=RC$?
if [ $ret code == "RC0" ]
then
    IP="`$IFCONFIG $NETIF | $AWK  /$NETIF/'{next}//{split($0,a,":");split(a[2],a,"
");print a[1];exit}'`"

    NETMASK="`$IFCONFIG $NETIF | $AWK
/$NETIF/'{next}//{split($0,a,":");split(a[4],a," ");print a[1];exit}'`"

    PREFIX=`$IPCALC -p $IP $NETMASK | $AWK --field-separator "=" '{print
$2;exit}'`

    network ip=$IP
    network netmask=$NETMASK
fi

export network_ip
export network_netmask
}

network domain=NONE
nis_domain=NONE
DEBUG_SCRIPT=none
server_name=none
private network domain=NONE

export OPTIND=1
while getopts :V:S:v:s: OPT; do
    case $OPT in
        S) # input server name
            export server name=${OPTARG}
            if [ $DEBUG SCRIPT != "none" ] ; then
              echo [configure-nfsv4]:server name=${OPTARG}
            fi
            ;;
        V) # debug script,
            export  DEBUG SCRIPT=$OPTARG
            ;;
        s) # input NFS server name
            export server_name=${OPTARG}
            if [ $DEBUG SCRIPT != "none" ] ; then
              echo [configure-nfsv4]:server name=${OPTARG}
            fi
            ;;
        v) # debug script,
```

```
            export  DEBUG SCRIPT=$OPTARG
            ;;
        *)
           echo [configure-nfsv4]: OTHER: $OPT $OPTARG
           usage
           ;;
     esac
done

if [ ! -e $HOME/post_install/backup ] ; then mkdir  $HOME/post_install/backup ; fi

backup date stamp=`date +%d%m%Y %H%M%S`
/bin/cp /etc/idmapd.conf $HOME/post install/backup/idmapd.conf-backup-
${backup date stamp}

#/bin/sed "s/localdomain/$NETWORK_DOMAIN_NAME/g;"
$HOME/post install/backup/idmapd.conf-backup-${backup date stamp} |  /bin/sed
"s/nobody/nfsnobody/g;" > $HOME/post install/config files/idmapd.conf
/bin/sed "s/nobody/nfsnobody/g;"  $HOME/post install/backup/idmapd.conf-backup-
${backup date stamp} > $HOME/post install/config files/idmapd.conf

/bin/cp -v  $HOME/post_install/config_files/idmapd.conf /etc/idmapd.conf

if [ ! -e /local/home ] ; then mkdir -pv /local/home; fi
if [ ! -e /export ] ; then mkdir -pv /export ; fi

if [ $server_name == "none" ] ;
then
    if [ ! -e /export/home ] ; then mkdir -pv /export/home; fi
    if [ ! -e /export/share ] ; then mkdir -pv /export/share; fi
    if [ ! -e /export/share/atlas ] ; then mkdir -pv /export/share/atlas; fi
    if [ ! -e /export/share/condor-etc ] ; then mkdir -pv /export/share/condor-
etc; fi
    if [ ! -e /export/share/pilot ] ; then mkdir -pv /export/share/pilot; fi
    if [ ! -e /export/share/pilot/scheduler ] ; then mkdir -pv
/export/share/pilot/scheduler; fi
    if [ ! -e /export/share/pandat3-output ] ; then mkdir -pv
/export/share/pandat3-output; fi

    mkdir /NFSV4exports
    mkdir /NFSV4exports/home
    mkdir /NFSV4exports/share
    mkdir /NFSV4exports/share/atlas
    mkdir /NFSV4exports/share/condor-etc
    mkdir /NFSV4exports/share/pilot
    mkdir /NFSV4exports/share/pandat3-output
    get network info $EXTIF
    if [ $network_netmask != "none" ]
    then
        network_domain=$network_ip"/"$network_netmask
        echo "Determined PN - " $network domain
        echo "/NFSV4exports "${network domain}"(fsid=0,insecure,no subtree check)"
>> exports.new
        echo "/NFSV4exports/home
"${network_domain}"(rw,nohide,insecure,no_subtree_check,root_squash)" >>
exports.new
        echo "/NFSV4exports/share/atlas
"${network domain}"(rw,nohide,insecure,no subtree check,root squash)" >>
exports.new
        echo "/NFSV4exports/share/condor-etc
"${network_domain}"(rw,nohide,insecure,no_subtree_check,root_squash)" >>
exports.new
        echo "/NFSV4exports/share/pilot
"${network domain}"(rw,nohide,insecure,no subtree check,root squash)" >>
exports.new
        echo "/NFSV4exports/share/pandat3-output
"${network domain}"(rw,nohide,insecure,no subtree check,root squash)" >>
exports.new
    fi

