Institute for Cyber-Enabled Research

Dr. Dirk Colbry
Director, High Performance Computing Center

MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6QFiAQ

ICER

Agenda

- What is iCER / HPCC
- Common classes problems
- Overview of Hardware
- Getting started, Seven Steps to High Performance
- Running in parallel
- Tips and tricks

Institute for Cyber Enabled Research

The Institute for Cyber-Enabled Research (iCER) at Michigan State University (MSU) was established to coordinate and support multidisciplinary resource for computation and computational sciences. The Center's goal is to enhance Michigan's national and international presence and competitive edge in disciplines and research thrusts that rely on advanced computing.

https://wiki.hpcc.msu.edu/x/6QFiAQ

WMU/ MSU Partnership

- WMU researchers are HPCC Buy-in User

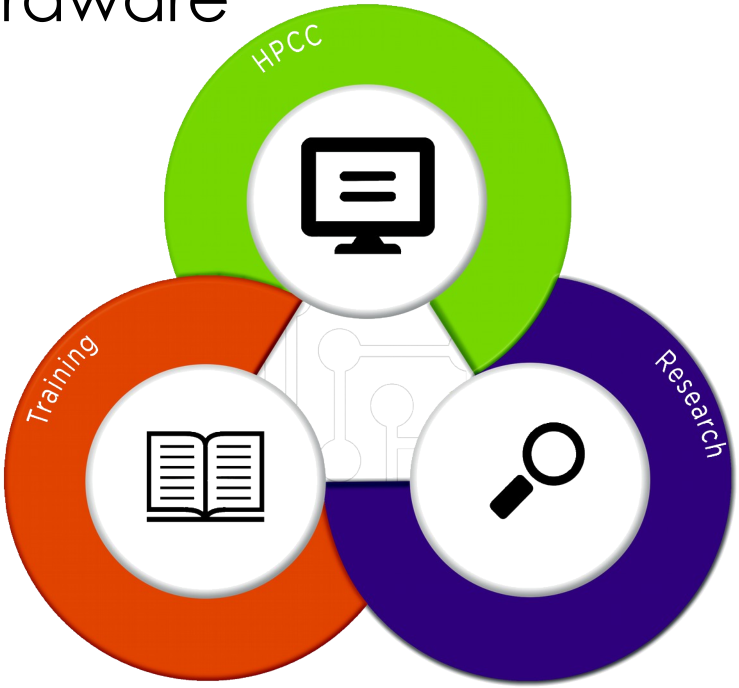
  wmichhelp@hpcc.msu.edu

- Please make sure you send us your login name
  - Emailing us from your wmich account should be more than sufficient.

Research Resource

iCER is a research unit at MSU. We provide:

- Advanced computing hardware
- Software-as-a-service
- Training
- Consulting
- Proposal writing support



MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6QFiAQ

ICER

Advanced Architectures

- Anything more advanced than your desktop
- Local resources
  - Lab, Department, Regional (iCER)
- National resources
  - NSF (XSEDE), DOE (Titan) , Others
- Commercial Resources (cloud computing)
  - Amazon, Azure, Liquid Web, Others

Why use Advanced Computing Hardware?

- Science takes too long
- Computation runs out of memory
- Needs licensed software
- Needs advanced interface (visualization/database)
- Lots of file i/o

Bigger & Better ?

- The goal of iCER is NOT:
  - Kflops / second
- Instead, the goal of iCER IS:
  - KSciences / second
- Doing More Science, Faster
  - Reducing the "Mean time to Science"
- iCER is designed to help researchers do their science and when appropriate scale them up to national resources

MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6QFiAQ

ICER

Hardware Highlights

- > 600 nodes, 7600 computing cores, 50 TB RAM
  - Large Memory Nodes (up to 6TB!)
  - GPU clusters (K20, M1060)
  - Xeon PHI cluster (5110p)
  - + 8000-core condor cluster
- High-speed file servers
  - 360TB parallel scratch file space
  - 1PB replicated home/research file servers
- High-speed network (FDR,QDR,SDR,10g)
- Evaluation nodes

https://wiki.hpcc.msu.edu/x/6QFiAQ

Shared System

- Development Nodes
  - Interactive computers for testing software and compiling code
- Evaluation Nodes
  - Interactive computers for evaluating older and cutting edge systems
- Queuing system
  - Submit jobs to the queue
  - Jobs will run as resources become available
  - Up to one-week walltime
  - Up to 520 processing cores per user at any time
  - The larger the job, the longer/harder it is to schedule

https://wiki.hpcc.msu.edu/x/6QFiAQ

Software Stack

- Compiled open-source software stack
  - Close to 2000 titles!
- Optimized Math/Communications libraries
- Some commercial software available
  - Due to licensing limitations we can not provide all of our software to WMU users
  - We can help install software for you

