Introduction to Cluster Computing:

Linux, shell scripting, queuing systems, cluster architecture

Instructor: Dr. Peggy Lindner (plindner@uh.edu)

DASh

UNIVERSITY of HOUSTON
CENTER FOR ADVANCED COMPUTING & DATA SYSTEMS

Objectives

- comfortably use basic UNIX/Linux commands from the command line (from a terminal window)
- organize and manage their files within the UNIX/Linux file system
- organize and manage their processes within UNIX/Linux
- usefully combine UNIX/Linux tools using features such as filters, pipes, redirection, and regular expressions
- customize their UNIX/Linux working environment
- be knowledgeable enough about basic UNIX/Linux shell scripting to be able to successfully read and write bash shell scripts;
- know how to use UNIX/Linux resources to find additional information about UNIX/Linux commands.
- Hands-On

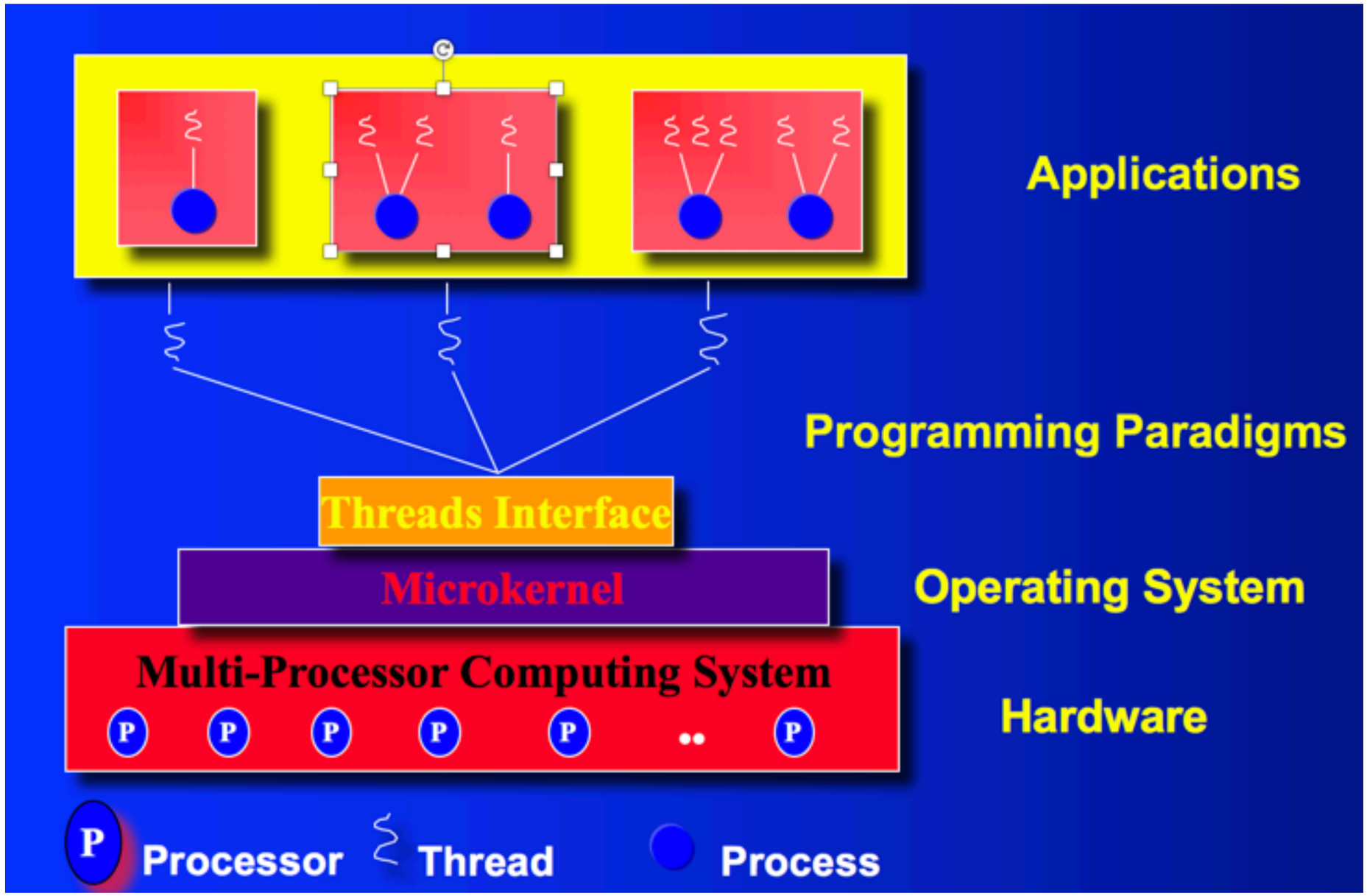
Course Organization

- We will meet 5 times from 4-7pm (5 x 3 = 15 contact hours)
- Location: MREB 200
- We will utilize the Moodle framework for course material, home work, grades etc. https://www.cacds.uh.edu/training/course/view.php?id=75