    if [ -e /etc/exports ] ; then /bin/cp -v /etc/exports /etc/exports.save ; fi
    /bin/cp -vf exports.new /etc/exports

    if [ -e fstab.new ] ; then rm fstab.new; fi
    if [ -e /etc/fstab ] ; then cp -fv /etc/fstab fstab.new ; else touch
fstab.new; fi
```

```
    echo "/export/home                 /NFSV4exports/home       none    bind
0 0" >> fstab.new
    echo "/export/share/atlas          /NFSV4exports/share/atlas none    bind
0 0" >> fstab.new
    echo "/export/share/condor-etc     /NFSV4exports/share/condor-etc none    bind
0 0" >> fstab.new
    echo "/export/share/pilot          /NFSV4exports/share/pilot none    bind
0 0" >> fstab.new
    echo "/export/share/pandat3-output     /NFSV4exports/share/pandat3-output none
bind            0 0" >> fstab.new

    if [ -e /etc/fstab ] ; then /bin/cp -v /etc/fstab /etc/fstab.save; fi
    /bin/cp -vf fstab.new /etc/fstab
    echo "[configure-nfsv4.sh] mount all of bind mounts with command - mount -va -
t none "
    mount -va -t none
    echo "[configure-nfsv4.sh] /bin/df -h"
    /bin/df -h
    /sbin/service nfs restart
    /sbin/chkconfig nfs on
else
    if [ -e /export ] ; then echo "/export exists" else mkdir /export ; fi
    if [ -e auto.export ] ; then rm auto.export; fi
    echo "/export/home -fstype=nfs4 -rw,nodev,nosuid ${server name}:/home " >>
auto.export
    echo "/export/share/atlas -fstype=nfs4 -rw,nodev,netdev,nosuid
${server_name}:/share/atlas" >> auto.export
    echo "/export/share/condor-etc -fstype=nfs4 -ro,nodev,nosuid
${server name}:/share/condor-etc " >> auto.export
    echo "/export/share/pilot -fstype=nfs4 -rw,nodev,nosuid
${server_name}:/share/pilot " >> auto.export
    echo "/export/share/pandat3-output -fstype=nfs4 -rw,nodev,nosuid
${server_name}:/share/pandat3-output " >> auto.export

    if [ -e /etc/auto.export ] ; then /bin/cp -v /etc/auto.export
/etc/auto.export.save; fi
    /bin/cp -vf auto.export /etc/auto.export
    echo "#  Direct mount of the /export/home /export/share areas " >>
/etc/auto.master
    echo "/- /etc/auto.export" >> /etc/auto.master
    echo " " >> /etc/auto.master
    /sbin/service autofs restart
fi

/sbin/chkconfig --level 345 rpcidmapd on
/sbin/service rpcidmapd restart
```

Şekil A.21: "configure-nfsv4-public.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post_install/scripts:$PATH
source $HOME/post install/node-environment.config
echo "---------------------------------- Condor section --------------------------
-----"
for file in condor config.cluster condor config.head.local
condor_config.interactive.local condor_config.worker.local
do
  fetch config file $file
  ret code=$?
  if [ $ret code != "0" ] ; then
      echo "Failed to fetch "$file
      exit 1
  fi
  if [ $file == "condor config.head.local" ] ; then
    cat $HOME/post install/config files/${file} | sed
's/192.168.100.1/'${CONDOR HEADNODE}'/g' >
$HOME/post_install/config_files/${file}.sed
    mv -v $HOME/post_install/config_files/${file}.sed /export/share/condor-
etc/$file
     ret code=$?
  else
     mv -v $HOME/post install/config files/$file /export/share/condor-etc/
     ret_code=$?
  fi
  if [ $ret code != "0" ] ; then
      echo "Failed to move "$file
```

```
      exit 1
  fi
done

/bin/echo "#  added  during condor installation step on nfs server" >>
/export/share/condor-etc/condor config.cluster
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "Failed to add information to /export/share/condor-
etc/condor_config.cluster"
    exit 1
fi

/bin/echo "CONDOR HOST="$CONDOR HEADNODE >> /export/share/condor-
etc/condor_config.cluster
ret_code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "Failed to add information to /export/share/condor-
etc/condor config.cluster"
    exit 1
fi

/bin/echo "USE NFS = TRUE" >> /export/share/condor-etc/condor config.cluster
ret code=RC$?
if [ $ret_code != "RC0" ] ; then
    echo "Failed to add information to /export/share/condor-
etc/condor_config.cluster"
    exit 1
fi

/bin/echo "UID_DOMAIN="$NETWORK_DOMAIN_NAME >> /export/share/condor-
etc/condor_config.cluster
ret_code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "Failed to add information to /export/share/condor-
etc/condor config.cluster"
    exit 1
fi

/bin/chown -R condor:condor /export/share/condor-etc
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo " Failed to change ownership of condor configuration files "
    exit 1
fi

exit 0
```