Full list: http://wiki.hpcc.msu.edu

MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6QFiAQ

ICER

Miscellaneous Musings

iCER provides many other services:

- Data sharing (Globus Online Subscriptions)
- Visualization Servers
- Virtual Computing Lab
- Specialized bioinformatics support (BICEP)
- Support for scaling to XSEDE resources
- + … ?

Agenda

- What is iCER / HPCC
- Common classes problems
- Overview of Hardware
- Getting started, Seven Steps to High Performance
- Running in parallel
- Tips and tricks

What problems are we solving?

- Simulations
- Data Analysis
- Search

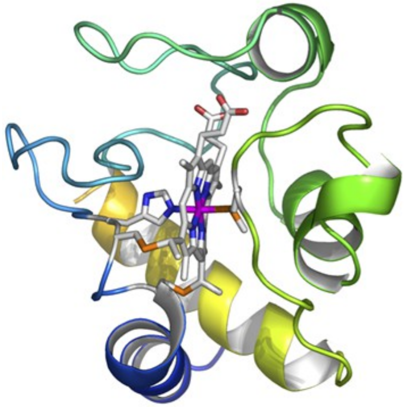
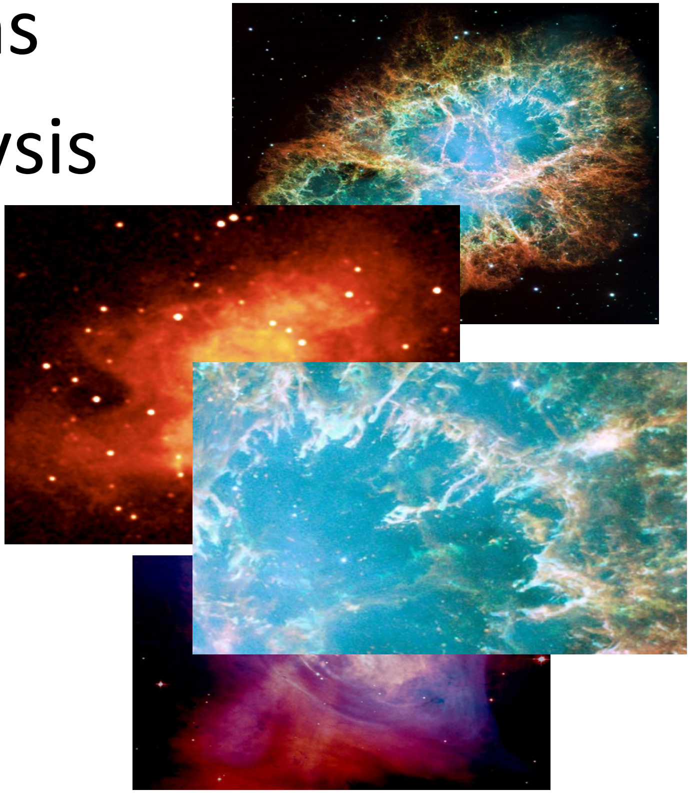


Image Provided by Dr. Warren F. Beck, MSU



Images from, “Understanding the H2 Emission from the Crab Nebula”, C.T. Richardson, J.A. Baldwin, G.J. Ferland, E.D. Loh, Charles A. Huehn, A.C. Fabian, P.Salomé

