Methods and Practice of Scientific Computing (NERS/ENGR 570)

Prof. Brendan Kochunas

COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Outline

- Overview of the course
- Overview of Contemporary Scientific Computing
- Introduction to Linux

Learning Objectives: By the end of Today's Lecture you should be able to

- Understand objectives of course
- Have a sense/expectation for the course format and logistics
- Be able to give a definition scientific computing
- Feel prepared to perform hands on exercises in Linux
- Provide an educated response to the survey

Course Overview

Who are we?

Dr. Brendan Kochunas

- at University of Michigan
  - Assistant Professor (2019)
  - Adjunct Lecturer since (2016)
  - Assistant Research Scientist since (2015)
  - Post-doc (2014)
  - PhD student (2009-2013)
- Education
  - PhD in Nuclear Engineering from UM (2013)
  - M.S.E in Nuclear Engineering from UC Berkeley (2008)
  - B.S. in Nuclear Engineering from Purdue University (2006)
- Developed this course from scratch
  - 4^{th} year teaching it

GSI and Guest Lecturers

- GSI – Qicang Shen (qicangsh@umich.edu)
  - PhD Candidate in NERS
  - Did exceptional in the course the first year as a student
  - Hoping to defend at the end of this semester
- Potential Guest Lecturers
  - Malcolm Miranda (CAEN)
  - Jason Sonk (CAEN)
  - Charles Antonelli (CSCAR)
  - Prof. Brian Kiedrowski (NERS)

Other Acknowledgements

- Michigan Institute for Computational Discovery in Engineering (MICDE)
  - http://micde.umich.edu/
  - Director – Prof. Krishna Garikipati
  - Associate Director – Prof. Karthik Duraisamy
  - Assistant Director – Mariana Carrasco-Teja
- CAEN and Advanced Research Computing – Technology Services

Course Objectives

- Enable students who complete the course to produce software in their research that can eventually grow into high quality software used in:
  - industry
  - national labs
  - the open source community
- What this means is, you'll learn:
  - Best practices in software engineering
  - How to optimize code
  - How to use HPC resources
  - What tools are available
  - Overall become better and more productive programmers and computational scientists

Course Policies (1)

Lecture and Lab

- Lecture (10% of grade)
  - Simple assessment quizzes for each lecture
  - If in person, please silence cell phones
  - Laptops & Tablets are encouraged
  - **YOU ARE NOT REQUIRED TO ATTEND IN PERSON**
- Lab (40% of grade)
  - Typically 9-10 days to complete labs
  - This is to accommodate asynchronous and remote instruction.
    - However, you will have multiple assignments at once.
  - Strongly recommend synchronous participation
- Remote Instruction
  - Synchronous participation strongly encouraged!
  - All lectures and labs will be recorded and posted to Canvas

Homework and Project

- Homework (15% of grade)
  - Assignment deliverables should be your own work—unless it is a group activity
  - Your are encouraged to work with others on assignments
  - It is ok to use the internet as a resource for completing exercises
    - But if you do, you should reference the sources for your solution.
    - If we find you neglected to cite others' work you get a zero for that exercise.
- Project (35% of grade)
  - Teams of 2 to 4 (will try to divide evenly)
  - You will propose projects and iterate with us on topic, objectives, scope, etc.
    - So be thinking about them!
  - We want you to choose topics that are relevant to your research.

Course Policies (2)

Format of assignments

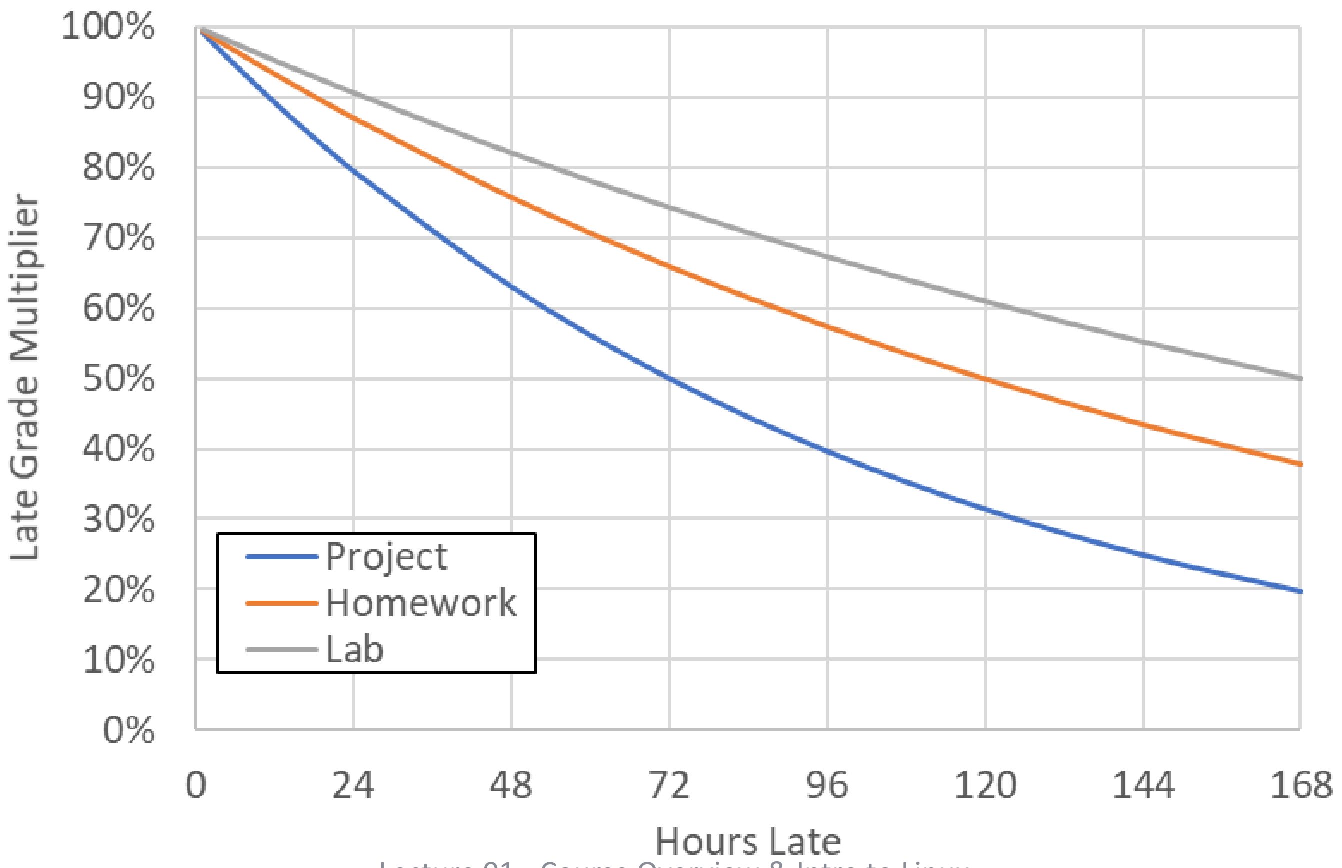
- The format of assignment deliverables will be specified on a case by case basis.
- Expected formats include:
    - Typed documents
    - Source code
    - LaTeX
    - Google Forms
    - Program output
    - Repository commits
    - Canvas Quizzes

Late Assignments

- Late **lab** assignments will have a ***7-day*** half-life.
    - e.g. If a lab assignment is completed 1-week late its maximum possible value is 50%
- Late project deliverables have a ***3-day*** half-life.
- Late homework assignments have a ***5-day*** half-life.