Şekil A.22: "nfsnode-condor-public.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post install/scripts:$PATH
source $HOME/post install/node-environment.config
if [ ! -e $HOME/post install/logs/ ] ; then mkdir -p $HOME/post install/logs ; fi;
date stamp=`date +%d%m%Y %H%M%S`

config_firewall="PENDING"
config local users="PENDING"
config nfsv4="PENDING"
config ganglia="PENDING"
config_ldap="PENDING"
config_condor="PENDING"
config cvmfs="PENDING"
config mail alias="PENDING"
config kerbos="PENDING"
config additional lib="PENDING"

echo "--------------------- firewall setup --------------------"
fetch shell script interactivenode-firewall-public.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config firewall="FAILED"
else
  $HOME/post_install/scripts/interactivenode-firewall-public.sh >&
$HOME/post install/logs/interactivenode-firewall-$date stamp.log
  ret code=RC$?
```

```
  if [ $ret code == "RC0" ] ; then
     config firewall="PASSED"
  else
     config_firewall="FAILED"
  fi
fi

echo "---------------------- fetch additional libraries setup ---------------------"

fetch_shell_script fetch_additional_libraries.sh
ret code=$?
if [ $ret code != "0" ] ; then
    config additional lib="FAILED"
else
    $HOME/post_install/scripts/fetch_additional_libraries.sh >&
$HOME/post_install/logs/interactivenode-additional_libs_$date_stamp.log
    ret code=RC$?
    if [ $ret code == "RC0" ] ; then
       config additional lib="PASSED"
    else
       config_additional_lib="FAILED"
    fi
fi

echo "---------------------- kerberos setup ---------------------"

fetch_shell_script  fetch_kerberos_config_file.sh
ret code=$?
if [ $ret code != "0" ] ; then
    config kerberos="FAILED"
else
    $HOME/post_install/scripts/fetch_kerberos_config_file.sh
    ret_code=RC$?
    if [ $ret code == "RC0" ] ; then
       config kerberos="PASSED"
    else
       config_kerberos="FAILED"
    fi
fi

echo "---------------------- cvmfs setup ---------------------"

fetch_shell_script configure-cvmfs.sh
ret_code=$?
if [ $ret code != "0" ] ; then
   config cvmfs="FAILED"
else
  $HOME/post_install/scripts/configure-cvmfs.sh >&
$HOME/post_install/logs/interactivenode-configure-cvmfs-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config cvmfs="PASSED"
  else
     config_cvmfs="FAILED"
  fi
fi

echo "---------------------- ganglia setup ---------------------"
fetch shell script gmond-client.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config ganglia="FAILED"
else
  $HOME/post install/scripts/gmond-client.sh >&
$HOME/post install/logs/interactivenode-ganglia-$date stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config ganglia="PASSED"
  else
     config ganglia="FAILED"
  fi
fi

echo "---------------------- local users setup ---------------------"
fetch shell script local users.sh
ret code=$?
if [ $ret_code != "0" ] ; then
```