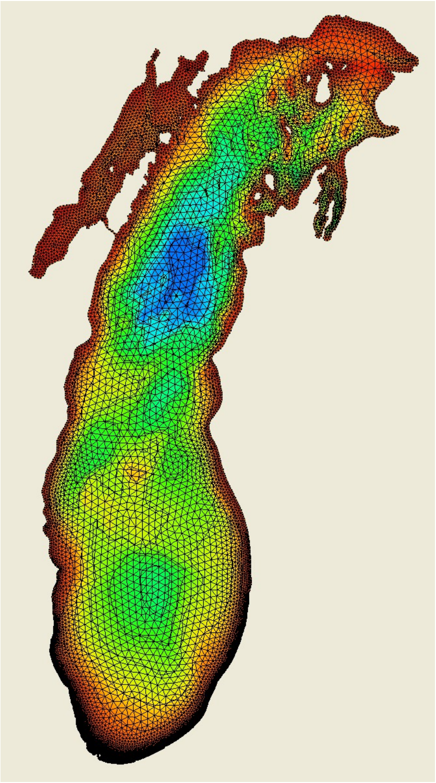


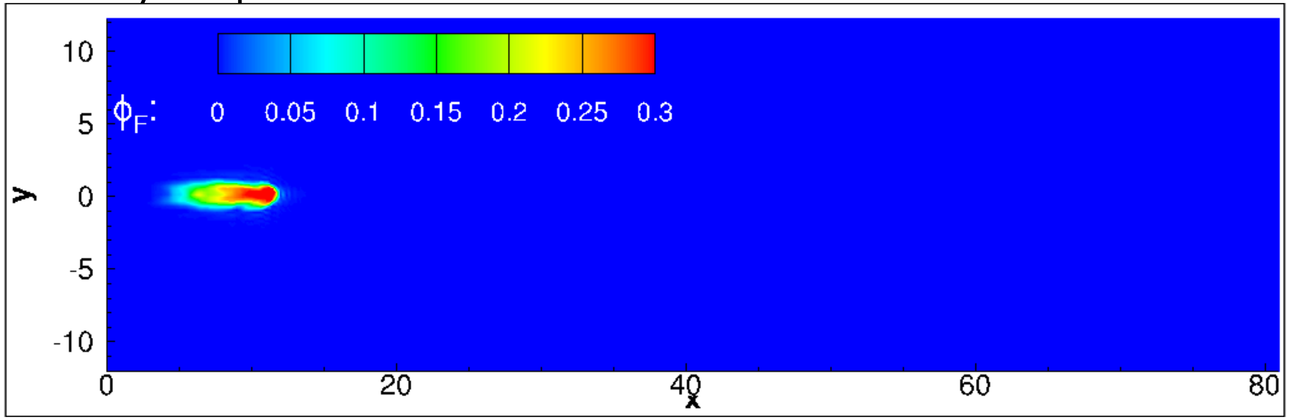
Image Provided by Dr. Mantha Phanikumar, MSU

https://wiki.hpcc.msu.edu/x/6QFiAQ

Simulations

- Typically System of PDE (Partial Differential equations)
    - Fluid dynamics
    - Finite element analysis
    - Molecular dynamics
    - Weather
    - Etc.
- Mathematically equivalent to inverse of a matrix

Premixed mixture of H2 -air auto igniting and flame propagation at supersonic flow Provided by Dr Jabari and Mani (Abolfazl) Irannejad