Course Resources

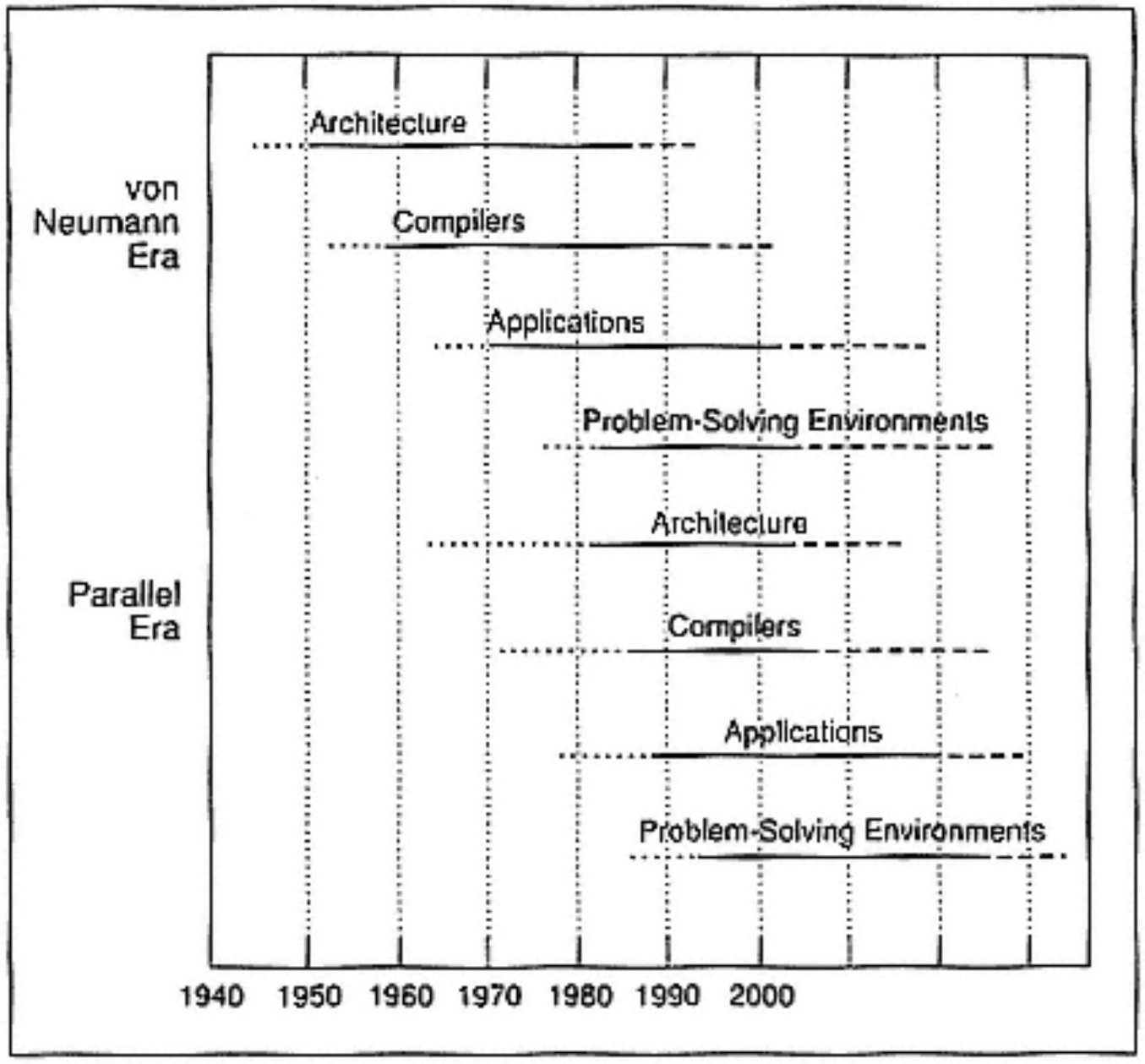
- PCs in the classroom: use cougarnet account for login
- You will receive another login information for the training cluster (which you can also access from home)
- Store your exercises & slides etc. in the H drive on the PC in MREB 200
- To practice at home you may need to install some software:
  - Windows: http://mobaxterm.mobatek.net/
  - Mac: terminal

Cluster Computing - Motivation



Computing Elements

Cluster Computing - Motivation



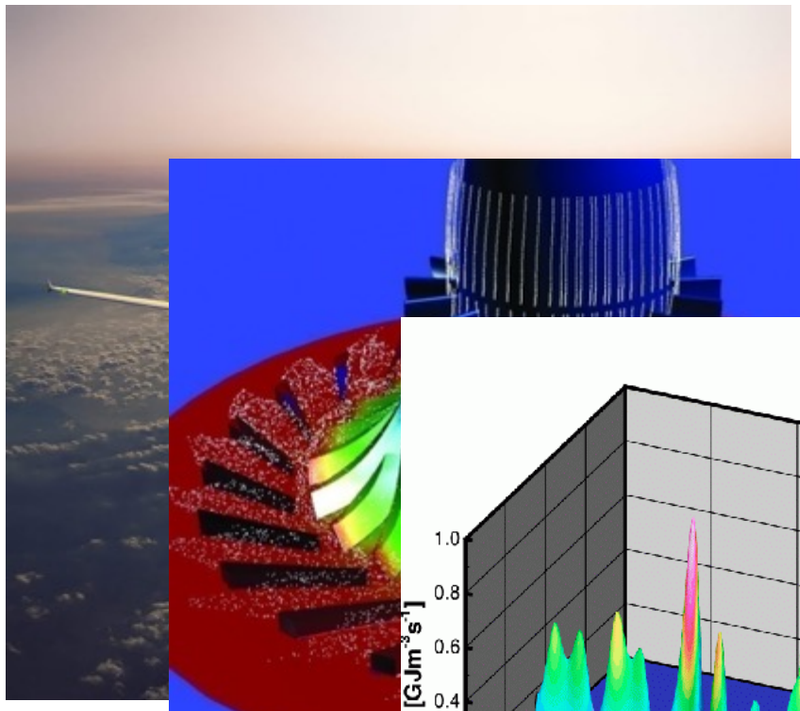
Two Eras of Computing

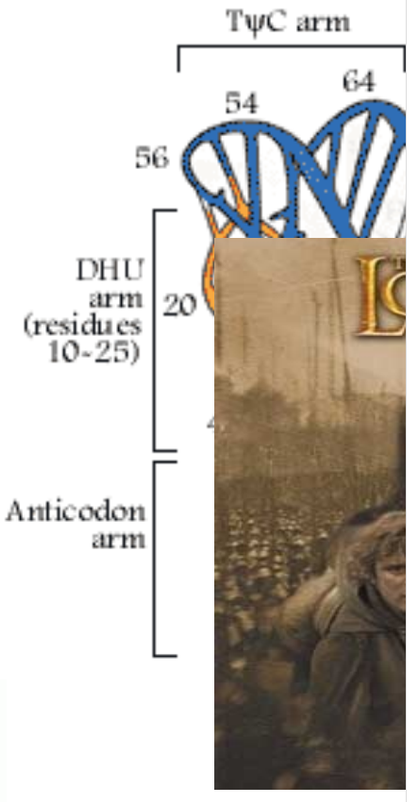
Image courtesy of:
http://publishing.cdlib.org/ucpressebooks/view?docId=ft0f59n73z;chunk.id=0;doc.view=print

Cluster Computing - Motivation

- What drives High Performance Computing (HPC)?

  Solving grand challenge applications using computer modeling, simulation and analysis





TOP500 Sublist for November 2004

R_{max} and R_{peak} values are in GFlops. For more details about other fields, please click on the button "Explanation of the Fields"

EXPLANATION OF THE FIELDS

13 Entries found

Rank	Site Country/Year	Computer / Processors Manufacturer	R_{max} R_{peak}
46	Credit Suisse/First Boston United Kingdom/2004	**BladeCenter HS20 Xeon 3.06 GHz, Gig-Ethernet** / 1500 IBM	3755 9180
50	Bank (H) Germany/2004	**BladeCenter Xeon 3.06 GHz, Gig-Ethernet** / 1064 IBM	3755 6511.68
120	UBS Warburg United Kingdom/2004	**BladeCenter Xeon 2.8 GHz, Gig-Ethernet** / 650 IBM	2026 3640
133	UBS Warburg United Kingdom/2004	**BladeCenter HS20 Xeon 3.06 GHz, Gig-Ethernet** / 512 IBM	1922.56 3133.44
166	SG SGBI France/2003	**xSeries Cluster Xeon 2.4 GHz - Gig-E** / 968 IBM	1685.49 4646.4
239	Financial Services (H) United States/2004	**BladeCenter Xeon 3.06 GHz, Gig-Ethernet** / 336 IBM	1261.68 2056.32
250	Societe Generale France/2004	**DL360G3, Pentium4 Xeon 3.2 GHz, Myrinet** / 320 HP	1228 2048
259	Banco Azteca Mexico/2004	**Integrity Superdome, 1.5 GHz, HPlex** / 352 HP	1210 2112
320	UBS Warburg United States/2004	**xSeries Xeon 3.06 GHz - Gig-E** / 300 IBM	1126.5 1836
382	CIE Gegetel SI France/2004	**SuperDome 875 MHz/HyperPlex** / 512 HP	1013 1792
391	Financial Services Company (E) United States/2003	**BladeCenter Cluster Xeon 2.4 GHz, Gig-Ethernet** / 440 IBM	1009.17 2112
401	Financial Services Company (G) United States/2004	**eServer Opteron 2.0 GHZ, Gig-E** / 900 IBM	987.1 3600
403	Unicredito Italy/2004	**eServer Opteron 2.2 GHZ, Gig-E** / 800 IBM	987.1 3520