```
   config local users="FAILED"
else
  $HOME/post_install/scripts/local_users.sh >&
$HOME/post_install/logs/interactivenode-local_users-$date_stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config local users="PASSED"
  else
     config_local_users="FAILED"
  fi
fi

echo "--------------------- nfsv4 client setup --------------------"
fetch shell script nfsv4-client.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config nfsv4="FAILED"
else
  $HOME/post install/scripts/nfsv4-client.sh >&
$HOME/post install/logs/interactivenode-nfsv4-$date stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config nfsv4="PASSED"
  else
     config_nfsv4="FAILED"
  fi
fi

echo "--------------------- condor setup --------------------"
fetch shell script interactivenode-condor.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config_condor="FAILED"
else
  $HOME/post install/scripts/interactivenode-condor.sh >&
$HOME/post install/logs/interactivenode-condor-$date stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config_condor="PASSED"
  else
     config condor="FAILED"
  fi
fi

echo "--------------------- mail alias setup --------------------"
fetch shell script mail alias.sh
ret code=$?
if [ $ret_code != "0" ] ; then
   config_mail_alias="FAILED"
else
  $HOME/post install/scripts/mail alias.sh >&
$HOME/post install/logs/interactivenode-mail alias-$date stamp.log
  ret code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config_mail_alias="PASSED"
  else
     config mail alias="FAILED"
  fi
fi

/usr/bin/yum -y update
```

Şekil A.23: "post-install-interactive-configuration-public.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post install/scripts:$PATH
source $HOME/post_install/node-environment.config
echo "---------------------------------- Condor section --------------------------
-----"

if [ ! -e /etc/yum.repos.d/condor-stable-rhel6.repo ] ; then
    echo "[`/bin/basename $0`] Fetch condor yum repository"
    true
    ret_code=RC$?
    if [ $ret code != "RC0" ] ; then
       echo " Could not fetch the condor yum repository "
```

```
        exit 1
    fi
fi
#  Install Condor with yum
echo "[`/bin/basename $0`] /usr/bin/yum -y install condor"
/usr/bin/yum -y install condor
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo "Could not install condor with yum"
    exit 1
fi

echo "[`/bin/basename $0`] fetch shell script create-condor-credential-file.sh"
fetch shell script create-condor-credential-file.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
    echo "could not fetch script to create condor-credential-file.sh"
    exit 1
fi

echo "[`/bin/basename $0`] $HOME/post_install/scripts/create-condor-credential-
file.sh"
$HOME/post install/scripts/create-condor-credential-file.sh
ret code=RC$?
if [ $ret_code != "RC0" ] ; then
    echo " Could not create condor credential file "
    exit 1
fi

echo "[`/bin/basename $0`] modify /etc/condor/condor config.local"
#   add information at the end of the condor_config file to replace information
defined earlier
/bin/echo "##  Next configuration to be read is for the T3 cluster setup" >>
/etc/condor/condor config.local
ret code=RC$?
if [ $ret code != "RC0" ] ; then
    echo " Could not modify /etc/condor/condor_config.local"
    exit 1
fi
/bin/echo "LOCAL CONFIG FILE = /export/share/condor-etc/condor config.cluster
/export/share/condor-etc/condor config.interactive.local" >>
/etc/condor/condor config.local
ret_code=RC$?
if [ $ret_code != "RC0" ] ; then
    echo " Could not modify /etc/condor/condor config.local"
    exit 1
fi

/bin/echo "NETWORK_INTERFACE = "$PRIVATE_IP >> /etc/condor/condor_config.local
ret_code=RC$?
if [ $ret code != "RC0" ] ; then
    echo " Could not modify /etc/condor/condor config.local"
    exit 1
fi

echo "[`/bin/basename $0`] check if can read the condor configuration files "
if [ ! -e /export/share/condor-etc/condor config.cluster ] ; then
   if [ ! -e /export/share/condor-etc/condor config.interactive.local ] ; then
       echo "Could not find remote config files"
       exit 1
   fi
fi

/sbin/chkconfig --level 235 condor on
service condor start

exit 0
```

Şekil A.24: "interactive-condor.sh" Dosyası

```
#!/bin/bash

/usr/sbin/condor_store_cred -f /var/lib/condor/condor_credential -p atlast3
```

Şekil A.25: "create-condor-credential-file.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post install/scripts:$PATH
source $HOME/post_install/node-environment.config

wget -nv -O /etc/yum.repos.d/cernvm.repo
http://cvmrepo.web.cern.ch/cvmrepo/yum/cernvm.repo
ret code=RC$?
if [ $ret code != "RC0" ] ; then exit 1 ; fi

wget -nv -O /etc/pki/rpm-gpg/RPM-GPG-KEY-CernVM
http://cvmrepo.web.cern.ch/cvmrepo/yum/RPM-GPG-KEY-CernVM
ret code=RC$?
if [ $ret code != "RC0" ] ; then exit 1 ; fi

yum -y install fuse cvmfs cvmfs-init-scripts cvmfs-auto-setup
ret_code=RC$?
if [ $ret code != "RC0" ] ; then exit 1 ; fi