λ Half-Life Example

- Let's solve a differential equation!



Course Policies (3)

Announcements & Materials

- Announcements
  - Email announcements will be made on Canvas
  - Announcements may also be given verbally in class
    - These will be posted on Canvas
- Materials
  - Electronic material presented during lectures will be posted before the lecture.

Resources

- CAEN Lab
  - Redhat Linux environment
- HPC Machines (DO NOT ABUSE)
  - Flux/Great Lakes (University of Michigan)
- Git & Continuous Integration Servers
  - For some assignments

Engineering Honor Code https://elc.engin.umich.edu/honor-council/

- Engineers must possess personal integrity both as students and as professionals. They must be honorable people to ensure safety, health, fairness, and the proper use of available resources in their undertakings.
- Students in the College of Engineering community are honorable and trustworthy persons.
- The students, faculty members, and administrators of the College of Engineering trust each other to uphold the principles of the Honor Code. They are jointly responsible for precautions against violations of its policies.
- It is dishonorable for students to receive credit for work that is not the result of their own efforts.

Office Hours & Contact info

- Prof. Kochunas
  - Email: bkochuna@umich.edu
  - Office: ERB-1 4105
  - Phone: 734-763-3867
- Office Hours (Tentative)
  - Prof. Kochunas: Wednesday 3:30 pm to 4:30 pm
    - Preferably virtual
  - GSI's: Tuesday and Thursday
  - Location: ERB-1 4117.

Course Schedule (Part 1)

Date	Lecture	Lab	Topic
08/31	1		Course Overview & Introduction to Linux
09/02	2		Programming Languages: C, C++, Fortran
09/04		*1*	*Introduction to Linux*
09/09	3		Scripting with Bash and Python
09/11		*2*	*Scripting*
09/14	4		Elements of Development: Configuring, Compiling, Linking
09/16	5		Tools of the Trade: Version Control, Dev. Env
09/18		*3*	*Introduction to Great Lakes and Git*
09/21	6		Algorithms for Linear Algebra
09/23	7		Sci. Computing Libs: BLAS, LAPACK, PETSc, Trilinos
09/25		*4*	*Working with Third Party Libraries*
09/28	8		Object-Oriented Programming, Design Patterns, UML
09/30	9		Software Engineering Practices & Development Workflows
10/02		*5*	*Workflows in Practice*
10/05	10		Serial and Parallel Architectures
10/07	11		Performance and Serial Optimization
10/09		*6*	*Micro-Benchmarks and Measuring Performance*

Course Schedule (Part 2)

Date	Lecture	Lab	Topic
10/12	12		Parallel Programming Models
10/14	13		OpenMP
10/16		*7*	*Parallel Computing: OpenMP*
10/19	14		The Message Passing Interface I
10/21	15		The Message Passing Interface II
10/23		*8*	*Parallel Computing: MPI*
10/26	16		Class Selected Module 1
10/28	17		
10/30		*9*	
11/02	18		Class Selected Module 2
11/04	19		
11/06		*10*	
11/09	20		Class Selected Module 3
11/11	21		
11/13		*11*	
11/16	22		Class Selected Module 4
11/18	23		
11/20		*12*	
11/23—11/27	THANKSGIVING BREAK		

Course Schedule (Part 3) – Guaranteed Remote

Date	Lecture	Lab	Topic
11/30*	24†		Special Topic / Term Project Presentations
12/02*	25		Term Project Presentations
12/04*	OPEN LAB - Work on Term Projects		
12/07*	26		Term Project Presentations

Class Selected Modules

Topic			Description
GPU Parallelism	Lecture 1		Heterogeneous Architectures
	Lecture 2		Programming models for GPUs
		Lab	*Hardware Abstraction with Kokkos*
Testing, Testing, Testing	Lecture 1		Testing, Verification, and Validation
	Lecture 2		How to write a Unit Test
		Lab	*Automated Testing Infrastructure*
Data and Mesh Libraries	Lecture 1		Data Format Libraries: HDF5, NetCDF, SILO
	Lecture 2		Mesh Libraries: Libmesh, Exodus, others
		Lab	*Working with Data Libraries*
Package Management & Containers	Lecture 1		Package and Dependency Management with Spack
	Lecture 2		Containers: Docker and Singularity
		Lab	*Spack and Singularity on Great Lakes*
Python for HPC	Lecture 1		Using Jupyter Notebooks with HPC
	Lecture 2		Packages for Scientific Computing
		Lab	TBD
Debugging and Profiling Tools	Lecture 1		Debugging: DDT, GDB, and Valgrind
	Lecture 2		Performance: MAP, HPCToolKit, TAU
		Lab	*Make it work; Make it fast -- Debug and Optimize*
Misc. Topics	Lecture 1		QA, deployment, copyrights, and licensing
	Lecture 2		Visualization Tools and Best Practices
		Lab	*Working with Paraview*

Disclaimers

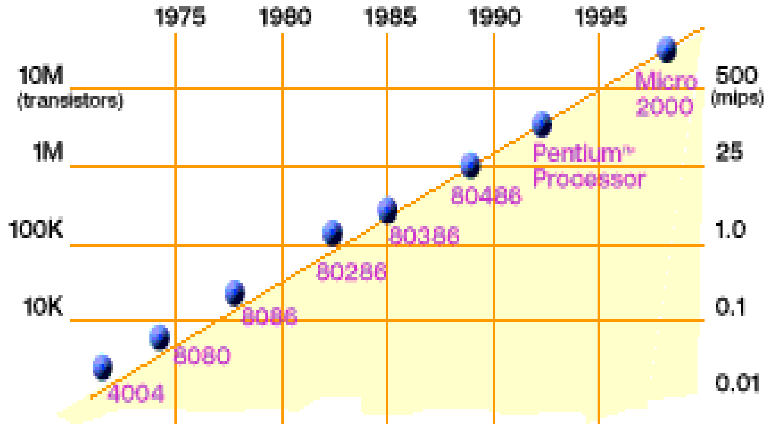
- Live programming is hard and computers are temperamental… please bear with us.
- Be prepared to learn (our expectations)
  - …how to figure things out for yourself.
    - This is an invaluable skill as a researcher and computational scientist.
  - …how to program in different languages.
    - You'll have assignments in C/C++ and Fortran.
    - None of your grade will be directly based on code written in MATLAB.
  - …a lot about a lot of things
    - Several of the topics deserve their own semester long course, and its our job to condense this into a lecture (or 2) and give you an overview, but some depth as well.
    - By the end you'll each be a "Jill" or "Jack" of "all trades", but a "master of none"
- There is no silver bullet. This is not magic. You'll learn through failure, frustration, hardwork.