Cluster Computing - Motivation

- There are 3 ways to improve performance:

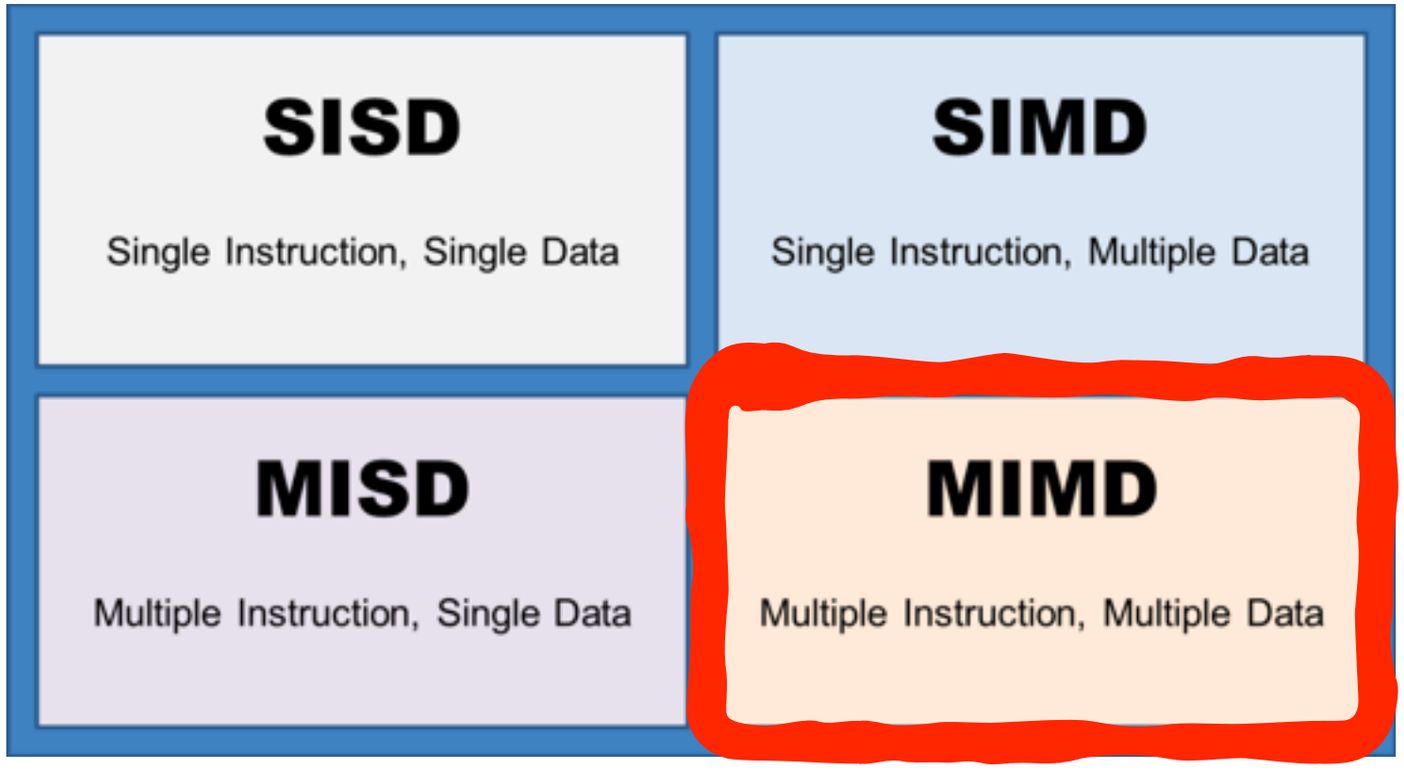
  1. Work Harder
  2. Work Smarter
  3. Get Help

- Computer Analogy

  1. Use faster hardware: e.g. reduce the time per instruction (clock cycle).
  2. Optimized algorithms and techniques
  3. Multiple computers to solve problem: That is, increase no. of instructions executed per clock cycle.

