LINUX AND COMPUTATIONAL SERVER

CHUMPOL NGAMPHIW, PH.D.

NATIONAL CENTER FOR GENETICS ENGINEERING AND BIOTECHNOLOGY (BIOTEC)
NATIONAL SCIENCE AND TECHNOLOGY DEVELOPMENT AGENCY (NSTDA)

"GENOME ASSEMBLY AND ANNOTATION" : AUGUST 6-9, 2018 @ KUKPS

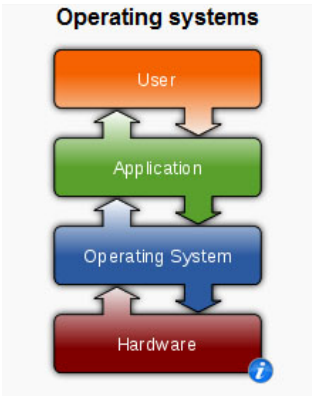
CONTENT

- What is Linux ?
- Why Linux ?
- Why Linux in Bioinformatics ?
- Linux Distribution
- Linux File System
- Access to Linux Server
- Basic Linux Commands
- Basic Shell Programming
- HPC Resources

WHAT IS LINUX ?

The Linux operating system (OS) was first coded by a Finnish computer programmer called **Linus Benedict Torvalds** in 1991, when he was just 21! He had got a new 386, and he found the existing DOS and UNIX too expensive and inadequate.

In those days, a UNIX-like tiny, free OS called **Minix** was extensively used for academic purposes. Since its source code was available, Linus decided to take Minix as a model.



WHY LINUX ?

- Linux is a complete operating system:
  - stable - the crash of an application is much less likely to bring down the OS under Linux.
  - Reliable - Linux servers are often up for hundreds of days compared with the regular reboots required with a Windows system.
  - extremely powerful
- Linux comes with a **complete development environment**, including compilers, toolkits, and scripting languages

WHY LINUX ? (CONT.)

- Linux comes with **networking** facilities, allowing you to share hardware.
- Ideal environment to run servers such as a web server, or an ftp server.
- A wide variety of commercial software is available if not satisfied by the free software
- Easily upgradeable.
- Supports multiple processors.
- True multi-tasking, multi-user OS.
- An excellent window system called X, the equivalent of Windows but much more flexible.
- Full source code is provided and free.

WHY LINUX IN BIOINFORMATICS ?

- One definition of bioinformatics is "the use of computers to analyze biological problems."
- As biological data sets have grown larger and biological problems have become more complex, the requirements for computing power have also grown.
- Computers that can provide this power generally use the Unix/Linux operating system - so you must learn Unix/Linux.
- Linux/UNIX has powerful text processing tools which are highly suited to working with sequence data.
- While many bioinformatics tools have Web interfaces, many more are available via the UNIX/Linux command line.

WHY LINUX IN BIOINFORMATICS ? (CONT.)

- Linux/Unix is very stable - computers running Linux/ Unix almost never crash.
- Linux/Unix is very efficient
  - it gets maximum number crunching power out of your processor (and multiple processors)
  - it can smoothly manage extremely huge amounts of data
- Most new bioinformatics software is created for Unix/Linux - its easy for the programmers.

Linux Distribution

Which Linux Distribution is better ?

- > 300 Linux Distributions
- Slackware (one of the oldest, simple and stable distro.)
- Redhat
  - **RHEL** (commercially support)
  - Fedora (free)
- **CentOS** (free RHEL, based in England)
- **SuSe** (based in German)
- Gentoo (Source code based)
- Debian (one of the few called GNU/Linux)
- **Ubuntu** (based in South Africa)
- Knoppix (first LiveCD distro.)
- **Bio-Linux** (http://environmentalomics.org/bio-linux-download/)
- …

BIO-LINUX

- http://environmentalomics.org/bio-linux-software-list/

Bio-Linux Software List

Software on Bio-Linux 8

The bioinfomatics software on Bio-Linux consists of the packages below, which includes our own packages as well as bioinformatics packages from the main Debian and Ubuntu repositories. This list was last updated in September 2015 and new and updated packages may have been added since then.

