Basic unix commands and quick Sheet for truba
http://evomics.org/learning/unix-tutorial/

Basic Commands		
pwd	Print working directory	
hostname	çalıştığın bilin adını verir	
clear	terminal ekranını temizler	
ps	o anda bilde çalışanları gösteri	
htop		
cd	cd ~/directory/	Change Directory
cd ..		Doksyadan çıkmayı sağlar
mkdir	mkdir newdirectory	Make Directory newdirectory
mv	mv [-f] ~/source ~/destination	Move or rename file or directory
cp	cp [-fr] ~/source ~/newcopy	Copy a file or directory
ls	ls [-lhtr] [file][directory]	list all files or files in directory
sort	sort [file]	dosya içindekielri ekrana yazar
rm	rm ~/file	Delete file file
rm -r	İç içe dosyaları siler	
man	man command name	Show manual page on command command
ssh	ssh hostname [-l username]	Connect to remote host with user name username
scp	scp ~/source hostname:~/dest/	Copy file source to host hostname
vi	vi [~/file]	A powerful text editor
nano	nano [~/file]	A pico like text edito
emacs	emacs [~/file]	A powerful text editor
cat	dosya oluşturmak dosyaların içeriğini kopyalamak	
less	doyayı aşar manupule etmeni sağlar	
gedit	dökuman açılmasını sağlar	
open	dosyayı açar	
grep	grep [-iv] string file	Search for string in file
wc	wc [-l] file	Count the number of word/lines/chars in file
tail	tail [-f][-n#] file	Show last few lines of file
head	head [-n#] file	Show first few lines of file
dos2unix	dos2unix file	Change dos files into Unix/Linux files
df	df [-h][/home]	Show the size and amount free of your quota
du	du [-sh][directory]	Show total size of items in directory
Ctrl C	sonlandırma tüm çalışan pfrogramlar için geçerli	
:wq	vim ile açılan dosyadan çıkmak ctrl C ve sonra :wq kullanılır	
zip	zip [file]	zipleme yapar

Temel SLURM komutları		
Temelde sbatch ve squeue komutları iş göndermek işinizin durumunu izlemek için yeterli olsa da ek bazı SLURM komutlarını biliyor ve kullanıyor olmak, kullanım sırasında karşılaşılabilecek pek çok sorunu hızlıca çözmek ve sistemin genelini daha iyi kavramak açısından önemlidir.		
--mem-per-core		
sinfo	Kuyrukların kullanım durumuna, paylaşılan dolu ya da boş olan node ve çekirdeklerin durumunu ekranda gösterir	Buradan edinilecek bilgi ile kuyruğa gönderilecek işin kaynak miktarı planlanarak en hızlı şekilde başlayabileceği kuyruğa yönlendirilebilir. Kullanılacak ek parametrelerle, listelenecek bilginin türü ve miktarı değiştirilebilir.
squeue	Kullanıcının kuyruklarda bekleyen ve çalışan işlerini görüntüler.	Kullanılacak ek parametrelerle, listelenecek bilginin türü ve miktarı değiştirilebilir. Kullanıcının tüm işleri listelenebileceği gibi (varsayılan), işin iş numarası parametre olarak verilerek, o işe özel bilgilerin dökümü de alınabilir.
sbatch	Hazırlanan iş betiğini kuyruğa göndermek için kullanılır.	Parametreler betik dosyasında verilebileceği gibi komutun yanına da yazılabilir. İşin akışını, verimini ve kontrolünü sağlayacak pek çok parametresi vardır.
srun	Alternatif olarak işler sbatch yerine srun	srun kullanılırken çalıştırılacak komut doğrudan srun ve