Overview of Contemporary Scientific Computing

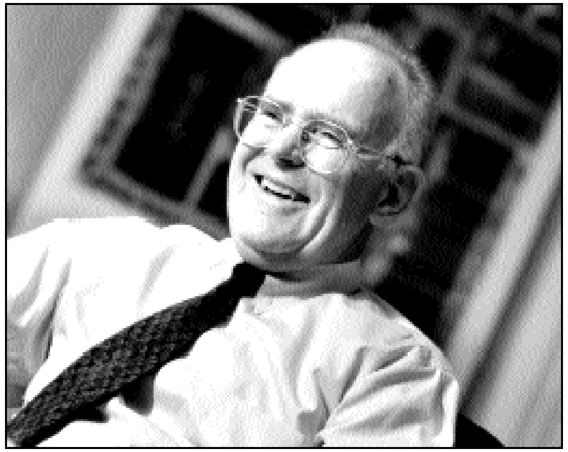
e.g. what's been going on since you've been born

Moore’s Law

- The number of transistors on a microprocessor will double every 18 months.
  - Largely been realized.
  - What does this mean for “performance”?



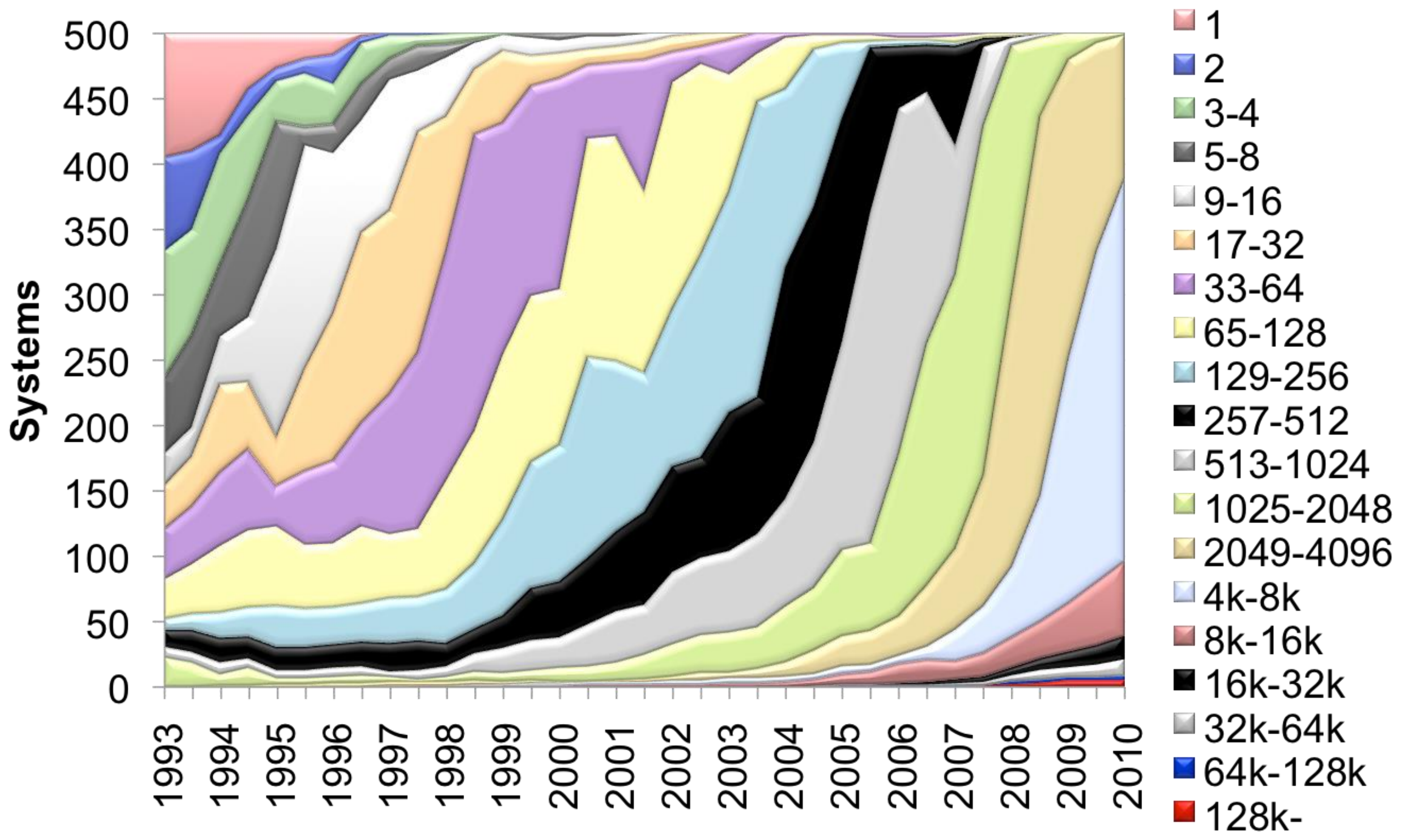
Microprocessors have become smaller, denser, and more powerful.



Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

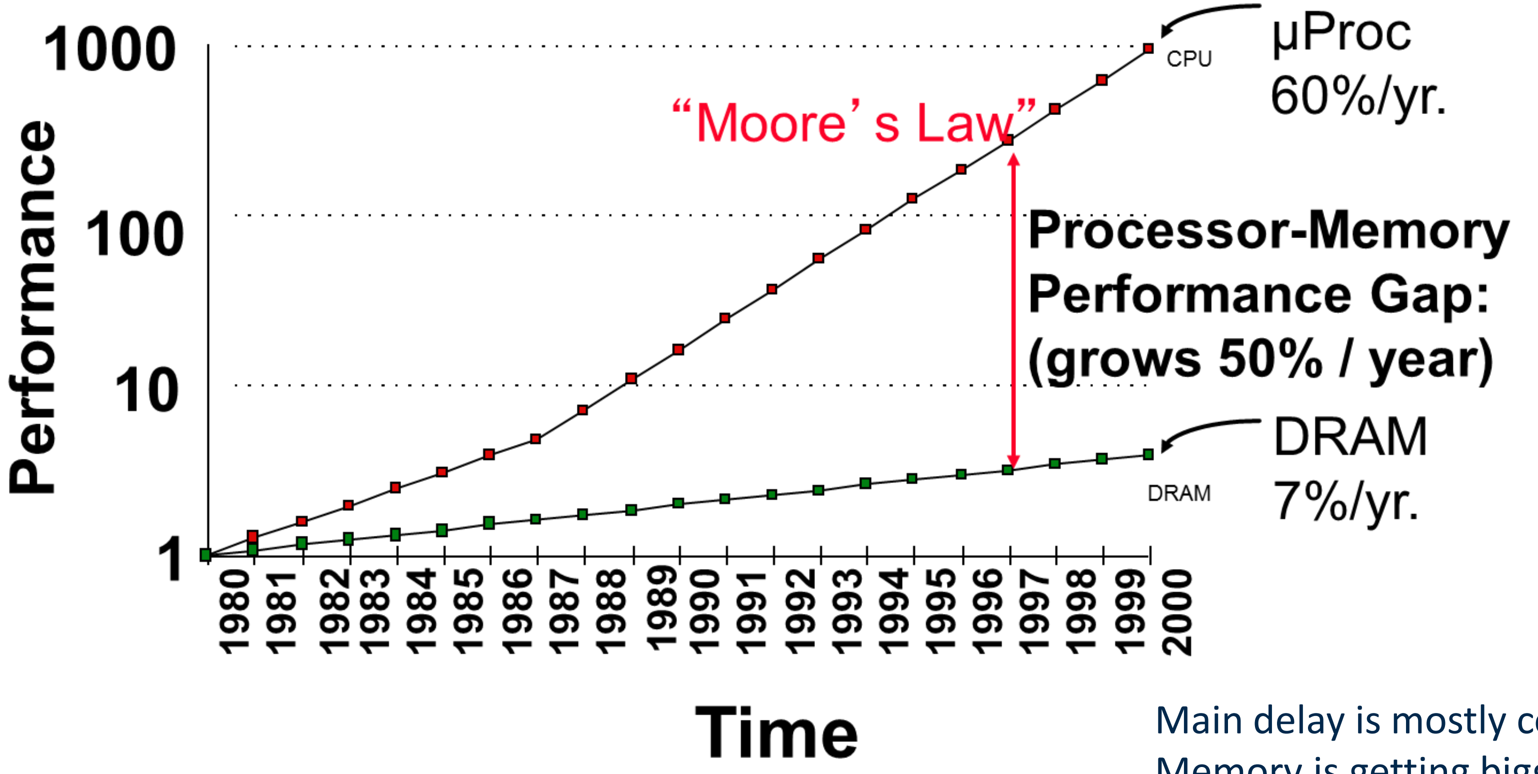
Slide Source: Jim Demmel

Core Counts



Slide Source: Jim Demmel

Processor-DRAM Gap (latency)

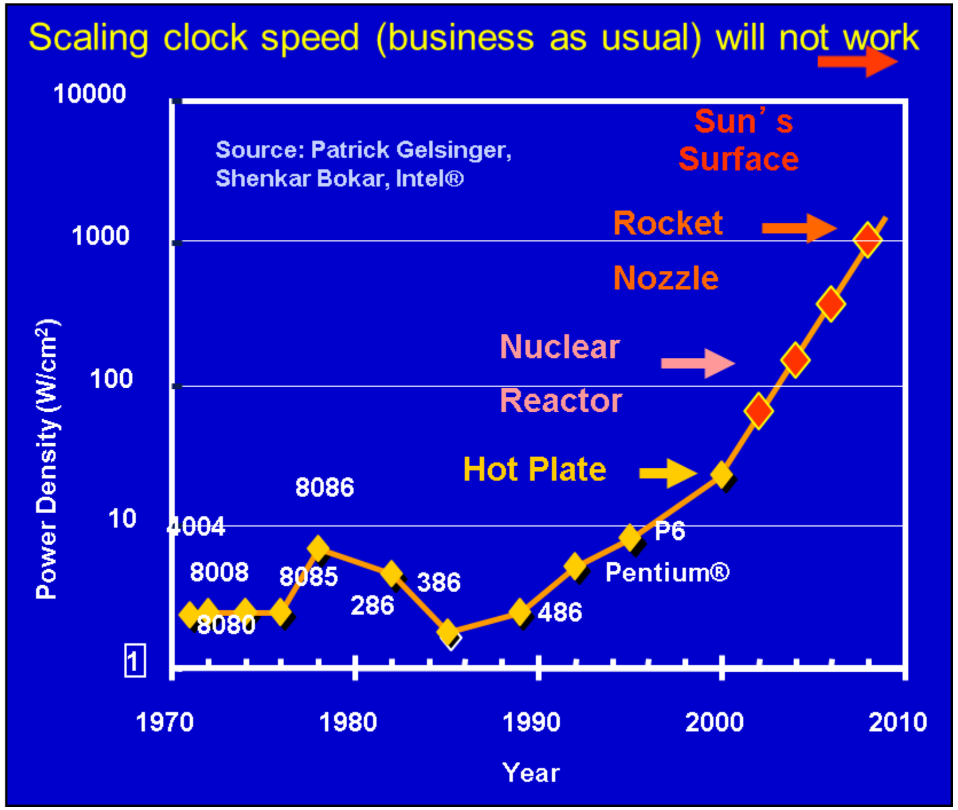


Main delay is mostly communication time
Memory is getting bigger (increases access times)

Slide Source: Jim Demmel

Present (2000-2020)

- A somewhat stable period of growth.
  - No significant changes to architecture (x86 and x86_64)
- Power density begins to limit serial performance
  - clock-speeds stop increasing
- However, Moore's law continues (in an altered form)
  - Multi-core processors are de-facto standard now.
    - Parallelism (albeit modest) becomes commercial
- Heterogeneous architectures begin to appear
  - NVIDIA GPUs
  - Intel MICs
- Multicore is more energy efficient
  - Lower clock speeds
  - Less waste of power: e.g. speculation, dynamic dependence checking



Power Density on microprocessors
(The "Power Wall")