Most of these packages work from the command line, so just type the name of the program to run it, or for R packages activate them using the "library(...)" function. To get more info on a package type "dpkg -s *packagename*" or to see the local docs for a package type "go /usr/share/doc/*packagename*". GUI applications can be located through the Installed Applications section of the main Dash menu (the button in the top left corner of the screen).

Package	Version	Description
abyss	1.9.0	de novo, parallel, sequence assembler for short reads
ampliconnoise	1.29	Programs for the removal of noise from 454 sequenced PCR amplicons
arb	5.5	No description
archaeopteryx	0.972.beta9M	A phylogenetic tree viewer and editor
artemis	16.0.0	Free genome viewer and annotation tool
assembly-conversion-tools	0.01	Conversion tools for handling 454 assemblies.
axiome	1.7biolinux1	QIIME and mothur automation toolkit
axiome-dev	1.7biolinux1	QIIME automation toolkit — Plugin Development
backups	0.3	Bio-Linux 5/6/7 backup utilities.
bamtools	2.4.0+dfsg	toolkit for manipulating BAM (genome alignment) files
beam2	0.1+20101008	SNP-SNP interaction association mapping based on SNP-block models
bedtools	2.17.0	suite of utilities for comparing genomic features
berkeley-express	1.5.1	Streaming quantification for high-throughput sequencing
big-blast	1.0.3	The big-blast script for annotation of long sequences
biocode	2.3.3	Biocode plugin for Geneious workbench