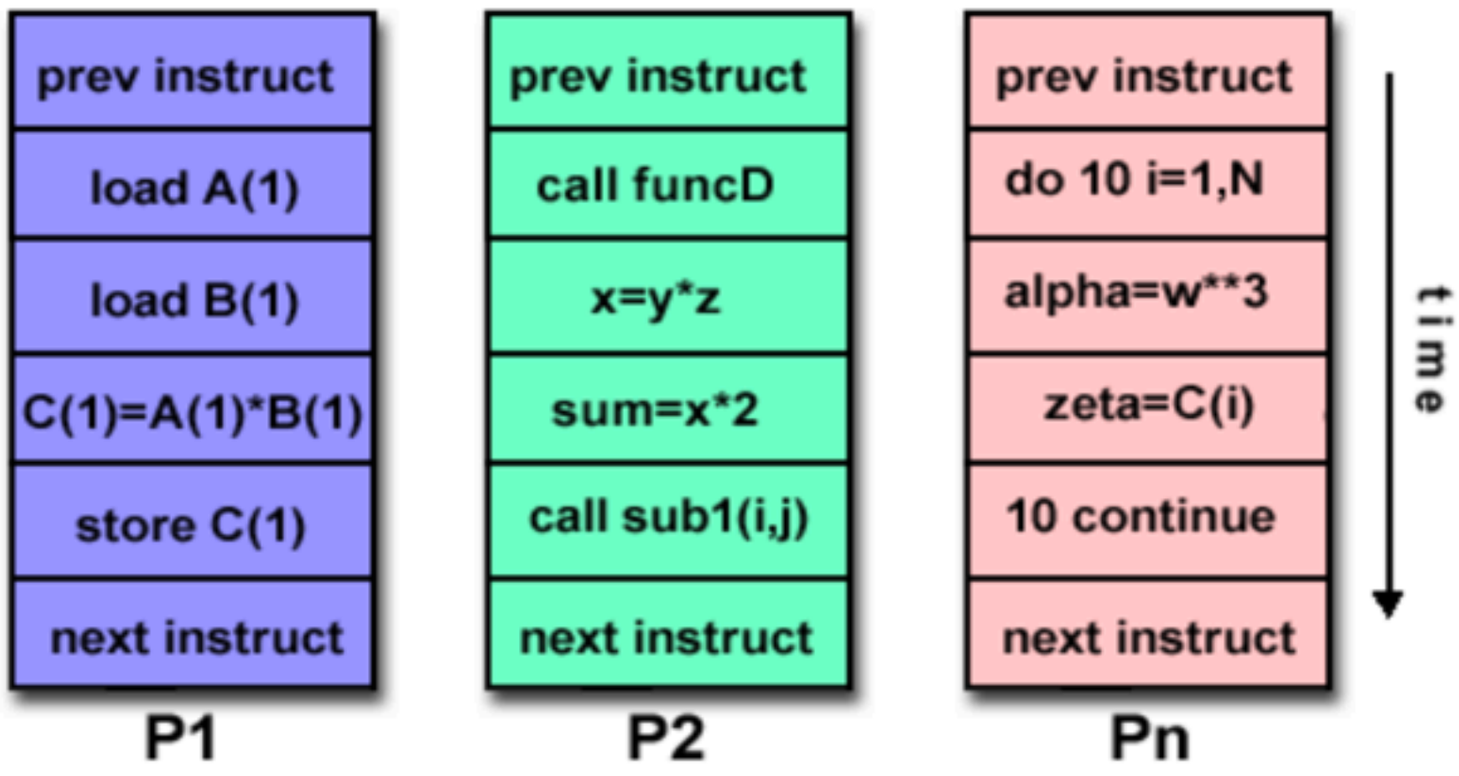
Cluster Computing - Motivation

- There are different ways to classify parallel computers
- One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy.
    - Based on the number of concurrent instructions and data streams available in the architecture

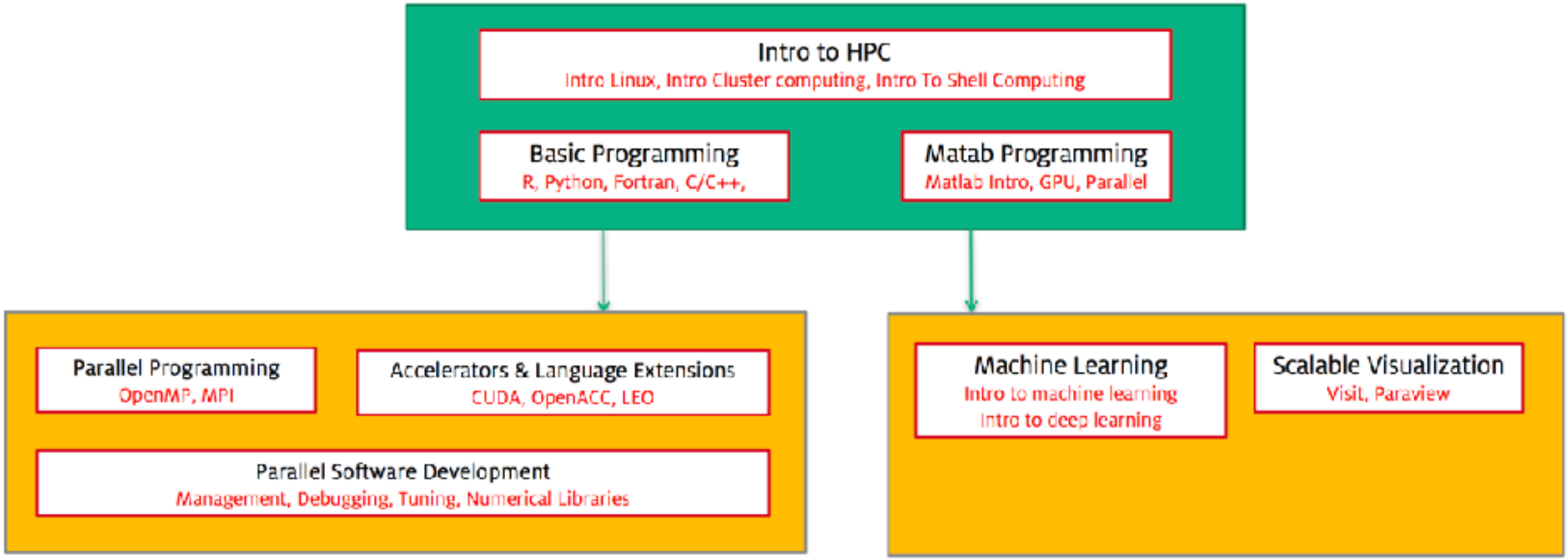


Cluster Computing - Motivation

- Multiple Instruction, Mingle Data (MIMD):
  - Multiple autonomous processors simultaneously executing different instructions on different data
  - Multiple Instruction: Every processor may be executing a different instruction stream
    - Multiple Data: Every processor may be working with a different data stream