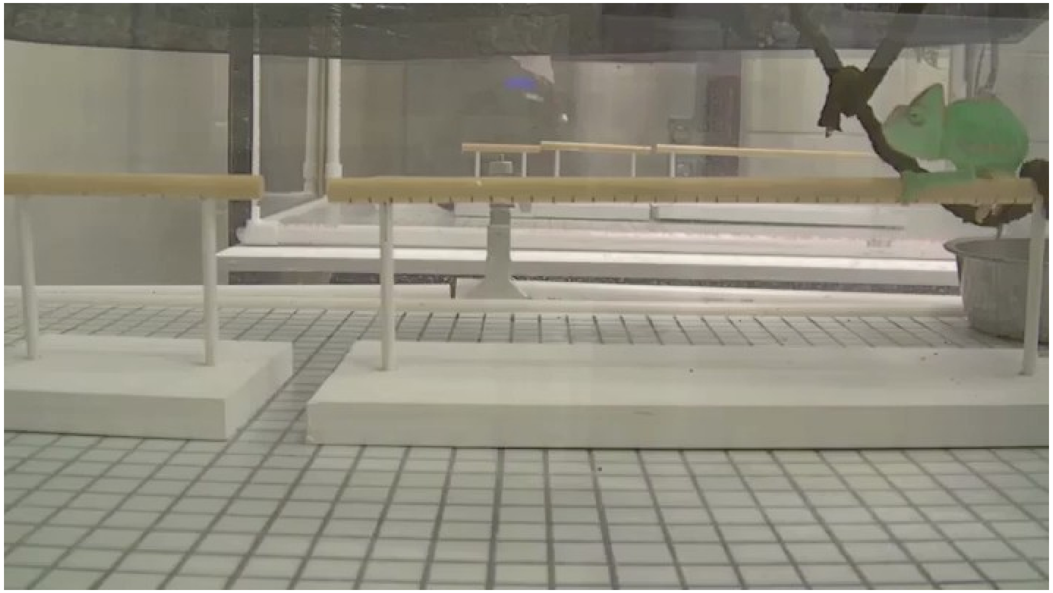


https://wiki.hpcc.msu.edu/x/6QFiAQ

Data Analysis

- Computer vision tasks
- Some Bioinformatics
- Astrophysics
- Etc.

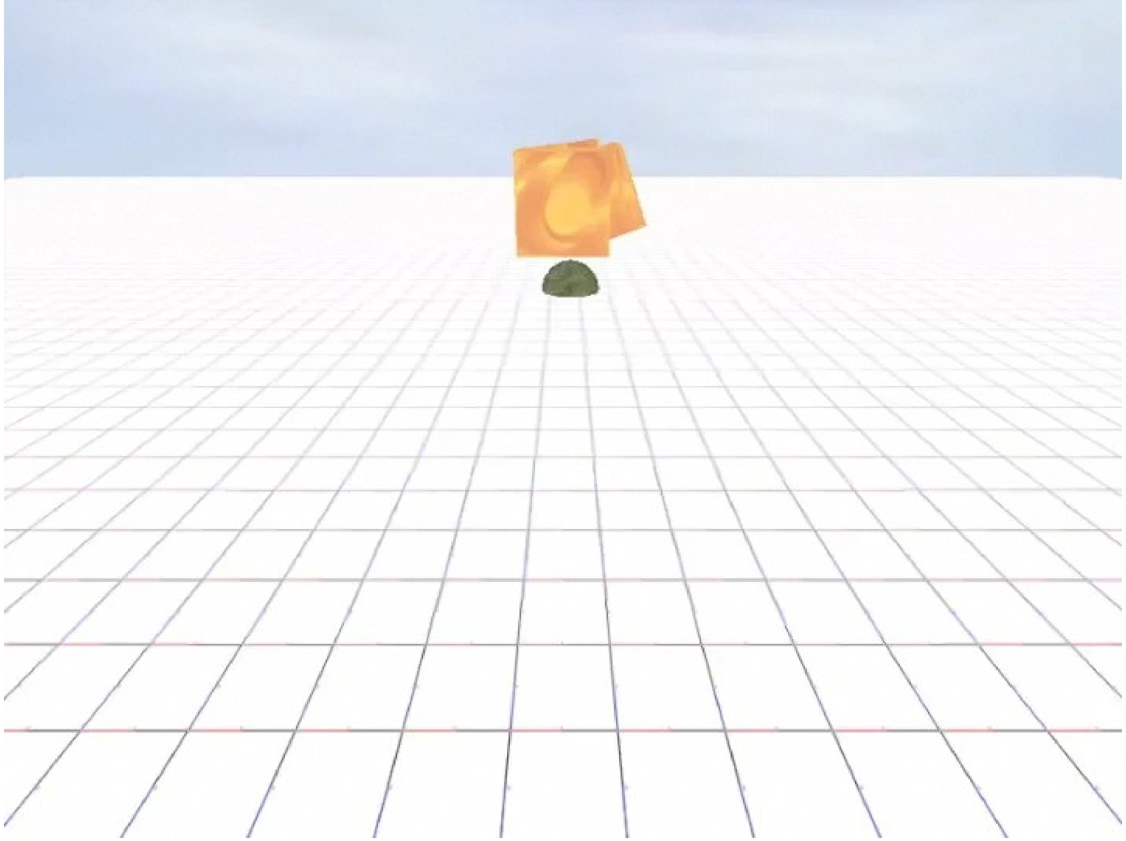
Video Provided by Dr. Fred Dyer



https://wiki.hpcc.msu.edu/x/6QFiAQ

Search

- Genome sequencing
- Analytics
- Optimization
- Etc.