Slide Source: Jim Demmel

TOP500

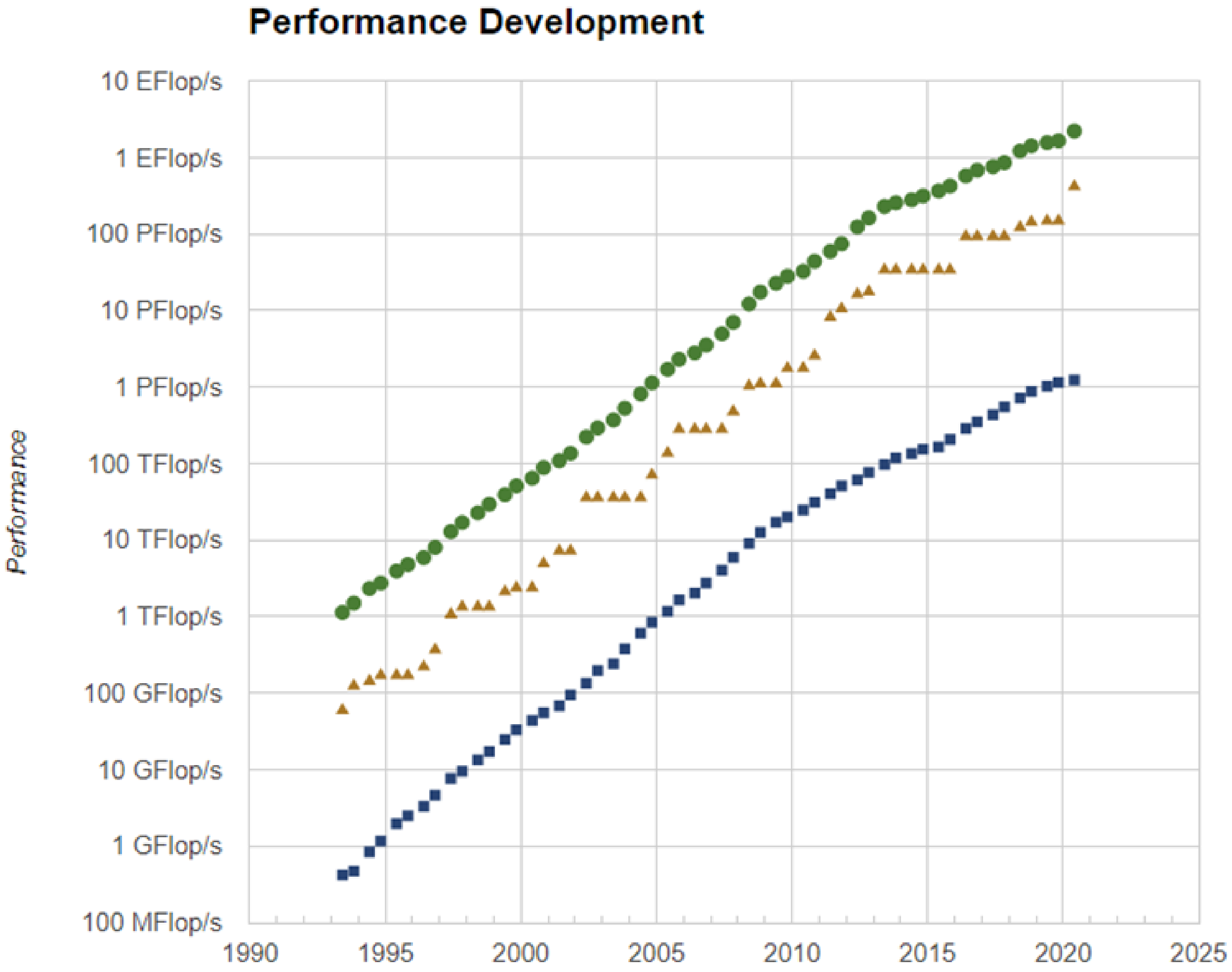
Summit System
(Currently worlds fastest computer @ 148.6 Petaflops)

- List the 500 most powerful computers in the world.
- Yardstick: Rmax of LINPACK
  - Solve Ax=b, dense problem, matrix is random
  - Problem is dominated by dense matrix-matrix multiply
- List is updated twice a year
  - International Super Computing (ISC) conference in June in Germany
  - Super Computing conference in November in U.S.
- All information available from the TOP500 website: www.top500.org
  - Additional lists: www.green500.org (for most energy efficient supercomputers)

Top 10 on the Top 500

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
6	**HPC5** - PowerEdge C4140, Xeon Gold 6252 24C 2.1GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband, Dell EMC Eni S.p.A. Italy	669,760	35,450.0	51,720.8	2,252
7	**Selene** - DGX A100 SuperPOD, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	272,800	27,580.0	34,568.6	1,344
8	**Frontera** - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR, Dell EMC Texas Advanced Computing Center/Univ. of Texas United States	448,448	23,516.4	38,745.9	
9	**Marconi-100** - IBM Power System AC922, IBM POWER9 16C 3GHz, Nvidia Volta V100, Dual-rail Mellanox EDR Infiniband, IBM CINECA Italy	347,776	21,640.0	29,354.0	1,476
10	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100, Cray/HPE Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384

Trends in HPC Platforms (Top 500)

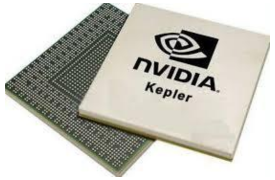


- #1 Computer
- #500 Computer
- Sum of Top500

https://www.top500.org/statistics/perfdevel/

HPC in the next 5 years (Exascale)

- Gigahertz-Kilocore-Meganode = Exascale
  - 109 x 103 x 106 = 1018
- We have GHz processors:
  - 2.x GHz
- We almost have Kilocore nodes:
  - 192 cores (2048 threads)
- What about Meganode clusters:
  - Not quite there… 18,688 nodes



Scaling to Exascale

	Tianhe-2 (2013)	Exa (2020)	Ratio to go
Number of nodes	16,000 (each 2 Ivy + 3 Phi)	1,000,000	~60
Node concurrency	24 Ivy + 171 Phi = 195 cores	1,000	~5
Node memory (GB)	88 Ivy + 8 Phi = 96	64	(1)
Node peak perf (GF/s)	422 Ivy + 3,009 Phi = 3,431	1,000	(1)
Total concurrency	3,120,000	1 B	~320
Total memory (PB)	1.536	64	~40
Total peak perf (PF/s)	54.9	1,000	~20
Power (MW)	17.8 (+ 24 MW cooling !)	20	(1)

- Big challenges to Exascale on the hardware side
  - How do you control power?
  - How do you cool it?
  - How do you handle concurrency?
- Next generation of leadership computers recently deployed
  - OLCF – Summit (IBM & NVIDIA)
  - ALCF – Aurora (Cray & Intel)
  - NERSC – Cori (Cray & Intel)

Energy to operate Supercomputer

- Today's (…well yesterdays) power costs
  - DP – Double Precision
  - FMADD –Fused Multiply Add

Operation	approximate energy cost
DP FMADD flop	100 pJ
DP DRAM read-to-register	4800 pJ
DP word transmit-to-neighbor	7500 pJ
DP word transmit-across-system	9000 pJ

- Remember that a pico (10-12) of something done exa (1018) times per second is a mega (106) somethings per second
  - 100 pJ at 1 Eflop/s is **100 MW!!!!!!!!!!**
    - For the flop/s!!!!!
- In the USA, the average home uses ~1.25 kW continuously (on average)
  - Commercial nuclear power plant produces 1000 MW.