	komutu ile de çalıştırılabilir.	parametrelerinin sonuna yazılır. Basit ve küçük işleri çalıştırmak için hızlı bir yöntemdir. Sbatch'de olduğu gibi pek çok önemli parametresi vardır.
scancel	Kuyrukta sırada bekleyen ya da o anda çalışmakta olan işleri iptal etmek için kullanılır.	
salloc	Komut satırından interaktif iş çalıştırmak için kullanılır.	Salloc komutu ile öncelikle istenilen kaynak miktarı "allocate" edilerek salloc komut satırına geçilir, sonrasında srun komutu işe işler interaktif olarak çalıştırılır. Çalışma sırasında kullanıcı işin çalışmasına müdahale edebilir.
scontrol	Küme, kuyruk (partition) ya da herhangi bir iş ile ilgili bilgilerin dökümü alınabilir, izin verildiği ölçüde, müdahale edilebilir.	Kuyruğa gönderilmiş olan işler üzerinde, işleri silmeden güncelleme yapılmasına imkan sağlar.
sacct	Beklemede, çalışmakta ya da daha önce çalışmış ve sonlanmış olan işler ya da tek bir iş hakkında ayrıntılı rapor ve bilgi alınmasına imkan verir.	Pek çok parametre içerir. Örneğin belli tarihler arasında başlamış ve bitmiş işlerin listesi, çalışma süresi, kullandığı bellek miktarı, üzerinde çalıştığı sunucuların adresleri vs, gibi iş/işler ile ilgili bilgi alınması mümkündür.
sstat	Çalışmakta olan işin kullandığı, kullanmakta olduğu sistem kaynakları hakkında bilgi verir	**--format=** ile verilecek bilgi türleri seçilebilir.
scontrol show partition=	scontrol show partition=kuyru_adi	Herhangi bir kuyruğun bilgisine ulaşabilmek için

SLURM betiği özellikleri		
Herhangi bir SLURM betiği aslında BASH kabuk betiğidir. BASH kabuk programlama dilinin tüm öğelerini bu betikte kullanarak, pek çok detaylı analizi iş kuyruklarında yaptırmak mümkündür. BASH dilini betikte kullanarak, SLURM'un (ya da herhangi bir başla kaynak yöneticisinin) sağlayamayacağı esnekliği betiklere eklemek mümkündür. Kabuk programlama ile ilgili wiki sayfamızda bulunan "Temel Kabuk Programlama" dökümanına göz atabilirsiniz.		
Kabuk programlamadan bağımsız olarak, Herhangi bir SLURM betiği temel olarak 3 ana bölümden oluşur. Başlangıç (Parametreler, tanımlar), gövde ve iş başlatma/çıkış.		
Başlangıç		
Bu bölümde, hesaplama sırasında kullanılacak banka hesabı ve işin çalışacağı yer, zaman sınırı, çekirdek ve node sayısı gibi işin temel özellikleri tanımlanır. Her tanım satırı #SBATCH işe başlar. Bu bölümde işin akışını belirleyecek pek çok tanım yapılabileceği gibi, temel birkaç tanımın yapılması işin çalışması için yeterlidir. Bu tanımların bir kısmı betik dosyasının içerisinde yapılabileceği gibi, bir kısmı sbatch komutu ile dosya kuyruğa gönderilirken komutun yanında parametre olarak da kullanılabilir. örneğin sbatch -N 4 -n 16 is_dosyasi_adi		
#!/bin/bash	Betiği yorumlayacak interptreter.	Bu şekilde kalması gerekir.
#SBATCH -p	İşin çalıştırılacağı kuyruk adı (partition)	
#SBATCH -A	İş için kullanılacak bankaccount.	
	Grup halinde yapılan çalışmalarda kullanıcı adından farklı olabilir. ARDEB kanalı ile açılan TBAG proje hesaplarına ayrıcalıklı olarak iş göndermek için, bu kısma ilgili (ve kullanıcı hesabınız için izin verilen) tbag hesabının yazılması gerekir.	
#SBATCH -J	İşin kuyrukta görülecek adı.	
#SBATCH -n	Görev sayısı (mpi işleri için, uygulamanın çalıştırılacağı kopya sayısı). Normalde sbatch herhangi bir görev çalıştırmaz. İş, çalışacağı sunucuya düşüp, master nodda çalışmaya başladığında, betik linux komut satırından çalıştırılmış gibi davranır. Betikte işin çalıştırılacağı komut satırında özel bir durum belirtilmemişse işin tek kopyası çalıştırılır. Ancak satırda mpirun ya da mpiexec komutları kullanılmışsa, ilgili uygulamanın **-n** kopyası çalıştırılır.	
#SBATCH -c	Her bir görev için kullanılabilecek en fazla çekirdek sayısını belirtir (cores per task) . Varsayılan değeri 1'dir. Eğer hibrid işler (mpi+openmp) yada multitask (openmp sadece) çalıştırılacaksa, bu parametrenin kullanılması gerekir. Değeri OMP_NUM_THREADS değişkeninin değeri işe aynı olacak şekilde seçilmelidir. Değeri 1 sunucudaki çekirdek sayısından fazla olamaz. Eğer aynı sunucusu üzerinden 1 den fazla task (n) çalıştırılacaksa, sunucu başına düşen görev sayısı x görev başına düşen çekirdek miktarı en fazla ilgili sunucudaki çekirdek sayısı kadar olabilir. Örnegin 2 adet 28 çekirdekli sunucunun tamamı kullanarak 8 processli (task) bir MPI+openmp hibrid işi çalıştırılacaksa, process başına en fazla 7 çekirdek kullanılabilir. (sunucu başına 4 process düşer, sunucuda toplam 28	