Evolution of an artificial organism that can move and forage for food, Dr. Nicolas Chaumont

https://wiki.hpcc.msu.edu/x/6QFiAQ

Agenda

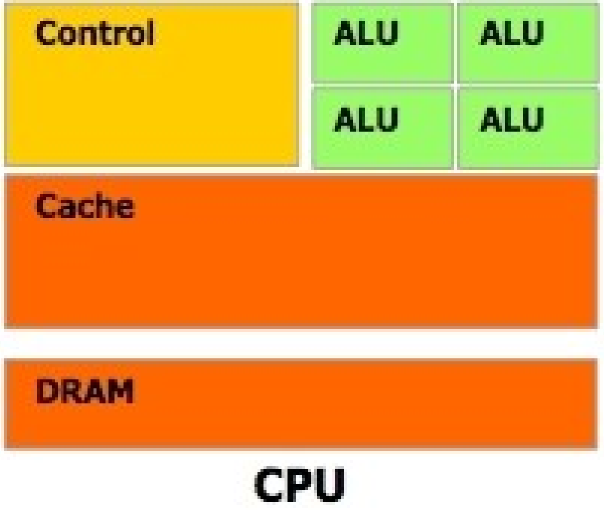
- What is iCER / HPCC
- Common classes problems
- Overview of Hardware
- Getting started, Seven Steps to High Performance
- Running in parallel
- Tips and tricks

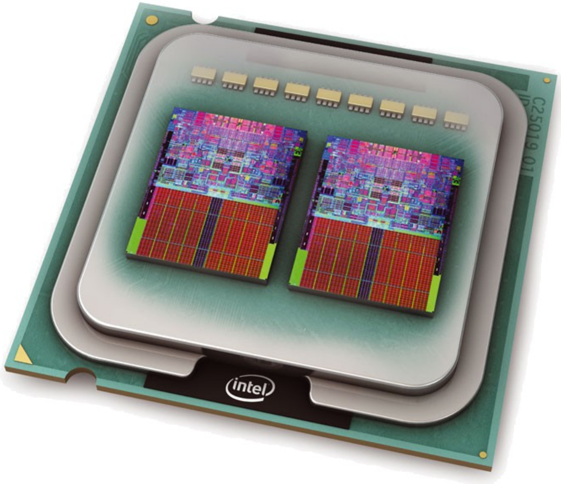
Large Shared Memory Systems (Fat Nodes)

https://wiki.hpcc.msu.edu/x/6QFiAQ

Shared Memory Communication

- Fast!
- Cores on a system share the same memory
- OpenMP
- Fat nodes
  - 96 cores
  - 6TB of memory





https://wiki.hpcc.msu.edu/x

Accelerated Systems

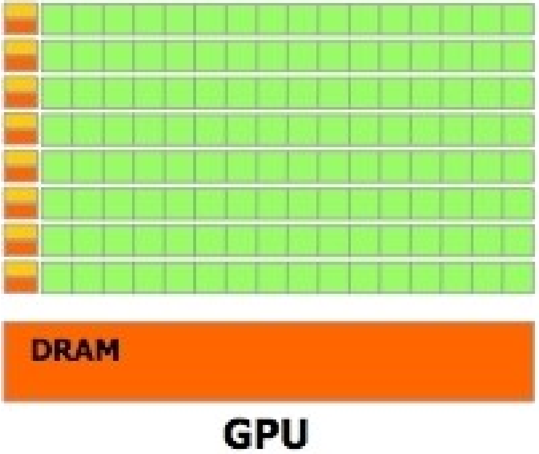
https://wiki.hpcc.msu.edu/x/6QFiAQ

GPU

NVIDIA

CUDA™ TEACHING CENTER

- Cards used to render graphics on a computer
- Hundreds of cores
- Not very smart cores
- But, if you can make your research look like graphics rendering you may be able to run really fast!