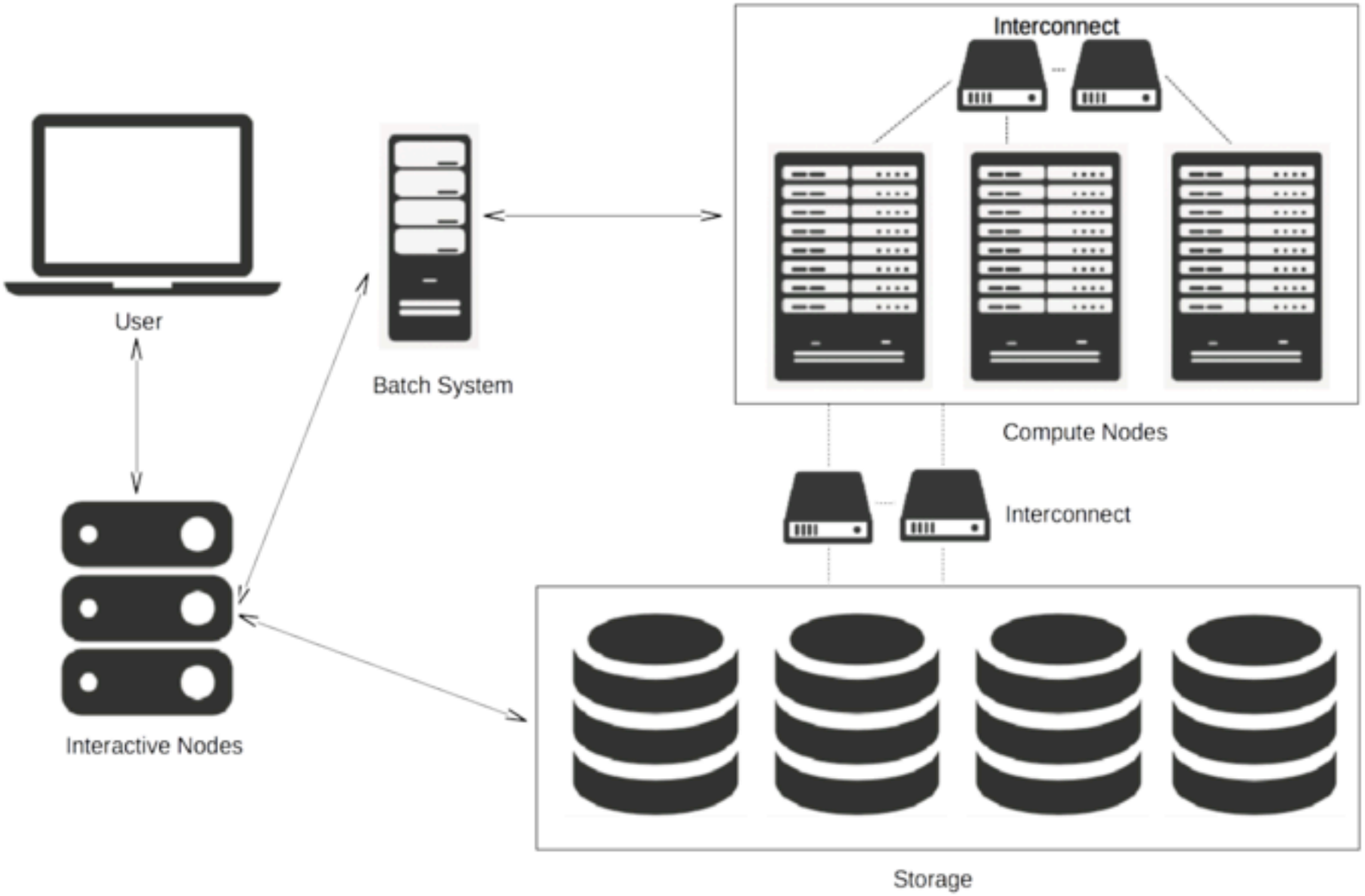
Further Courses/Reading



- Motivation for parallel computing:

  http://compsci.hunter.cuny.edu/~sweiss/course_materials/csci493.65/lecture_notes_2014/chapter01.pdf

What is computing Cluster?



Cluster Components (I)

- Compute nodes mostly based on regular PC technology
  - Intel or AMD processors
  - 1-4GB of main memory per core
- Operating Systems: typically Linux/UNIX, some Windows clusters also available
- Managements of resources: cluster scheduler
  - Manages allocation of compute nodes to users

Cluster Components (II)

- Networking metrics:
  - Latency: minimal time to send a very short message from one communication endpoint to an other endpoint
    - Unit: ms, µs
  - Bandwidth: amount of data which can be transferred from one processor to another in a certain time frame
    - Unit: Bytes/sec, …, GB/s; Bits/sec,…, Gb/s

- Of-the-shelf technology vs. High-End Technology
  - Gigabit-Ethernet and 10GEthernet vs. InfiniBand, Omnipath, Cray Gemini
  - Most clusters contain a high-end and a low-end network interconnect

- Network Topology of importance for large clusters

Some common terms

- **CPU Time:** the amount of time a CPU is in use, measured per CPU (so a 4 processor job that runs for 15 minutes takes 1 hour of cpu time)

- **Job:** any user submitted program executed on the cluster

- **Job File:** a file containing information about a job used by the resource manager and the scheduler to schedule the job

- **Queue:** an ordered group of jobs waiting to run

- **Wall-clock time:** the amount of time a job runs on the cluster (a 4 processor job that runs for 15 minutes takes 15 minutes of wall-clock time)

Computing Cluster - some examples

Blade servers

- Networking and power management integrated in the chassis
- Allows for higher density

CACDS Opuntia Cluster

- Both Shared and Distributed systems
  - single core, multicore (multithreaded) and MPI jobs
- 80+ compute nodes (1700 + cpu cores)
  - http://baragon.hpcc.uh.edu/ganglia/?c=Opuntia
  - 2 Xeon Phi 5110P
  - 4 Nvidia Tesla K40 GPUs
- Operating system: Linux
  - Distribution: Redhat 6.7
- 56 Gigabit network interconnect

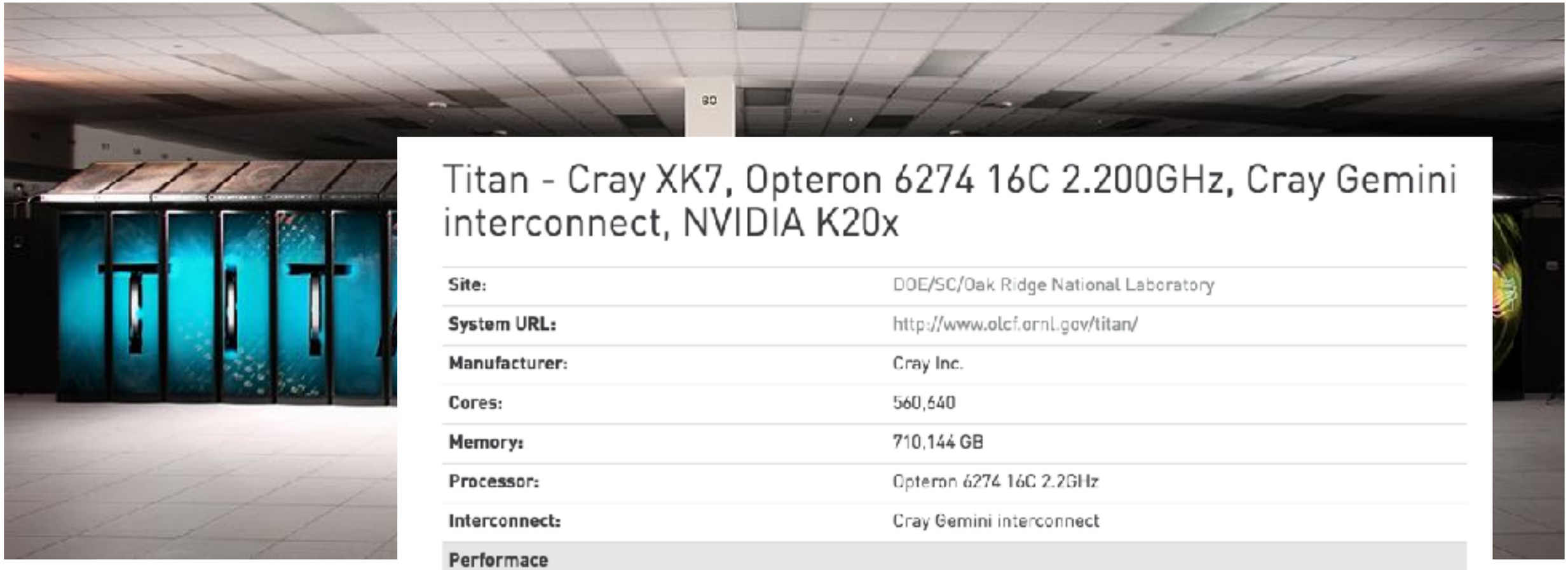
CACDS Cluster Resources

- Opuntia & Maxwell clusters are for research work under a UH PI
  - Faculty sponsored RESEARCH activity only
  - Open to all UH Central faculty for research purposes
  - Students & postdocs –under UH faculty guidance
  - Collaborators –under UH faculty guidance