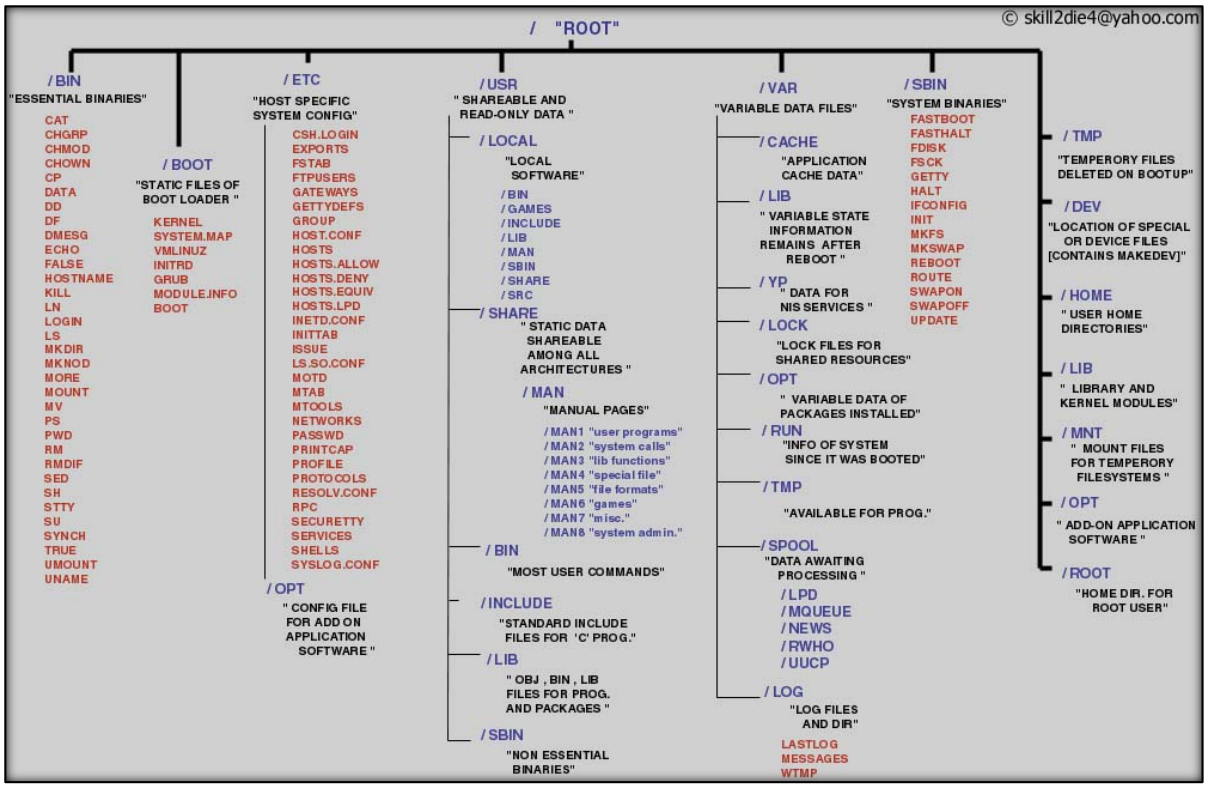
TYPE OF FILE SYSTEM IN LINUX

- File system types can be classified into **disk file** systems, **network file** systems and **parallel file** systems.
- A disk file system is a file system designed for the storage of files on a data storage device, most commonly a disk drive e.g. FAT, NTFS, **ext2**, **ext3, ext4** etc.
- A network file system is a file system that acts as a client for a remote file access protocol, providing access to files on a server e.g. **NFS**, **SMB** etc.
- A parallel file system is a file system that designed to store data across multiple networked servers and to facilitate high-performance access through simultaneous, coordinated input/output operations (IOPS) between clients and storage nodes e.g. **IBM GPFS**, **Lustre**.

LINUX FILE SYSTEM

- Linux has an hierarchical, unified file system
- Supports 256-character filenames.
- All command line entries are **case sensitive**.
- Use the slash(/) rather than the backslash(\) you have been using in DOS.

LINUX FILE SYSTEM (CONT.)

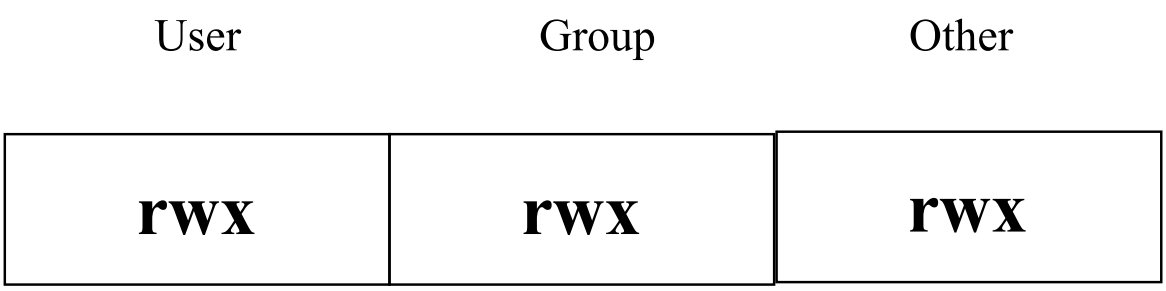


TYPES OF FILE

- Ordinary files
  - text files
  - data files
  - command text files
  - executable files
- directories
- Links
  - rather than having multiple copies of a file, Linux uses linking to one file to save disk space.
- special device files

FILE AND DIRECTORY PERMISSION

- When user create files or directories. System will stamp user owner and permission on the files or directory.
- Linux define 3 sessions



LINUX PERMISSION

- **r** = Read Permission
- **w** = Write Permission
- **x** = Execute Permission

- Change permission by command “chmod number filename”
- number calculate by
  - r = 4 (100), w = 2 (010), x = 1 (001), - = 0

LINUX PERMISSION

Example

- rwx = 7
- rw- = 6
- r-x = 5
- `rwxr-xr-x` = 755
- `rw-rw-rw-` = 666
- `rwxrwx---` = 770
- `drwxr-xr-x = directory`
- `-rwxr-xr-x = file`

ACCESS TO LINUX SERVER

- Use secure shell login (ssh)
- From Linux / MacOSX
  - Use Terminal application
    - $ ssh username@hostname
- From Windows OS
  - Putty (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)
  - SSHSecureShellClient (http://www4a.biotec.or.th/hpc/wp-content/uploads/2009/10/SSHSecureShellClient-3.2.9.exe)