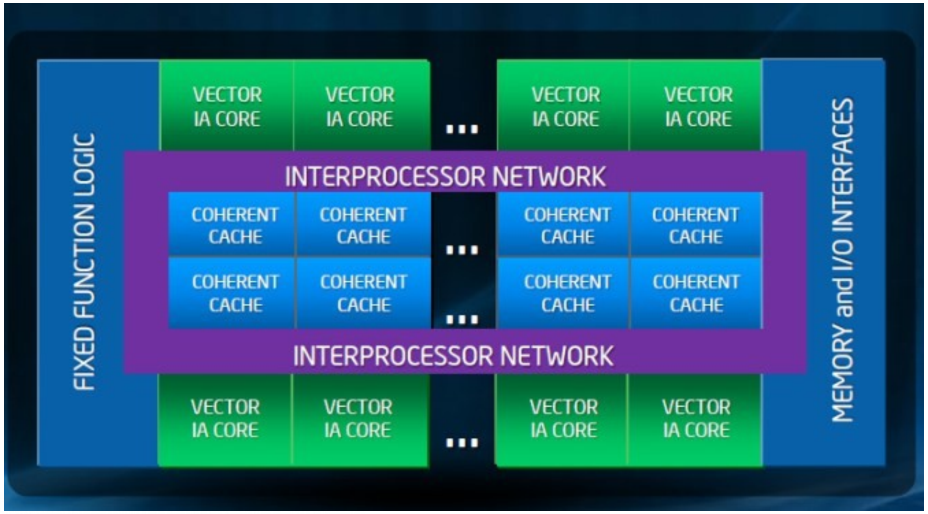


MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6(

Intel Xeon Phi

- Cross between CPU and GPU
- About 61 Pentium III cores
  - Less cores/slower than GP''
  - Easier to use than GP



MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6QFiAQ

ICER

High Throughput HTCondor Cluster

https://wiki.hpcc.msu.edu/x/6QFiAQ

MSU HTCondor Cluster

- Runs like a screen saver and Scavenges CPU cycles:
  - Approximately 400+ nodes
  - Approximately 7000 cores
  - Windows 7

HTCondor High Throughput Computing

MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6QFiAQ

ICER

Agenda

- What is iCER / HPCC
- Common classes problems
- Overview of Hardware
- Getting started, Seven Steps to High Performance
- Running in parallel
- Tips and tricks

Steps in Using the HPCC

1. **Get an account**
2. Install needed software (SSH, SCP, X11)
3. Transfer input files and source code
4. Compile/Test programs on a developer node
5. Write a submission script
6. Submit the job
7. Get your results and write a paper!!

Accounts

- Pis can request accounts though Don Weber:

  donald.weber@wmich.edu

  - Each account has access to:
    - 50 GB of replicated file spaces
    - 520 processing cores
    - 360 TB of high-speed scratch space
  - Also available: shared group folders

  - We have temporary accounts for today.

https://wiki.hpcc.msu.edu/x/vo2y

https://wiki.hpcc.msu.edu/x/6QFiAQ

Steps in Using the HPCC

1. Get an account
2. **Install needed software (SSH, SCP, X11)**
3. Transfer input files and source code
4. Compile/Test programs on a developer node
5. Write a submission script
6. Submit the job
7. Get your results and write a paper!!

Required Software

- Secure Shell (ssh)
- File transfer
  - Secure Copy (scp)
  - Mapping home directories
- Graphical User Interface (x11)
  - Optional

Apple

- Run Terminal program
  - ssh – already installed

```
ssh –X userid@hpcc.msu.edu
```

  - scp – already installed

```
scp ./mylocalfile userid@hpcc.msu.edu:~/mylocalfile
```

- May need to install Xquarts (mac X11 Server)
  - Installer should be on USB drive

https://wiki.hpcc.msu.edu/x/6QFiAQ

Windows Software

- PuTTY:
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/
- Xming:
  - http://www.straightrunning.com/XmingNotes/
- Xming install:
  - https://wiki.hpcc.msu.edu/x/swAk
- WinSCP:
  - http://winscp.net

MobaXterm (windows)