- You will be using a surrogate cluster for this training (not Opuntia nor Maxwell)

- If you start working with UH faculty, you may request an account after authorization from your supervisor https://www.cacds.uh.edu/opuntia-account-request

Top 500 Example



Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x

Site:	DOE/SC/Oak Ridge National Laboratory
System URL:	http://www.olcf.ornl.gov/titan/
Manufacturer:	Cray Inc.
Cores:	560,640
Memory:	710,144 GB
Processor:	Opteron 6274 16C 2.2GHz
Interconnect:	Cray Gemini interconnect
Performace	
Linpack Performance (Rmax)	17,590 TFlop/s
Theoretical Peak (Rpeak)	27,112.5 TFlop/s
HPCG [TFlop/s]	322.3
Power Consumption	
Power:	8,209.00 kW (Submitted)
Software	
Operating System:	Cray Linux Environment

Shell

- is a user program or it's environment provided for user interaction

  - A program that takes keyboard commands and passes them to the operating system to carry out

- is an command language interpreter that executes commands read from the standard input device (keyboard) or from a file

- is not part of system kernel, but uses the system kernel to execute programs, create files etc.

Shell

- Different shell types exist (bash,
- Default shell shipped with most Linux Systems is the BASH shell
- Other shell types could also be installed

The “terminal” emulator

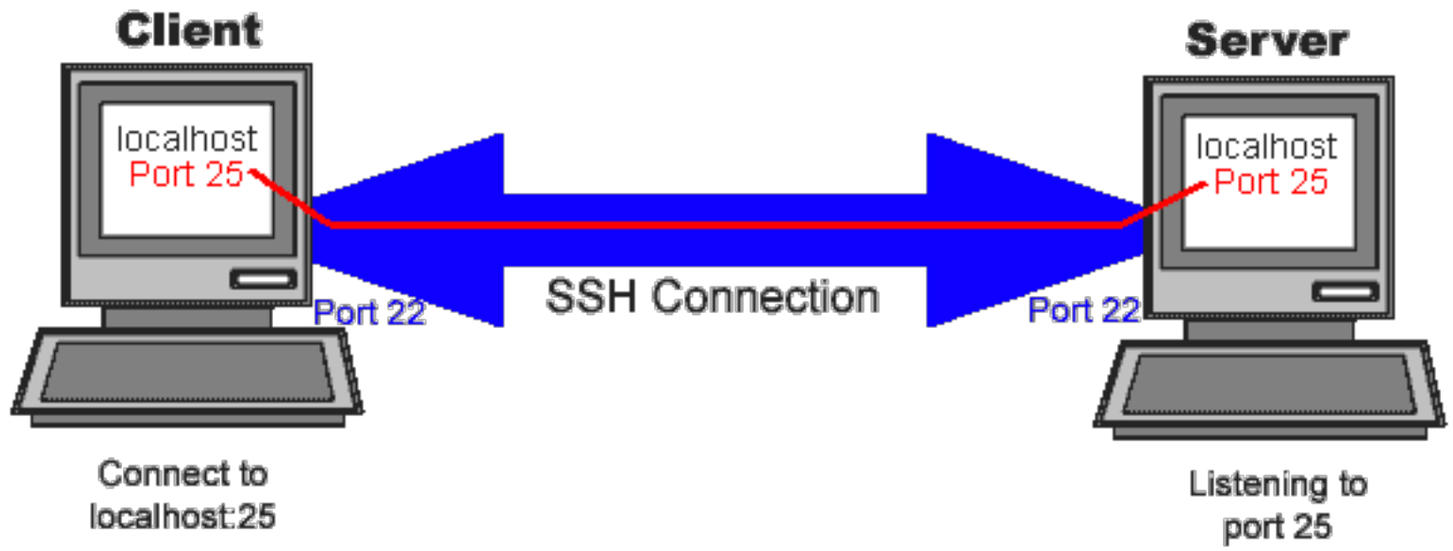
- Commonly used to refer to the shell
- Graphical User Interface (GUI) program that allows users to interact with the shell
  - Examples: konsole, gnome-terminal, Lxterminal, xterm
  - Terminal emulator give us access to the shell

The bash shell

- Enhanced version of SH shell (SH was written by Steve Bourne)
- BASH also known as Bourne Again SHell
- The BASH shell (command line interpreter) is an open-source version of the original UNIX Bourne shell “sh”.
- Usually the default shell in a Linux environment
- Similar to Explorer in Windows, or Finder in Mac OSX
- Uses specific syntax (like $ to indicate variable names)

Access remote system- via ssh

SSH - Secure Shell



- Log into your accounts
  - Username or login = cacdsX
  - Where x = sign in serial number 01 … 25
  - Password = cacds-X

```
> ssh username@whale.cs.uh.edu
```