BASIC COMMANDS

- **ls**
  - $ ls -l
  - $ ls -a
  - $ ls -la
  - $ ls -lt
  - $ ls -lS
- **cd**
  - $ cd /usr/bin
- **pwd**
  - $ pwd
- **~**
  - $ cd ~
- ***~user***
  - $ cd ~user1
  - What will “cd ~/user1” do ?
- **which**
  - $ which ls
- **whereis**
  - $ whereis ls
- **locate**
  - $ locate stdio.h
  - $ locate iostream
- **rpm**
  - $ rpm -q bash
  - $ rpm -qa
  - $ rpm -qa | sort | less
- **find**
  - $ find / | grep stdio.h
  - $ find /usr/include | grep stdio.h

BASIC COMMANDS (CONT)

- **echo**
  - $ echo “Hello World”
  - $ echo -n “Hello World”
- **cat**
  - $ cat /etc/motd
  - $ cat /proc/cpuinfo
- **cp**
  - $ cp foo bar
  - $ cp -a foo bar
- **mv**
  - $ mv foo bar
- **mkdir**
  - $ mkdir test
  - $ mkdir -p test/test1
- **rm**
  - $ rm foo
  - $ rm -rf foo
  - $ rm -i foo
  - $ rm -- -foo
- **finger**
  - $ finger
- **chgrp**
  - $ chgrp bar /home/foo
- **chsh**
  - $ chsh foo
- **chfn**
  - $ chfn foo
- **chown**
  - $ chown -R foo:bar /home/foo

BASIC COMMANDS (CONT)

- **man**
  - $ man ls
- **tar**
  - $ tar cvfp lab1.tar lab1
- **gzip**
  - $ gzip -9 lab1.tar
- **untar & ungzip**
  - $ gzip -cd lab1.tar.gz | tar xvf –
  - $ tar **x**vf**z** lab1.tar.gz
- **tar & bzip2**
  - $ tar **c**vf**j** lab1
- **touch**
  - $ touch foo
  - $ cat /dev/null > foo
- **Pipe**
  - $ cal > foo
  - $ cat /dev/zero > foo
  - $ cat < /etc/passwd
  - $ echo 'Test append' >> test1.txt
  - $ who | cut -d' ' -f1 | sort | uniq | wc -l
- **backtick**
  - $ echo "The date is `date`"
  - $ echo `seq 1 10`
- **Hard, soft (symbolic) link**
  - ln vmlinuz-2.6.24.4 vmlinuz
  - ln -s firefox-2.0.0.3 firefox

BASIC COMMANDS (CONT)

- **Disk usage**
  - $ df -h /
- **File space usage**
  - $ du -sxh ~/
- **Display Linux processes**
  - $ top
- **Clear screen display**
  - $ clear
- **Create multiple terminals**
  - $ screen
  - https://www.tecmint.com/screen-command-examples-to-manage-linux-terminals/

BASIC COMMANDS (CONT)

- **Change file or directory permission**
  - **Example**
    - $ mkdir public_html
    - $ chmod 755 public_html
    - $ ls -l

- **Change permission can use relative permission for change by 'u', 'g' or 'o'**
  - u = owner, g = group, o = other
  - + = add permission, - = remove permission
  - **Example**
    - rw-r--r-- change to rw-rw-r--
      - $ chmod g+w test
    - rw-r--r-- change to rwxrwxr-x
      - $ chmod ug+wx, o+x test
    - rwxrwxr-x change to rwxr--r--
      - $ chmod go-wx test

BASIC COMMANDS (CONT)

- **Change owner of files or directory**
- $ chown [username].[groupname] [option] files
- **option**
  - -R, Change owner in subdirectory
  - -f, Ignore error

  - $ chown user test.txt
  - $ chown test.test homework.c
  - $ chown test:test test.txt
  - $ chown user1.group1 -R /home/

BASIC COMMANDS (CONT)

- **View file contents**
  - $ more test1.txt
  - $ head test1.txt
  - $ tail **-f** test1.txt # output the last part of files; **output appended data as the file grows**
  - $ head -**n** test1.txt # output the first **n** lines of files
  - $ tail -**n** test1.txt # output the last **n** lines of files