- Complete toolbox for remote computing:
  - Multi-tab terminal
  - X11 server
  - SSH
  - File transfer
  - More

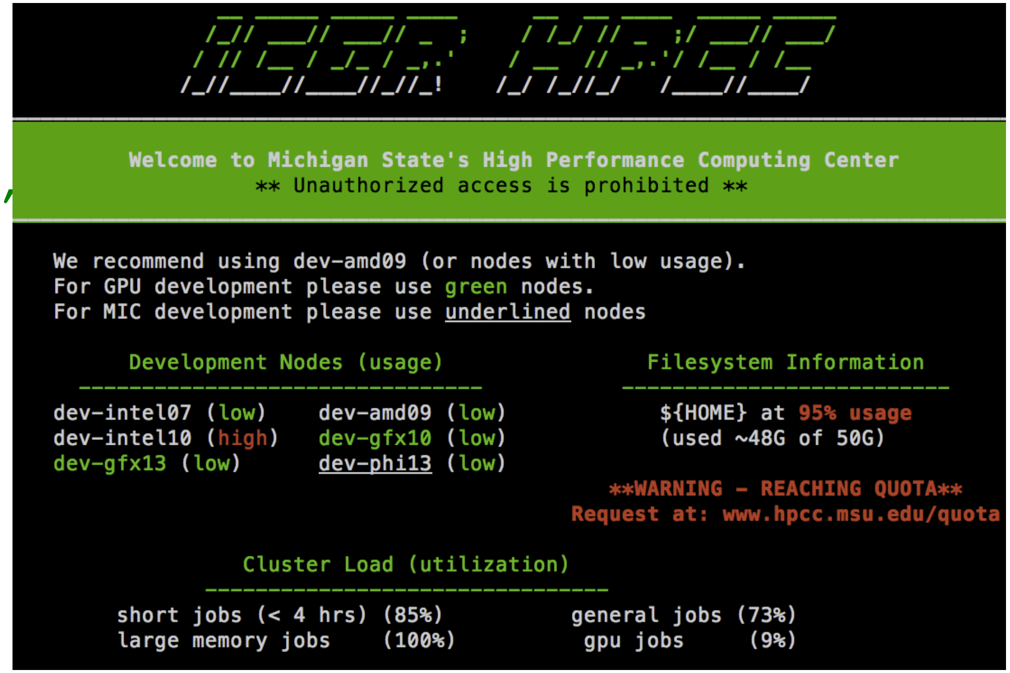
- Opensource
- http://mobaxterm.mobatek.net/

Exercise: Portable HPCC

- If you have Windows
- Plug in your USB thumb drive
- Open the thumb drive folder and select
  - PortableApps
- You should see a new menu in your system tray for navigating

Exercise: Connect to HPCC

- Step 1: Log into gateway.hpcc.msu.edu
- Step 2: ssh into a dev node (developer node)
  - `ssh dev-intel10`
- Step 3: execute a command
  - `echo “Hello world’`