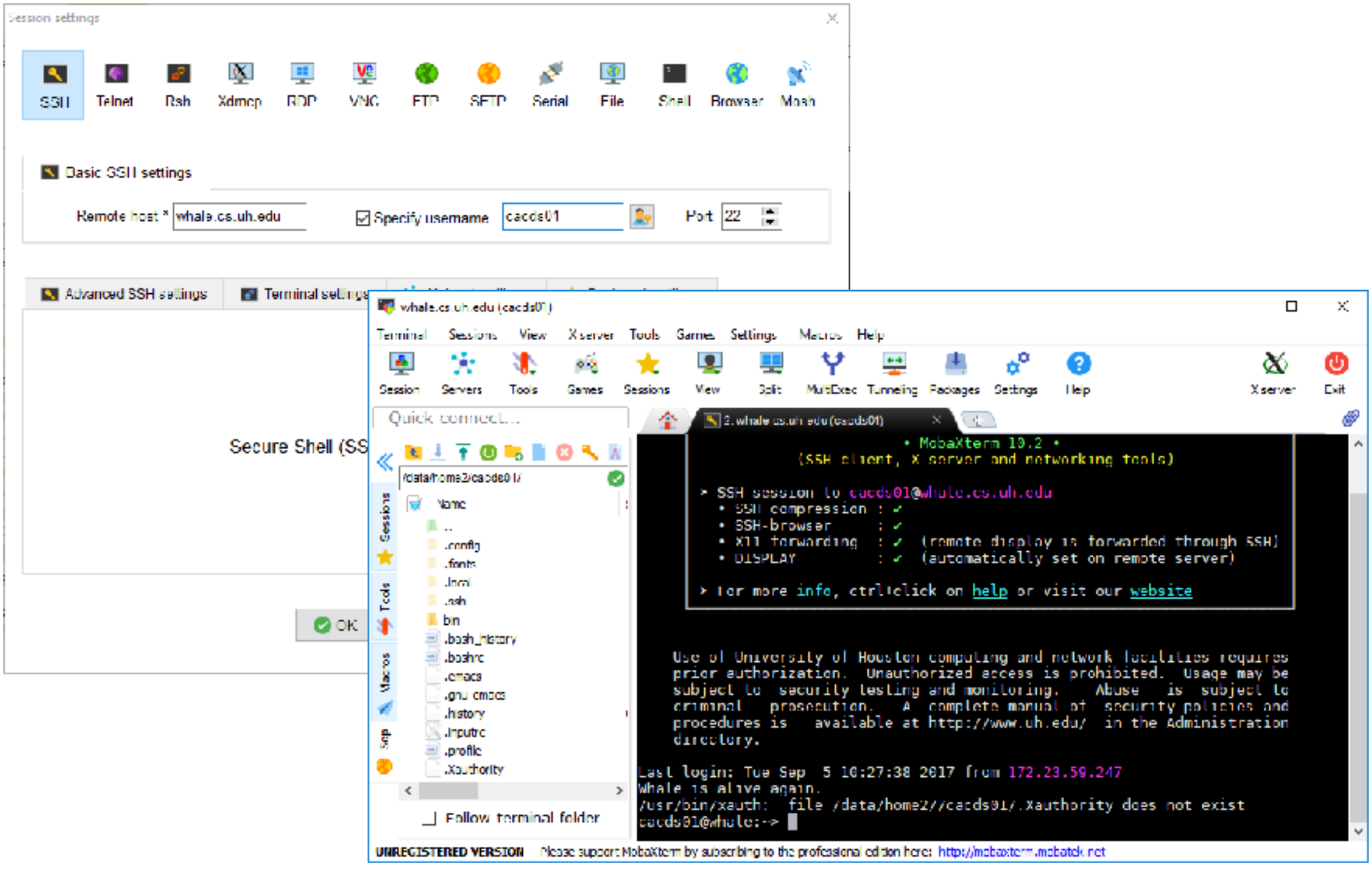
Connecting via SSH Client on Mac based Systems

```
[plindner@max:~$
[plindner@max:~$ ssh cacds01@whale.cs.uh.edu

    Use of University of Houston computing and network facilities requires
    prior authorization.  Unauthorized access is prohibited.  Usage may be
    subject to  security testing and monitoring.    Abuse   is  subject to
    criminal   prosecution.   A  complete manual of  security policies and
    procedures is   available at http://www.uh.edu/  in the Administration
    directory.

[Password:
[Password:
Last failed login: Tue Sep  5 10:27:34 CDT 2017 from 172.23.59.247 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Tue Sep  5 09:28:59 2017 from 172.23.59.247
Whale is alive again.
cacds01@whale:~>
```

Connecting via MobaXterm on Windows Systems



The bash shell prompt

```
cacds01@whale:~>
```

- is called a shell prompt
- in bash it's usually "$" (dollar symbol)
- appears whenever the shell is ready to accept input

Interacting with the shell

```
> date

Tue Sep  5 11:03:13 CDT 2017

> cal

September 2017

Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Interacting with the shell

- Find out which shell you are using?

```
> ps -p $$

PID TTY          TIME CMD

14861 pts/1    00:00:00 bash
```

- Need to use a different shell?

  `> /bin/csh` (will switch you to CSH shell)

  `whale` `/cacds01%`

- type `exit` or press *CTRL-D* to return to your previous shell

Files and directories

- List current directory `ls` and `ls -a`
- List path for current directory `pwd`
- Create directory `mkdir` e.g. `mkdir test`

- Absolute Path:

  `/data/home2//cacds01`

- Relative Path `test`
- We can use `..` to go one directory up
- `cd` without anything will always get back to our “home directory”

Wildcards and redirection

- wild cards in filenames
  * matches zero or more characters

- ? matches exactly 1 character redirection

- > redirects std-out to a file

- >> appends std-out to a file

- < redirects std-in from a file

How to edit a file

- Multiple options for command line editors (no GUI needed), most systems provide *emacs* and/or *vim*

- *v*im is easier to learn, emacs is way more powerful

- Learn one of the 2

- We will use vim for this course