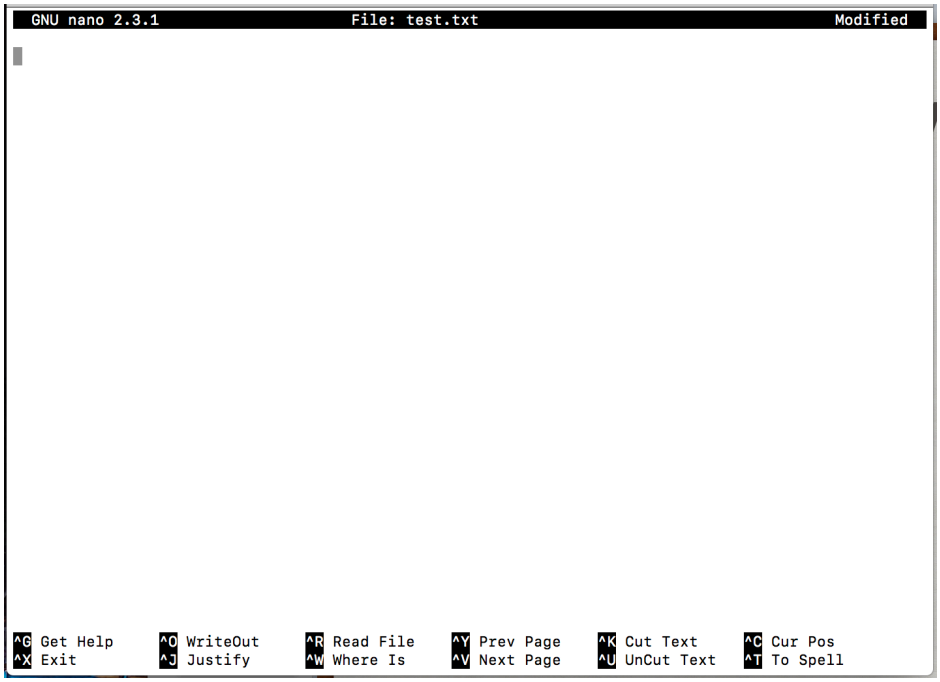
- **Print newline, word and byte counts for each file**
  - $ wc test1.txt

- **File pattern searcher**
  - $ grep 'test' file1 # match search
  - $ grep -v 'test' file1 # not matching search
  - $ grep -E 'test1|test2' file1 # contain test1 or test2 in a file

- **Cut out selected portions of each line from each file**
  - $ cut -d ' ' -f 1 test1.txt
  - $ cut -d$'\t' -f 2 test1.txt
    - -d define the field delimiter; use **$'\t'** for tab-delimited
    - -f define the specifies fields; we can specifies multiple fields such as -f 1,2,3

TEXT EDITOR WITH NANO/PICO

- $nano test1.txt



VI/VIM EDITOR

- **2 modes**
  - Input mode
  - Command mode
  - ESC to back to cmd mode
- **Cursor movement**
  - h (left), j (down), k (up), l (right)
  - ^f (page down)
  - ^b (page up)
  - ^ (first char.)
  - $ (last char.)
  - G (bottom page)
  - :1 (goto first line)
- **Switch to input mode**
  - a (append)
  - i (insert)
  - o (insert line after)
  - O (insert line before)
- **Delete**
  - dd (delete a line)
  - d10d (delete 10 lines)
  - d$ (delete till end of line)
  - dG (delete till end of file)
  - x (current char.)
- **Paste**
  - p (paste after)
  - P (paste before)
- **Undo**
  - u
- **Search**
  - /text
- **Save/Quit**
  - :w (write)
  - :q (quit)
  - :wq (write and quit)
  - :q! (give up changes)

BASIC SHELL COMMANDS

- **TCSH**
- **BASH** (Bourne Again Shell) http://www.tldp.org/LDP/Bash-Beginners-Guide/html/Bash-Beginners-Guide.html
- **For loop**

```
[user@agcipher ~]$ for i in `cat '/etc/passwd'`; do name=`echo $i | cut -d ':' -f 1`; echo $name; done
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
operator
games
ftp
User
nobody
systemd-bus-proxy
Bus
Proxy
systemd-network
Network
Management
dbus
message
bus
polkitd
for
polkitd
abrt
unbound
```