	çekirdek varsa, her bir process için en fazla 7 çekridek kullanılabilir)
#SBATCH --threads	işlemcilerin hyperthreading özelliklerini kullanmak için tanımlanır.. Mevcut işlemcilerde çekirdek başına 2 thread düşmektedir. Örnegin 28 çekirdekli bir sunucuda, bir openmp işini 56 threadle (OMP_NUM_THREADS=56) çalıştırabilmek için -N 1 -n1 -c28 --threads=2 tanımı kullanılabilir..
#SBATCH --mem=	Bu parametre ile iş için toplamda en fazla ne kadar bellek kullanılacağı belirtilmektedir. Kullanımı zorunlu değildir.
	Eğer Bu parametre kullanılmazda her bir çekirdek için DefMemPerCore kadar bellek ayrılır. Eğer daha fazla belleğe ihtiyaç duyulacaksa, bu parametre ile ihtiyaç duyulan bellek miktarı arttırılabilir. Ancak bu değer en fazla çekirdek_sayisi x MaxMemPerCore kadar arttırılabilir.
#SBATCH --mem-per-core=	Bu parametre ile her bir çekirdek için ihtiyaç duyulan bellek miktarı belirtilir. Kullanımı zorunlu değildir.
	Eğer Bu parametre kullanılmazda her bir çekirdek için DefMemPerCore kadar bellek ayrılır.. Eğer daha fazla belleğe ihtiyaç duyulacaksa, bu parametre ile ihtiyaç duyulan bellek miktarı arttırılabilir. Ancak bu değer en fazla MaxMemPerCore kadar olabilir.
#SBATCH –time=	İşin en fazla çalışma süresi. Bu süre zarfında tamamlanmamış olan işler, zaman dolduğunda otomatik olarak öldürülürler.
	Burada verilecek değer ilgili kümenin sınırından yüksek olamaz. Herhangi bir değer verilmeden gönderilen işler, çalışmaya başladıktan 1 dakika sonrasında sistem tarafından otomatik olarak sonlandırılırlar
#SBATCH –no-requeue	**requeue** İş çalışırken bazı durumlarda, hesaplama sunucusundan kaynaklı sebeplerle, iş hata alarak sonlanabilir, bu durumda işi kuyruğa otomatik olarak yeniden gönderilir
	İşin kuyruğa yeniden gönderilmesi genelde faydalı bir özelliktir, ancak eğer iş kaldığı yerden devam edecek şekilde yapılandırılmamışsa yada kullanılan uygulamanın böyle bir özelliği yoksa ve o ana kadar üretilen verilerin üzerine yeni verilerin yazılma ihtimali varsa, bu opsiyon kullanılarak işin kuyruğa otomatik olarak yeniden gönderilmesi engellenebilir.
#SBATCH –output=	İş çalışırken, kullandığınız uygulamanın ya da betiğinizde kullandığınız bash programla öğelerinin ekrana basacağı (STDOUT) bilgilerin yazılacağı dosyabın tam adresi ve adı.
	Bu adresin scratch dizinde olması zorunludur.
#SBATCH –error=	İş çalışırken, kullandığınız uygulamanın yada betiğinizde kullandığınız bash programla öğelerinin ekrana basacağı hata mesajlarının (STDERR) yazılacağı dosyanın tam adresi ve adı.
	Bu adresin scratch dizinde olması zorunludur. Eğer --error ve --output parametreleri belirtilmezse, tüm ekran çıktısı otomatik olarak slurm-JOBID.out dosyasına yönlendirilir.
#SBATCH -N	Hesaplama sırasında, kullanılacak çekirdeklerin kaç farklı node tarafından sağlanacağını belirler. Herhangi bir tanım girilmemişse, çekirdekler rasgele sayıdaki nodelardan rastgele sayıda sağlanırlar.
	Node sayısı için herhangi bir tanımlama yapmamak işlerin mümkün olan en hızlı şekilde başlamasını sağlar, ancak performans testlerinde alınacak sonuç, her iş için farklı olabilir, Eğer talep edilen çekirdeklerin nodelar tarafından eşit sayıda sağlanması istemiyorsa, -n -N parametresi yerine --ntasks-per-node ve -N parametreleri birlikte kullanılmalıdır. Örnegin işiniz için toplamda 16 çekirdeğin 4 sunucu tarafından eşit sayıda sağlanmasını istiyorsanız -N 4 --ntasks-per-node=4 parametresini kullanmalısınız.
#SBATCH -M	Birden fazla hesaplama kümesin tek bir arayüz üzerinden hizmet verdiği durumlarda, işin gideceği kümeyi belirtir.
	TRUBA'da şu an için farklı hesaplama kümeleri farklı kullanıcı arayüzlerinden hizmet vermektedirler.
#SBATCH –workdir=	işin başlayıp, output err dosyalarının yazılacağı dizinin adresidir. Scratch dizini işaret ediyor olması zorunludur.
	Eğer herhangi bir tanımlama yapılmaz ise, varsayılan olarak iş gönderilirken o an içinde bulunan dizin workdir dizini olarak kabul edilir.
#SBATCH –gres=	Ekstra özelliklerin sunulduğu kuyruklarda bu ekstra özelliklerin ve onlardan ne kadarının kullanılacağını belirtir.
	Cuda kuyruğundaki GPU kartlarını kullanabilmek için bu tanımın yapılması gerekir. Örneğin SBATCH –gres=gpu:1