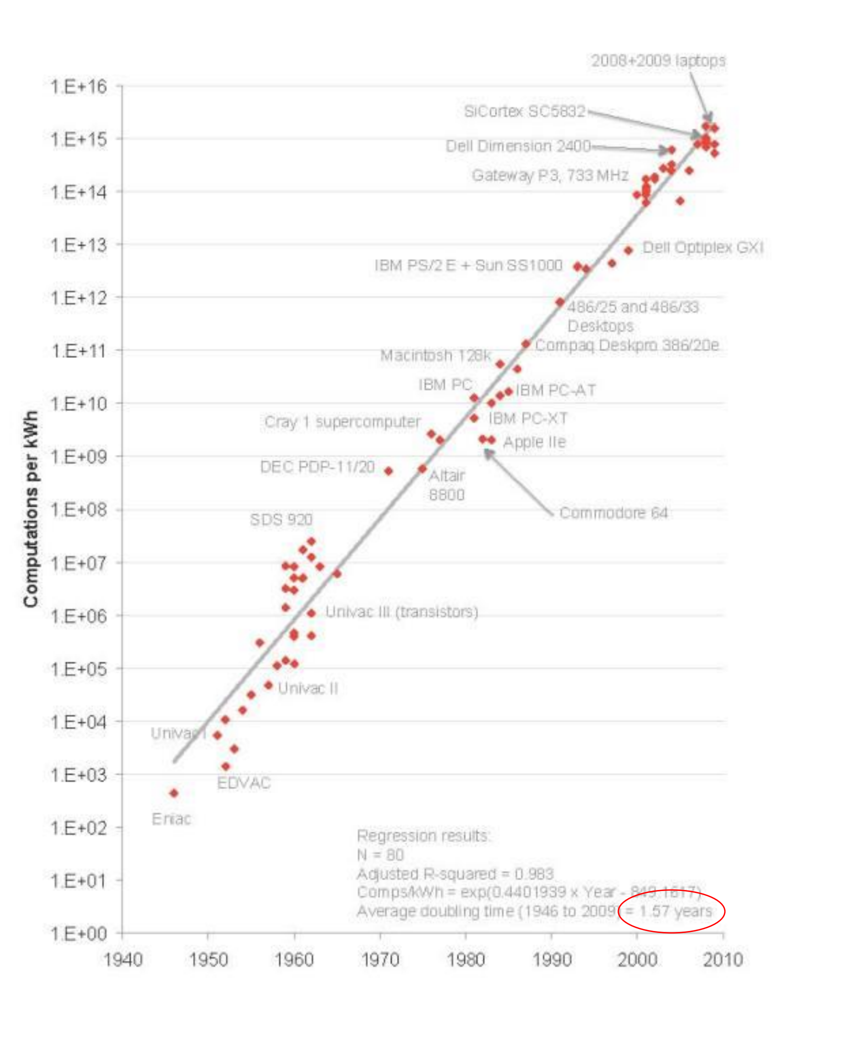
Slide Source: David Keyes

Koomey’s Law

- Relates computation to energy
  - Observed exponential behavior
- "at a fixed computing load, the amount of battery you need will fall by a factor of two every year and a half.“

Jonathan Koomey

- Computation is becoming more energy efficient
  - Cannot continue indefinitely. Projected to deviate ~2048

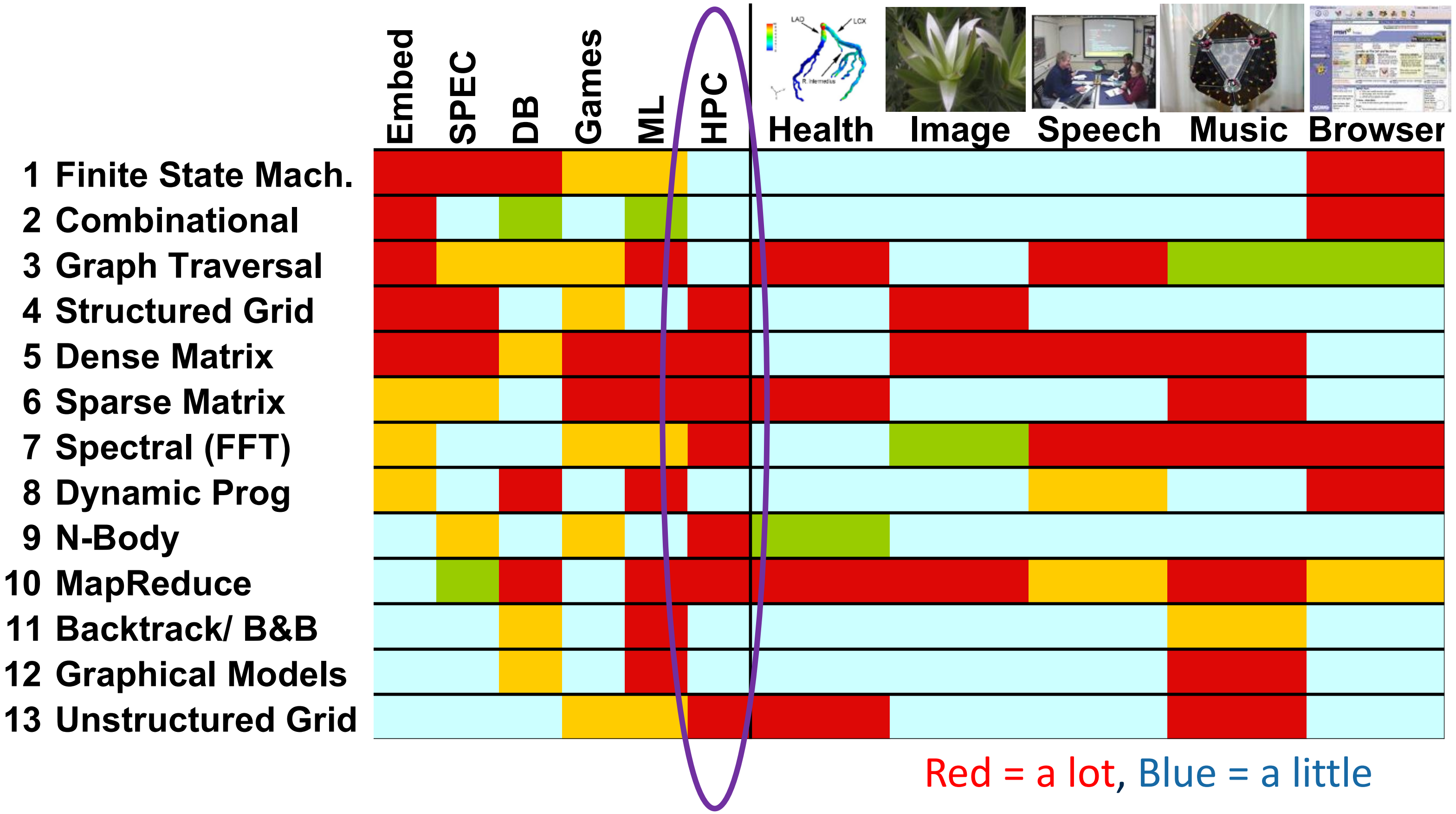


Third Pillar of Science

- Why so much effort in super computing?
- Traditional scientific and engineering method:
  - 1. Do theory or paper design
  - 2. Perform experiments, build prototypes, etc.
- Limitations
  - Too difficult—build a large wind tunnel
  - Too expensive—build a passenger jet and throw it away
  - Too dangerous—nuclear weapons
  - Too slow—climate change or astral evolution
- Computational science and engineering paradigm
  - 3. Use computers to simulate and analyze phenomenon

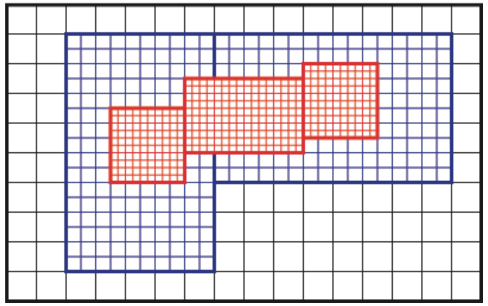
What are people doing with HPC?