BASH SHELL ENVIRONMENTS

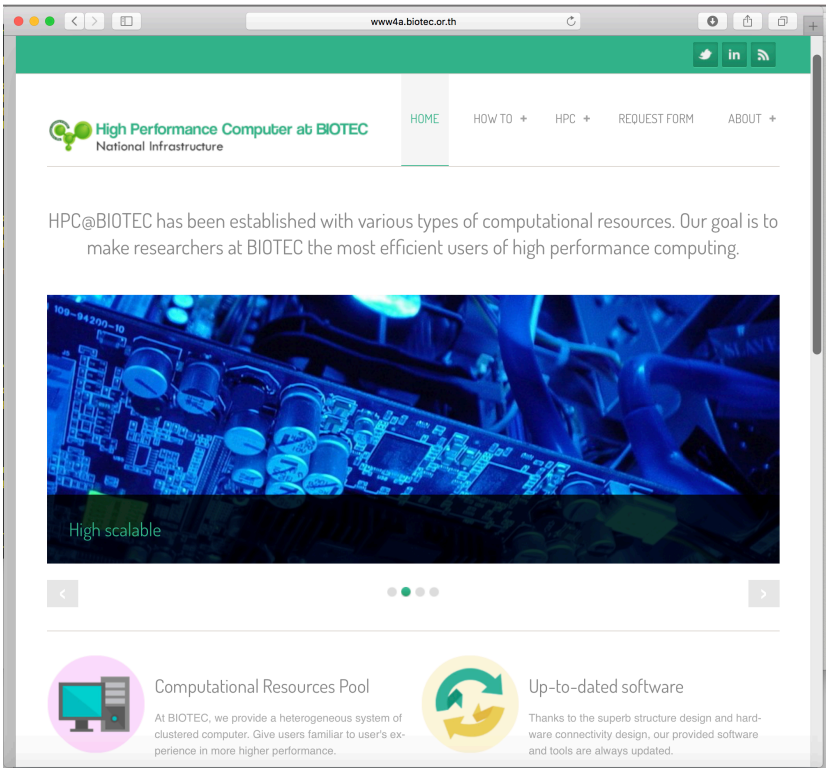
- **.bash_profile vs .bashrc**
  .bash_profile is executed for login shells, while .**bashrc** is executed for interactive non-login shells. When you login (type username and password) via console, either sitting at the machine, or remotely via ssh: .**bash_profile** is executed to configure your shell before the initial command prompt

- **export PATH**
- **export LD_LIBRARY_PATH**
- **Set shell prompt**
  - $ export PS1="[\u@\h \W]\\$"
- **Scripts arguments**
  - $ ./test.sh arg1 arg2 arg3

```
# test.sh
#!/bin/env bash
./a.out $1 $2 $3
```