#SBATCH –mail-type=	İş kuyruğa gönderildikten sonra, iş ile ilgili ne tür epostaların gönderileceğini tanımlar. BEGIN, END, FAIL, REQUEUE, ALL değerlerini alabilir.
	Herhangi bir tanım yapılmaz ise kullanıcı e-posta ile bilgilendirilmez.
#SBATCH –mail-user=	Yukarıda tanımlanan durumlarda e-postanın gönderileceği adresi tanımlar.

Gövde
Her program ve kullanıcı için gövde kısmı farklı olabilir. Bu kısımda işi çalıştırmadan önce yapılması gereken ön çalışma yapılır; load edilmesi gereken kütüphaneler, varsa çevre değişkenler vs. yüklenir. Kabuk dili öğeleri kullanılarak ön kontroller yapılarak gerekli dosyaların varlığı, içeriği vs. kontrol edilebilir. Bu kısım kullanıcın deneyimine ve ihtiyaçlarına göre şekillenir. Ancak standart olarak iş ile ilgili temel bilgilerin STDOUT'a yazılması daha sonra işi analiz ya da debug etmek için faydalı olabilir. Örneğin: aşağıdaki kısım da herhangi bir gaussian işini çalıştırmak için ihtiyaç duyulan kütüphaneler load edilerek çevre değişkenleri ayarlanıyor ve kullanılan kaynağın özellikleri STDOUT'a basılıyor.
echo "SLURM_NODELIST $SLURM_NODELIST" echo "NUMBER OF CORES $SLURM_NTASKS"
export OMP_NUM_THREADS=1 export g09root=$HOME export GAUSS_SCRDIR=/tmp . $g09root/g09/bsd/g09.profile
İşin çalıştırılması ve bitiş
Gövde kısmında işin programın çalıştırılması için gerekli kütüphaneler, çevre değişkenleri load edildikten ve gerekli kontroller yapıldıktan sonra, sıra işin çalıştırılmasına gelir. İş çalıştırma satırı, normalde işi komut satırından elle çalıştırırken kullanılan komut satırı ile aynıdır. Herhangi bir gaussian işi işin bu satır aşağıdaki gibi olabilir örneğin
$g09root/g09/g09 < gaussian_egitim.com exit
MPI işler için SLURM'un sağladığı bazı esneklikler ve kullanım kuralları vardır. Hesaplama sırasında kullanılak çekirdek sayısı ve host bilgisi yazılmasına OpenMPI (ve diğer bazı MPI kütüphanelerinde) gerek yoktur. Bu bilgi mpirun komutuna doğrudan kaynak yöneticisi tarafından sağlanır. Örneğin komut satırından bir MPI işini 4 çekirdek çalıştırırken normalde
mpirun -np 4 –machinefile=hosts_dosyasi <uygulamanin_tam_adresi_ve_adi> exit
gibi bir komut verilmesi gerekirken SLURM betiğinde aşağıdaki satır kullanılmalıdır.
mpirun <uygulamanin_tam_adresi_ve_adi> exit
Eğer işin o ana kadar kullanmış olduğu sistem kaynakları (bellek, walltime, runtime, disk vs) hakkında detaylı bilgi alınmak isteniyorsa **exit** satırından önce
sstat -j $SLURM_JOB_ID
komutunu yazabilirsiniz.
Örnek betik dosyaları
Her kullanıcın deneyimi, ve kullanacağı uygulamanın yeri, özellikleri, versiyonu , ihtiyaç duyduğu kaynak türü ve miktarı, derlendiği ortam ve kütüphanelere göre, herhangi bir uygulamayı çalıştırmak için kullanılabilecek betik dosyası farklılıklar gösterebilir.