/sbin/chkconfig --add cvmfs
ret code=RC$?
if [ $ret_code != "RC0" ] ; then exit 1 ; fi

/sbin/chkconfig --level 345 cvmfs on
ret code=RC$?
if [ $ret_code != "RC0" ] ; then exit 1 ; fi

if [ ! -e /var/cache/cvmfs ] ; then
    mkdir -p /var/cache/cvmfs
    ret code=RC$?
    if [ $ret code != "RC0" ] ; then exit 1 ; fi
fi

if [ ! -e /etc/cvmfs/ ] ; then mkdir -p /etc/cvmfs/ ; fi
if [ ! -e /etc/cvmfs/default.local ] ; then
    touch /etc/cvmfs/default.local
    ret code=RC$?
    if [ $ret_code != "RC0" ] ; then exit 1 ; fi
fi

echo "#" >> /etc/cvmfs/default.local
echo "#  Local configuration file for cvmfs " >> /etc/cvmfs/default.local
echo "#" >> /etc/cvmfs/default.local
echo "CVMFS_CACHE_DIR=/var/cache/cvmfs" >> /etc/cvmfs/default.local
echo "CVMFS_REPOSITORIES=atlas" >> /etc/cvmfs/default.local
echo 'CVMFS HTTP PROXY="http://'$SQUID NAME':3128;DIRECT"' >>
/etc/cvmfs/default.local
ret code=RC$?
if [ $ret_code != "RC0" ] ; then exit 1 ; fi

service cvmfs restartautofs
ret code=RC$?
if [ $ret code == "RC0" ] ; then
    echo "Finished configuring cvmfs "
else exit 1 ; fi

exit 0
```

Şekil A.26: "configure-cvmfs.sh" Dosyası

```
#!/bin/bash
export PATH=$HOME/bin:$HOME/post_install/scripts:$PATH
source $HOME/post_install/node-environment.config

if [ ! -e $HOME/post install/logs/ ] ; then mkdir -p $HOME/post install/logs ; fi;
date stamp=`date +%d%m%Y %H%M%S`

config_firewall="PENDING"
config_local_users="PENDING"
config nfsv4="PENDING"
config ganglia="PENDING"
config ldap="PENDING"
config condor="PENDING"
config_cvmfs="PENDING"
config_mail_alias="PENDING"
config kerbos="PENDING"
config addition lib="PENDING"
```

```
config restrict users="PENDING"
config file limits="PENDING"
config_xrootd_mounts="PENDING"

echo
echo "---------------------- firewall setup --------------------"
echo
fetch shell script workernode-firewall-public.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config firewall="FAILED"
else
  $HOME/post install/scripts/workernode-firewall-public.sh >&
$HOME/post install/logs/workernode-firewall-$date stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config firewall="PASSED"
  else
     config firewall="FAILED"
  fi
fi

echo "---------------------- fetch additional libraries setup --------------------"

fetch_shell_script fetch_additional_libraries.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
    config additional lib="FAILED"
else
    $HOME/post install/scripts/fetch additional libraries.sh >&
$HOME/post_install/logs/workernode-addition_lib-$date_stamp.log
    ret_code=RC$?
    if [ $ret_code == "RC0" ] ; then
       config additional lib="PASSED"
    else
       config additional lib="FAILED"
    fi
fi

echo "---------------------- kerberos setup --------------------"
fetch shell script  fetch kerberos config file.sh
ret code=$?
if [ $ret_code != "0" ] ; then
    config_kerberos="FAILED"
else
    $HOME/post install/scripts/fetch kerberos config file.sh >&
$HOME/post install/logs/workernode-kerberos-$date stamp.log
    ret_code=RC$?
    if [ $ret_code == "RC0" ] ; then
       config_kerberos="PASSED"
    else
       config kerberos="FAILED"
    fi
fi

echo "---------------------- cvmfs setup --------------------"

fetch shell script configure-cvmfs.sh
ret code=$?
if [ $ret_code != "0" ] ; then
   config_cvmfs="FAILED"
else
  $HOME/post install/scripts/configure-cvmfs.sh >&
$HOME/post install/logs/workernode-configure-cvmfs-$date stamp.log
  ret code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config cvmfs="PASSED"
  else
     config cvmfs="FAILED"
  fi
fi

echo "---------------------- ganglia setup --------------------"
fetch shell script gmond-client.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config_ganglia="FAILED"
```