```
> vim myfirstfile
```

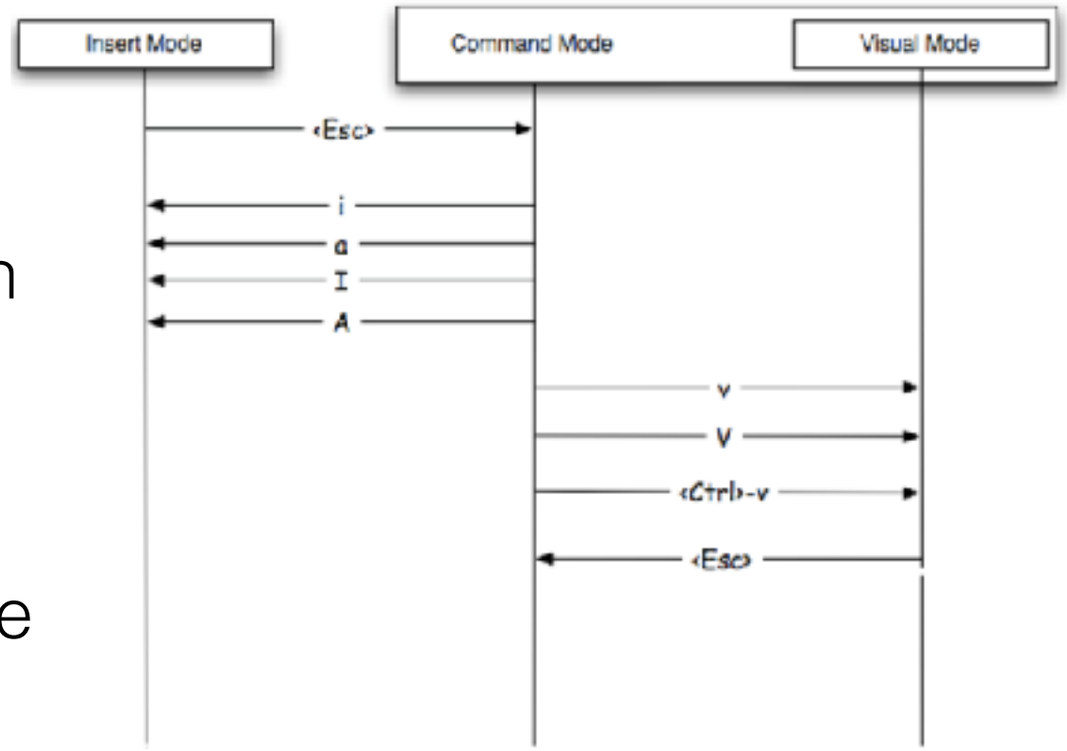
Vim editor - Modes

- 3 Modes:

1. Command mode: all keystrokes are interpreted as commands

2. Insert mode: most keystrokes are inserted as text (leaving out those with modifier keys)

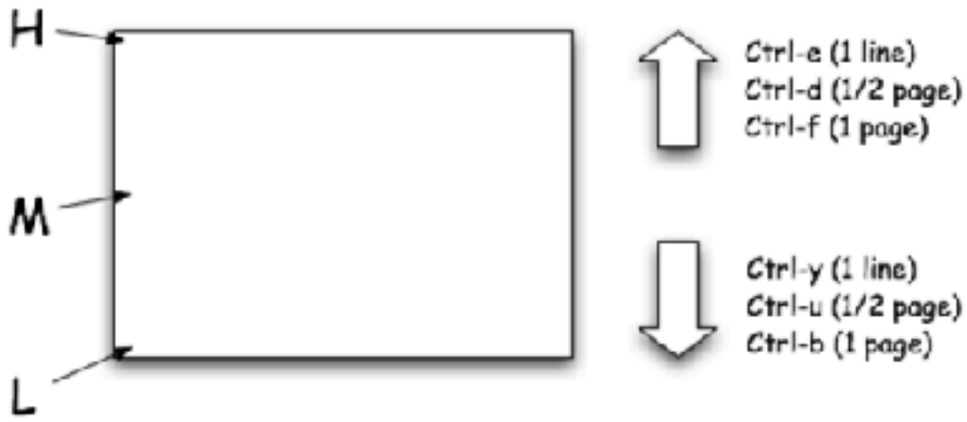
3. Visual mode: helps to visually select some text, may be seen as a submode of
the the command mode.



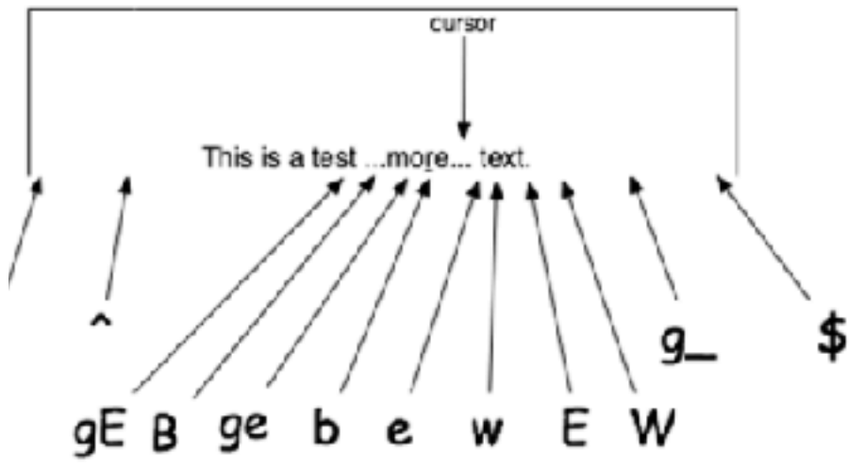
- To switch from the insert or visual mode to the command mode, type <Esc>.

Vim editor - Movement

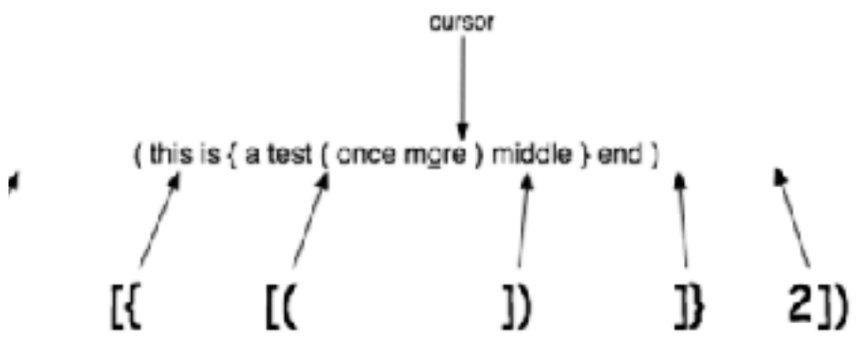
- h …move left
- l …move right
- k …move up
- j …move down



- `0` ...first column of the line
- `^` ...first non-blank character of the line
- `w` ...jump to next word
- `W` ...jump to next word, ignore punctuation
- `e` ...jump to word-end
- `E` ...jump to word-end, ignore punctuation
- `b` ...jump to word-beginning
- `B` ...jump to word-beginning, ignore punctuation
- `ge` ...jump to previous word-ending
- `gE` ...jump to previous word-ending, ignore punctuation
- `g_` ...jump to last non-blank character of the line
- `$` ...jump to the last character of the line



g0, g^ and g$ work for screen-lines (if the line is longer than the screen).



These commands work over many lines! Use % to jump to the matching bracket.

Vim editor - Editing

- Inserting text is pretty simple in Vim, just type i and start typing.
  - d …delete the characters from the cursor position up the position given by the next command (for example d$ deletes all character from the current cursor position up to the last column of the line).
  - c …change the character from the cursor position up to the position indicated by the next command.
  - x …delete the character under the cursor.
  - X …delete the character before the cursor (Backspace).
  - y …copy the characters from the current cursor position up to the position indicated by the next command.
  - p …paste previous deleted or yanked (copied) text after the current cursor position.
  - P …paste previous deleted or yanked (copied) text before the current cursor position.
  - r …replace the current character with the newly typed one.
  - s …substitute the text from the current cursor position up to the position given by the next command with the newly typed one.
  - . …repeat the last insertion or editing command (x,d,p…).
- Doubling d, c or y operates on the whole line, for example `yy` copies the whole line.

Vim editor - More

- Don’t be afraid to try the various commands, you can undo almost anything using u in the command mode – even undo is undoable using Ctrl-r.

- Save your work by using `w` in the command mode

Basic UNIX™ Commands

- `cat` - concatenate files
- `cp` - copy a file
- `grep` - scan for a string
- `head` - show first lines of a file
- `tail` - show last lines of a file
- `mv` - move or rename a file
- `rm` -f remove files (silently)
- `wc` - count lines, words, characters (wc output format varies between systems)

Variables

- A variable is a shell parameter that is associated with a value (sometimes referred to as shell variables)

- Syntax for assigning variable
  VARIABLE_NAME=value

- Accessed by adding “$” prefix to VARIABLE name
  $VARIABLE_NAME

- User-created variables ($AGE, $MYSTUFF)

- Keyword variables (usually inherited or initialized by the shell - $HOME, $USER)

Setting and unsetting Variables

- *export* - built-in command is used to set variable to a value

```
> export DATE=Monday

> echo “Right now it’s $DATE”

> export MYCITY=Houston && echo $MYCITY

> export AGE=99 && echo $AGE
```

- && means if preceding statement is successful then carry out the proceeding statement.

- *unset* - used to clear up a previously set variable

```
> unset DATE
```