HPC RESOURCES

HTTP://WWW4A.BIOTEC.OR.TH/HPC

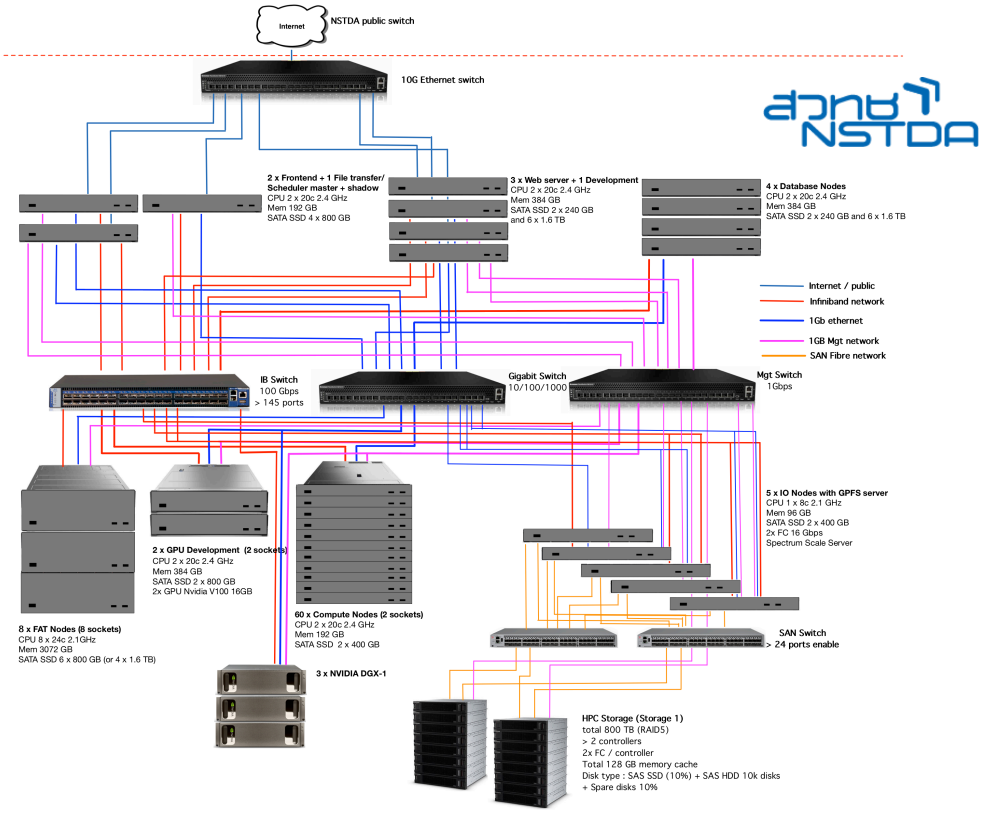


COLOSSUS CLUSTER

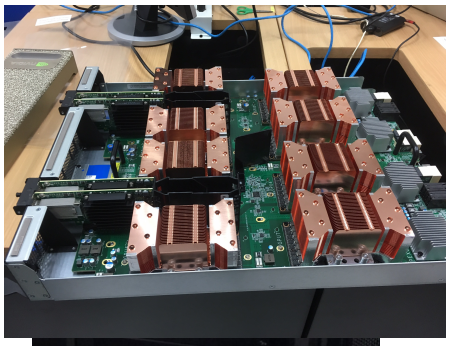
http://colossus.biotec.or.th/ganglia1

- **Storage 110 TB**
- **14 compute nodes**
- **4 x 8c Intel Xeon 2.2 GHz**
  - Memory
    - 1 x 768 GB
    - 8x 512 GB
    - 1 x 496 GB
    - 1 x 324 GB
    - 3 x 128 GB
- **SGE Job scheduler**

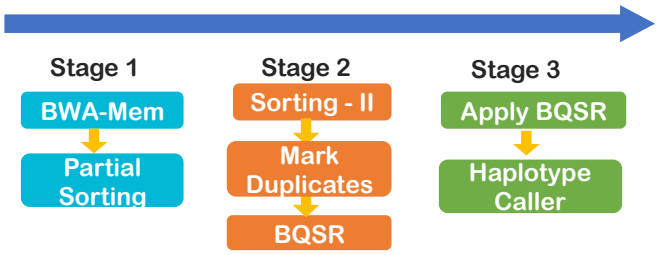
NSTDA HPC INFRASTRUCTURE

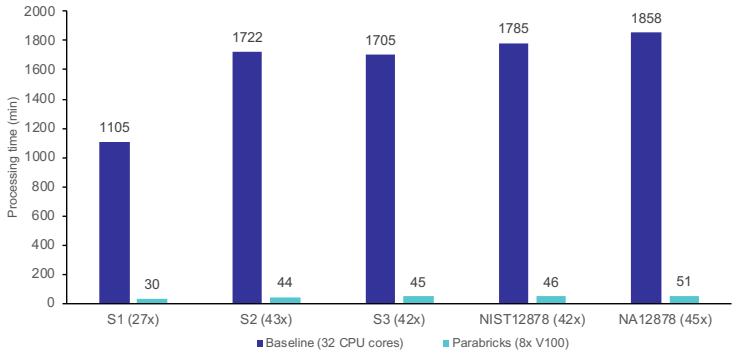


NVIDIA DGX-1



Parabricks





Q & A