- “Landscape view of Parallel Computing Research”
  - https://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf
  - https://pdfs.semanticscholar.org/presentation/515f/88754f5d8d1d22edaf94130cb1e6b4b0519c.pdf
- 13 Motifs (still dwarfs) in parallel computing
  - Previously 7 dwarfs

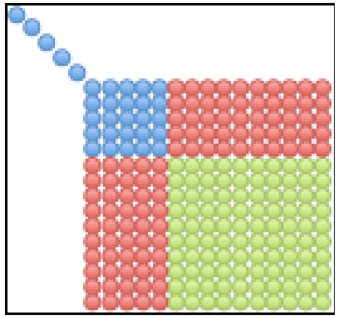


Red = a lot, Blue = a little

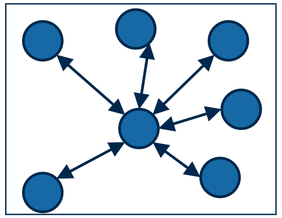
Motifs in HPC



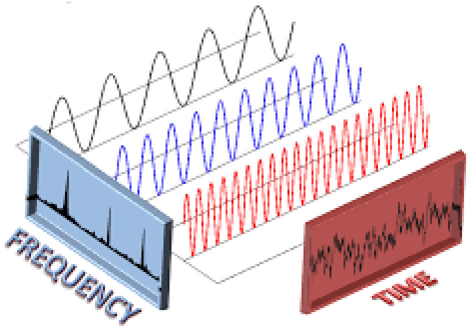
Structured Grid



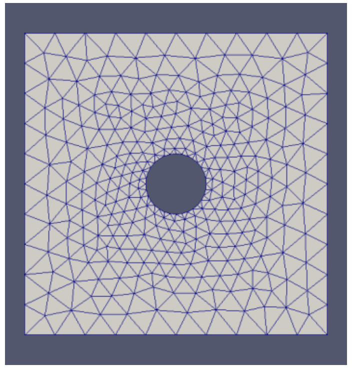
Dense linear algebra



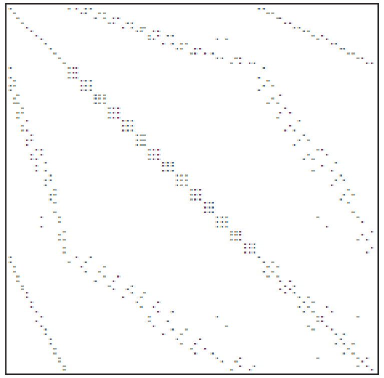
N-body
(Fast Multipole Method)



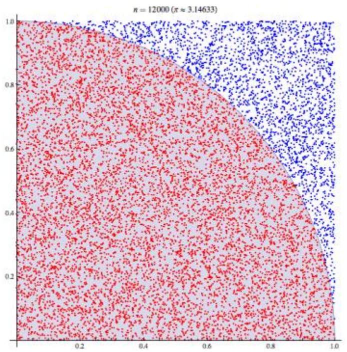
Spectral (FFT)



Unstructured Grid



Sparse linear algebra



MapReduce (Monte Carlo)

Introduction to Linux

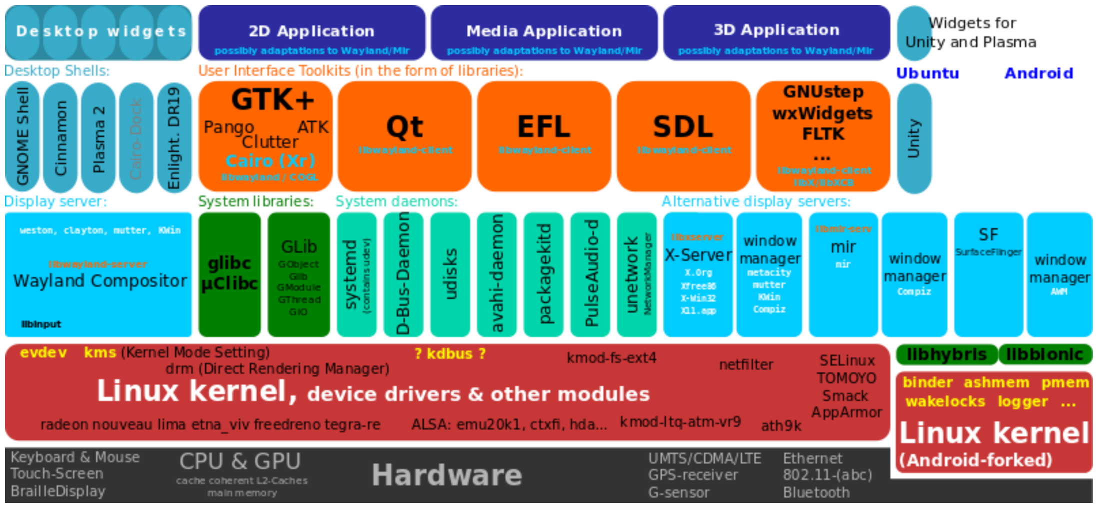
Lab on Friday: http://linuxcommand.org/tlcl.php

Background

- Linux is the open-source operating system.
  - Originally developed for the x86 architecture.
  - First released in 1991.
  - Available in many distributions both free and commercial
- Dominates the HPC world as the de-facto operating system.
  - Free or low-cost
  - Customizable
  - “Lighweight” (does not necessarily require a lot of system resources).
  - There is a Windows OS (Windows HPC Pack 2012)
- Generally preferred by programmers & developers
  - Consequently some of its features are tailored towards these activities.
- MacOS and Android are built on Linux
- Microsoft recently deployed the Windows Subsystem for Linux

Basic Linux Software Stack

https://en.wikipedia.org/wiki/File:Free_and_open-source-software_display_servers_and_UI_toolkits.svg



The Shell

- The linux shell is the environment (and set of rules) for the human/machine interface.
  - Command line interface
- The Shell is generally:
  - An interactive command language or scripting programming language
- Numerous types of shells
  - Bourne Shell (sh)
    - Written by ~~Jason Bourne~~ Stephen Bourne at Bell Labs
  - Bourne-Again Shell (bash)
    - Written as part of GNU project by Brian Fox. Default on most Linux and Mac systems.
  - Korn Shell (ksh).
    - Written by ~~hard rock group Korn~~ David Korne at Bell Labs, combined sh with csh
  - C Shell (csh)
    - Written by Bill Joy (as a graduate student at Berkeley), modeled on C
  - TENEX C shell (tcsh)
    - Fancier C shell. Written by Ken Greer for TENEX operating system.
  - Z Shell (zsh)
    - Extension of sh with some useful features from bash, tcsh, and ksh
- In this course we will focus on bash.