Teknik birim tarafından, tüm kullanıcıların kullanımı için standart özelliklerle derlenerek ortak dizine kurulmuş uygulamaların pek çoğu için örnek betik dosyaları hazırlanarak, kullanıcıların kullanımına sunulmuştur.
Örnek betik dosyalarına Mercan/Lüfer kümesi için /home_palamut1/scripts, Levrek kümesi için /truba/sw/scripts dizininden ulaşılabilir. Kullanıcıların burdaki betik dosyalarını kullanabilmesi için, onları scratch'deki kendi dizinlerine kopyalamaları ve betik dosyasında verilmiş tanımları kendi hesaplanın ve işlerinin özelliklerine göre değiştirmeleri gereklidir.
Betik dosyalarını içinde bulundukları dizinle birlikte kopyalamakta fayda vardır. Zira ilgili dizin içinde, uygulamanın test amacı ile çalıştırılması için örnek input dosyaları da bulunmaktadır.

PBS Commands		
qstat	qstat [-u user][-a][-n][jobid]	Show current state of queues
qsub	qsub [-l resources][-I] pbsfile	Submit job with definitions in pbsfile
qdel	qdel jobid	Delete queued/running job with id jobid
qalter	qalter [-l walltime=#] jobid	Alter required walltime/memory for job jobid
checkjob	checkjob jobid	Verbose information on job jobid

qpeek	qpeek [-fce –help] jobid	View STDOUT and STDERR of running job

Module Commands		
list	module list	Show current loaded modules
load	module load software[/version]	Load software module and version
Herhangi bir modülü yüklemek için module load centos6.4/app/espresso/5.0.2-impi-mkl		
unload	module unload software[/version]	Unload software module and version
Yüklü modülü kaldırmak içi module unload centos6.4/app/espresso/5.0.2-impi-mkl		
rm	module rm software	Removed loaded software software
swap	module swap old new[/version]	Swap module org for module new/version
avail	module avail [software]	Show all available modules
show	module show software[/version]	Show what module software does

Software Development		
mpicc	mpicc [options] source.c	Compile MPI C source
mpicc	mpiCC -fPIC [options] source.cpp	Compile MPI C++ source
Mpiif90	mpif90 [options] source.f90	Compile MPI F90/F77 source
mpirun	mpirun -np # executable	Run executable on # cpus.
pgcc	pgcc [options] source.c	Recommended C compiler
pgcc	pgCC -fPIC [options] source.cpp	Recommended C++ compiler
Pgf90	pgf90 [options] source.f90	Recommended F90/F77 compiler
time	time expression	Time how long expression takes
make	make [j#][f makefile]	Evaluate makefile in current directory
ddt	ddt executable	Parallel graphical debugger
opt-gui	opt-gui	Parallel Profiler
diff	diff file1 file2	Show changes between two files