```
else
  $HOME/post install/scripts/gmond-client.sh >&
$HOME/post_install/logs/workernode-ganglia-$date_stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config ganglia="PASSED"
  else
     config ganglia="FAILED"
  fi
fi

echo "---------------------- local users setup --------------------"
fetch shell script local users.sh
ret code=$?
if [ $ret_code != "0" ] ; then
   config_local_users="FAILED"
else
  $HOME/post install/scripts/local users.sh >& $HOME/post install/logs/workernode-
local users-$date stamp.log
  ret code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config_local_users="PASSED"
  else
     config local users="FAILED"
  fi
fi

fetch shell script increase file limits.sh
ret code=$?
if [ $ret code != "0" ] ; then
   config_file_limits="FAILED"
else
  $HOME/post_install/scripts/increase_file_limits.sh >&
$HOME/post install/logs/workernode-file limits-$date stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config_file_limits="PASSED"
  else
     config_file_limits="FAILED"
  fi
fi

fetch_shell_script restrict_login_access.sh
ret_code=$?
if [ $ret code != "0" ] ; then
    echo "Failed to fetch the restrict login access.sh"
    echo " Fetch restrict login access.sh  - FAILED" >>
$HOME/post_install/logs/workernode-configuration.log
   config_local_users="FAILED"
else
    echo
    echo " You will need to execute the script later by hand :"
    echo "$HOME/post install/scripts/restrict login access.sh"
    echo " Fetch restrict_login_access.sh  - PASSED" >>
$HOME/post_install/logs/workernode-configuration.log
fi

echo "---------------------- nfsv4 client setup --------------------"
fetch shell script nfsv4-client.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config nfsv4="FAILED"
else
  $HOME/post install/scripts/nfsv4-client.sh >&
$HOME/post install/logs/workernode-nfsv4-$date stamp.log
  ret_code=RC$?
  if [ $ret code == "RC0" ] ; then
     config nfsv4="PASSED"
  else
     config nfsv4="FAILED"
  fi
fi

echo "---------------------- condor setup --------------------"
fetch shell script workernode-condor.sh
ret code=$?
if [ $ret_code != "0" ] ; then
```

```
   config condor="FAILED"
else
  $HOME/post_install/scripts/workernode-condor.sh >&
$HOME/post_install/logs/workernode-condor-$date_stamp.log
  ret code=RC$?
  if [ $ret code == "RC0" ] ; then
     config condor="PASSED"
  else
     config_condor="FAILED"
  fi
fi

echo "--------------------- mail alias setup --------------------"
fetch shell script mail alias.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
   config mail alias="FAILED"
else
  $HOME/post install/scripts/mail alias.sh >& $HOME/post install/logs/workernode-
mail alias-$date stamp.log
  ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config mail alias="PASSED"
  else
     config_mail_alias="FAILED"
  fi
fi

echo "-----------------------   xrootd file mounts -------------- "
fetch shell script make xrootd file mounts.sh
ret_code=$?
if [ $ret_code != "0" ] ; then
    config_xrootd_file_mounts="FAILED"
else
    $HOME/post install/scripts/make xrootd file mounts.sh >&
$HOME/post install/logs/workernode-xrootd-mounts-$date stamp.log
    ret_code=RC$?
  if [ $ret_code == "RC0" ] ; then
     config_xrootd_file_mounts="PASSED"
  else
     config xrootd file mounts="FAILED"
  fi
fi

/usr/bin/yum -y update
```

Şekil A.27: “post-install-workernode-configuration-public.sh” Dosyası

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Ad Soyad **:** Agah Alıcı
Doğum Tarihi **:** 14.11.1981
Doğum Yeri **:** Nürtingen
E-mail **:** agah.alici@gmail.com, agah@aydin.edu.tr

ÖĞRENIM DURUMU

Yüksek Lisans **:** 2018, İstanbul aydın Üniversitesi, Bilgisayar Mühendisliği
Lisans **:** 2006, Sakarya Üniversitesi, Elektrik-Eloktronik Mühendisliği

DİL BİLGİSİ

İngilizce **:** İyi Seviye