The Command Line Interface

NO GRAPHICS: JUST TEXT

- Three standard “units”
  - Standard input (stdin) – the keyboard
  - Standard output (stdout) – the screen (may be buffered)
  - Standard error (stderr) – also the screen, but appears immediately.

```
Last login: Fri Sep  2 15:00:01 2016 from arc-10.adsroot.itcs.umich.edu
*************************************************************************
* By your use of these resources, you agree to abide by Proper Use of   *
* Information Resources, Information Technology, and Networks at the    *
* University of Michigan (SPG 601.07), in addition to all relevant      *
* state and federal laws.                                               *
* http://spg.umich.edu/policy/601.07                                    *
*************************************************************************

dukenukem% bash
bash-4.1$ echo $SHELL
/bin/csh
bash-4.1$ echo $USER
bkochuna
bash-4.1$ echo $PATH
/afs/umich.edu/user/b/k/bkochuna/bin:/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin:.
bash-4.1$ which bash
/bin/bash
bash-4.1$ █
```

Special Characters (bash)

- Special characters can be used to “redirect” the standard units when running a command
  - Output can be “piped”, (e.g. forwarded) to another command with the “|” character
    - e.g. redirect standard output to standard input of following command
  - Output can be redirected to a file with the “>” character
  - Other input can be redirected to a program with the “<“ character instead of using standard input
- Commands can be sent to the background with “&”
- Old commands can be executed with “!” character
- Commands can also be chained together with “;”, “&&”, and “||”
- Evaluate commands inline with “`” or “$()”

The Environment

- Environment is defined by environment variables
- Some common environment variables
  - `$PATH` – the default location(s) where the shell looks for executable files
  - `$LD_LIBRARY_PATH` – where to find shared libraries
  - `$HOME` – the home directory
  - `$USER` – you
  - `$HOSTNAME` – the machine you are on
- You can define your own environment variables
  - Useful for scripting. May be used by certain programs.
- Example
  - `export PATH=/usr/bin/`
  - `echo $PATH`
    `/usr/bin/`

Useful commands: How to RTFM

- Several commands to show more information about a program or command
  - `info`
  - `man`
  - `help`
  - Command options `-h`, `--help`
- Google is your best friend
  - Before asking the GSIs or me, try Google!
- All kinds of complicated commands can be found on Stack Overflow or other forums
  - 99.9% chance that someone else has already asked your question

Useful commands: Working with files

Directories

- `cd` – "change directory"
  - ".." is parent directory, "." is current directory, "-" is previous directory
- `ls` – "List Segments" (list files/directories)
- `pwd` – "Present Working Directory" (current location)
- `mkdir` – "Make Directory"
- `rmdir` – "Remove Directory" (must be empty)
  - `rm -r` "Remove (recursively)" can delete non-empty directories

Files

- `cp` – "Copy"
- `mv` – "Move"
- `rm` – "Remove"
- `ln` – link (e.g. create a shortcut)
- `chmod` – modify file permissions
- `chgrp` – "Change Group" ownership
- `chown` – "Change owner"
- `tar` – "Tape Archive." Stores/Extracts files from tape/disk archives
- `gzip` / `gunzip` – File compression/decompression

Useful commands: Probing

Specifically

- `top` (or `htop`) – "Table of Processes". List of all processes running on machine
- `who` – list all users logged in
- `ps` - list all processes running (brief version of top)
- `kill` – stop running a process
- `lscpu` - show machine specs
    - `cat /proc/cpuinfo`
- `hostname` – the name of the machine you're on
- `du` – "disk usage" see how big a directory is
- `df` – "disk free space" see how much space is left
- `which` – shows you full path to command

Generally

- Depending on the machine, there may be several other commands used to probe different
- For HPC platforms it is useful to be able to view who is running jobs, how long they have left, how many CPU
    - We will cover all of this later.

Users

- `finger` – list information about user
- `groups` – list groups that a user is a member of
- `date` – the current time

Useful commands: Navigating Servers

- `ssh` – Secure Shell. Basis for remote connections in Linux (e.g. login to a remote server)
- `scp` – Secure Copy. copy files to/from a remote server
- `sftp` – Secure File Transfer Protocol. more interactive version of `scp`
- `rsync` – remote synchronization, a little better than `scp`
- `wget` – Web get. Download stuff from webpages.

Useful commands: Searching

- `grep` – “Globally search Regular Expression Print”
  - Finds and prints all lines matching a given string/regex in files within a specified scope
- `find` – Finds all files matching a certain criterion
- `find -name “blah*”`
  - Find all files whose names begin with “blah”
- <tab> - auto-complete command, directory name, filename
  - Press it twice to list available matches

Useful commands: Parsing

- `sed`
  - Can print specific lines from a file
    - `sed -n '50,100 p' blah.txt`
    - Print lines 50 through 100 from blah.txt
  - Find & replace strings in a file
    - `sed -i 's/foo/bar/g' blah.txt`
    - Replace all instances of 'foo' with 'bar' in blah.txt
- `awk` (`gawk`) – Text file parsing language
- `cat` – concatenate two files
- `diff` (`sdiff`) – show differences between two files

Useful Commands: History

- ctrl+r – searches command history for a command
- `!<exp>` - rerun the last command starting with `<exp>`
- `history` – shows your command line history
  - `history -cw` – clear your history (what are you hiding?! admin passwords?!)
    - Don't forget the `.bash_history` file.

Useful commands: Misc

- ctrl+c – kills a process
- ctrl+z - pauses a process
- `bg` – resume paused process in background
- `fg` – either sends background process to foreground, or resumes paused process in foreground
- `pushd`, `popd`, `dirs` – useful for storing and returning to a specific directory path
- `head`, `tail`, `more`, `less` – quickly view (pieces of) files without entering a text editor
- `touch` – change the time-stamp of a file
- `mail` – send emails from the command line.

Regular Expressions (regex)

Regular Expressions

- A sequence of characters that define a search pattern
- Many special characters to facilitate advanced commands
  - . - wildcard (any character is a match)
  - ? - matches preceding element 0 or 1 times
  - * - matches preceding element 0 to many times
  - + - matches preceding element 1 to many times
  - [abc] = a OR b OR c
  - [^abc] = NOT (a OR b OR c)

Other Special Characters

- ^ = beginning of line
- $ = end of a line
- \d matches a digit (equivalent to [0-9])
- \D matches a non-digit (equivalent to [^0-9])
- | separates multiple expressions
- {M,N} gives a range of minimum M to maximum N matches
  - {M} matches exactly M times
  - {M,} matches at least M times
  - {0,N} matches at most N times

Examples:

- `egrep '^foo|bar$' blah.txt` – match any line in blah.txt that starts with foo or ends with bar
- `egrep '[0-9] +[^0-9]' blah.txt` – match any line with a digit, one or more spaces, and a letter
- `egrep '^[aeiou]' blah.txt` – match any line that starts with a vowel

Bonus Useful commands: Basic Sysadmin

- `sudo` – "Super User DO" execute command with administrator privileges
  - Will ask for admin password
  - Usually won't be available on a large shared machine, but be careful if you actually have privileges
- `useradd` – create a new user
- `usermod` – modify an existing user
- `passwd` – change password
- `yum` (`rpm`) – Yellowdog Updater, Modified (Redhat/CentOS). Manage installed software.
  - `apt` – Advanced Packaging Tool (Ubuntu). Manage installed software.
    - `apt-get`, `apt-cache`
- `cron` (`crontab`) – schedule a recurring task