Standart kullanıcı hesabı özellikleri Güncel olarak tüm kullanıcı hesaplarının tüm kuyruklara iş gönderme yetkisi vardır. Her bir kullanıcı sistemde yeterli kaynak miktarı olduğu sürece aynı anda en fazla 96 çekirdeğe kadar hesaplama kaynağı kullanabilir. Sistem üzerinde çalışmalara, yoğunluğa ve güncellemelere bağlı olarak bu miktar zaman içerisinde azaltılabilir yada arttırılabilir. Tüm standart kullanıcı hesaplarının ev dizini kotaları 100 GB kadardır. Yine zaman içerisinde TRUBA bu miktarı değiştirme esnekliğine sahiptir. TRUBA tarafından standart kullanıcılara verilen hizmet best-effort niteliğindedir. Bu nedenle kullanıcılara hizmetin devamlılığı, kalitesi, verilerin saklanması ve korunması, ve diğer TRUBA hizmetleri konusunda TRUBA'nın herhangi bir sorumluluğu yoktur, herhangi bir güvence yada garanti sunmaz
İşletim Sistemi Sistemlerde Centos Enterprise Linux 6.4 işletim sistemi kurulu bulunmaktadır.
Dosya Sistemleri Dosya sistemleri hakkında ayrıntılı ve güncel bilgi için kullanıcı el kitabındaki dosya sistemi sayfasına göz atınız.
/truba Bu dosya sisteminin toplam büyüklüğü 115 TB'dır. Dosya sistemi ZFS-LUSTRE tabanlıdır. Bu dosya sisteminde kullanıcı ev dizinleri ile merkezi uygulama/kütüphane dizinleri tutulmaktadır. Kullanıcı arayüz sunucusunda (levrek1) okunur-yazılır (read-write), hesaplama sunucularında ise salt-okunur(read-only) yapılandırmalarla bağlıdır.
SOFTWARE TRUBA Operasyon Merkezi tarafından kurulmuş olan kütüphane, uygulama ve derleyiciler "/truba/sw" dizini altında bulunmaktadır. Her bir kümede yer alan sunucuların işletim sistemine göre software dizinleri ayrı ayrı gruplanmıştır. $TRUBA_SW çevre değişkeni, her küme için o kümenin uygulama(software) dizinini işaret eder. Mesela levrek kümesi için

$TRUBA_SW dizini /truba/sw/centos6.4 dizinini işaret etmektedir.
HOME Kullanıcıların ev dizinleri /truba/home dizini altındadır. Ayrıca $TRUBA_HOME çevre değişkeni her kullanıcı için, ilgili kullanıcının ev dizinine işaret etmektedir ($HOME ile aynı anlamı taşımaktadır). Kullanıcı ev dizinlerinin özellikleri ve her bir kullanıcının sorumlulukları aşağıdaki gibidir: Ev dizinlerinde disk kotası uygulanmaktadır. Her kullanıcı için 40 GB soft 50 GB hard kota mevcuttur. Kullanıcı dizinleri uzun süreli depolama için kullanılmamalı, veriler düzenli olarak kullanıcılar tarafından kendi bilgisayarlarına indirilmelidir. * Kullanıcı dizinlerindeki veri güvenliğinden kullanıcıların kendileri sorumludurlar. Kullanıcı dizinlerinin yedekleri alınmamaktadır.
/truba_scratch /truba_scratch dizinine bağlanmıştır. Bu dosya sistemi SSD diskler ile kaşe imkanları arttırılmış ZTF-LUSTRE dosya sistemi üzerine kurulmuştur. Hesaplama sunucularında ve kullanıcı arayüz sunucusunda okunur-yazılır (read-write) opsiyonu ile bağlıdır. Kullanıcılar işlerini bu performanslı ancak sadece iş çalıştırılırken kullanılabilecek geçici bir alan olan /truba_scratch dizininde çalıştırmalıdırlar. /truba_scratch dizinde herhangi bir kota uygulaması bulunmamaktadır. Son erişim zamanı 45 günden fazla olan dosyalar her gün düzenli olarak dosya sisteminden silinirler. Hem Mercan/Lüfer kümesinde hem de Levrek kümesinde hesaplama performans alanı olarak /truba_scratch dosya sistemi kullanılmaktadır. Yapısı gereği kullanıcı ev dizinlerine göre boyut olarak daha küçük ancak daha hızlı çalışacak şekilde yapılandırılmıştır. Kullanıcıların işlerini bu dosya sisteminde çalıştırması amaçlanmıştır. Hesap sırasında oluşacak dosyaların boyutları tam olarak kestirilemediğinden bu dosya sisteminde kota uygulanmamaktadır. Ancak bu dosya sistemindeki dosyalar düzenli olarak kullanıcı tarafından silinmelidir. Dosya sisteminde, 30 gün boyunca hiç erişilmemiş dosyalar düzenli olarak sistem yöneticisi tarafından silinmektedir. Bu dosya sistemi kesinlikle kalıcı bir dosya depolama alanı olarak düşünülmemelidir. Dosya sistemindeki verilerin güvenliğinden kullanıcının kendisi sorumludur

Güncel Sunucu Aileleri	
Mercan	Mercan kümesi 192 adet HP SL165 sunucularından oluşmaktadır. Her bir sunucu üzerinde 2 adet 12 çekirdekli AMD Opteron 6176 işlemcisi, yani toplamda 24 adet çekirdek bulunmaktadır. Her bir sistem üzerinde 128GB DDR3 ECC bellek bulunur. Sunucular kullanıcı ev dizinlerine ve performans disk alanına üzerlerindeki 1 adet Mellanox Connectx2 QDR 40Gbps Infiniband kartı ile bağlıdırlar. Sunucular üzerinde işletim sistemi olarak RedHat Enterprise Linux türevi olan Scientific Linux 6.2 versiyonu koşturulmaktadır.
Levrek	Levrek sunucuları 128 adet Huawei Tecal RH1288 V2-8S model sunuculardan oluşmaktadır. Her bir sunucu üzerinde 8 çekirdekli Intel E5-2690 serisi işlemci, toplamda 16 çekirdek bulunmaktadır. Sistem üzerindeki bellek miktarı 256GB dır. Sunucular kullanıcı ev dizinlerinin ve performans alanının bulunduğu dosya sistemlerine üzerindeki Mellanox Connectx3 FDR Infiniband kartı ile bağlıdırlar. Sunucular üzerinde Redhat Enterprise 6 türevi olan Centos 6.5 Linux işletim sistemi koşturulmaktadır
Levrekv2	Levrekv2 sunucuları 64 adet Dell R630 model sunuculardan oluşmaktadır. Her bir sunucu üzerinde 8 çekirdekli Intel E5-2680v3 serisi işlemci, toplamda 24 çekirdek bulunmaktadır. Sistem üzerindeki bellek miktarı 256GB dır. Sunucular kullanıcı ev dizinlerinin ve performans alanının bulunduğu dosya sistemlerine üzerindeki Mellanox Connectx3 FDR Infiniband kartı ile bağlıdırlar. Sunucular üzerinde Redhat Enterprise 6 türevi olan Centos 6.5 Linux işletim sistemi koşturulmaktadır. 16 LevrekV2 sunucusunun üzerine 2 şer adet Nvidia M2090 GPU kartları bulunmaktadır.
Orkinos	Sunucu üzerinde 4128 GB bellek, 224 adet Intel Xeon e7-4850 V4 çekirdeği bulunmaktadır. Yüksek bellekli bir smp sunucusudur. Üzerinde Redhat 7.2 işletim sistemi bulunmaktadır.. Ortak dosya sistemine infiniband ağ katmanı ile bağlıdır. Ortak dosya sistemi üzerindeki uygulamaların büyük bir çoğunluğu bu sistem üzerinde çalışabilir durumdadır. Ancak uygulamalardın yüksek verimde çalışması için Intel derleyiciler ve MKL kütüphanesi ile yada GCC derleyicileri ve kütüphaneleri ile derlenirken V3 işlemcilerin vektör komut setlerinin (AVX2) kullanılması için özellikle parametre girilmesi gerekmektedir.