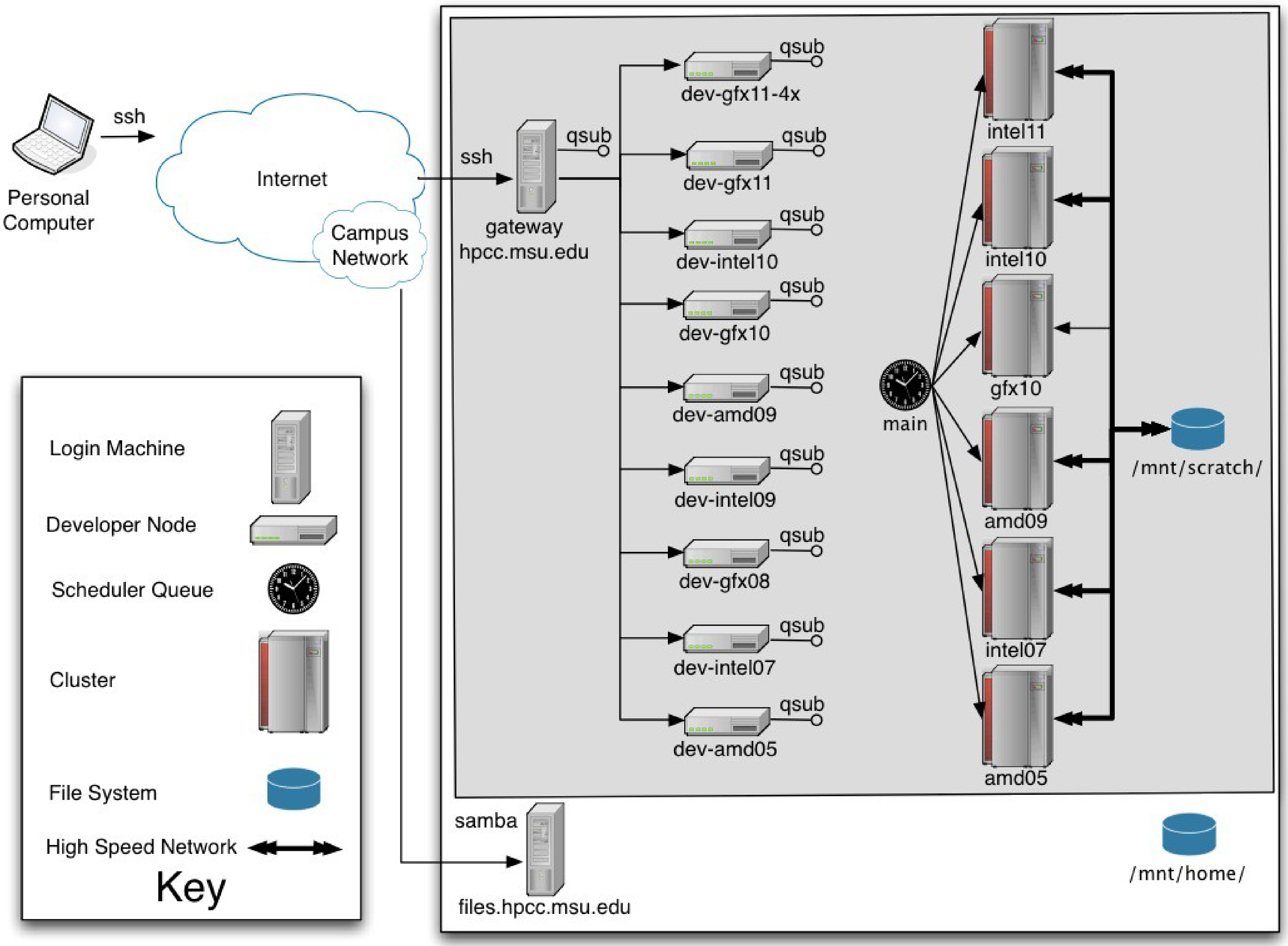
```
Welcome to Michigan State's High Performance Computing Center
           ** Unauthorized access is prohibited **

We recommend using dev-amd09 (or nodes with low usage).
For GPU development please use green nodes.
For MIC development please use underlined nodes

      Development Nodes (usage)                 Filesystem Information
    ----------------------------------        --------------------------
dev-intel07 (low)     dev-amd09 (low)            ${HOME} at 95% usage
dev-intel10 (high)    dev-gfx10 (low)            (used ~48G of 50G)
dev-gfx13 (low)       dev-phi13 (low)
                                               **WARNING - REACHING QUOTA**
                                             Request at: www.hpcc.msu.edu/quota

                       Cluster Load (utilization)
                  ----------------------------------
     short jobs (< 4 hrs)  (85%)          general jobs (73%)
     large memory jobs     (100%)          gpu jobs       (9%)
```

https://wiki.hpcc.msu.edu/x/6QFiAQ


ssh
Personal Computer
Internet
Campus Network
ssh
qsub
gateway
hpcc.msu.edu
qsub
dev-gfx11-4x
qsub
dev-gfx11
qsub
dev-intel10
qsub
dev-gfx10
qsub
dev-amd09
qsub
dev-intel09
qsub
dev-gfx08
qsub
dev-intel07
qsub
dev-amd05
main
intel11
intel10
gfx10
amd09
intel07
amd05
/mnt/scratch/
samba
files.hpcc.msu.edu
/mnt/home/
Login Machine
Developer Node
Scheduler Queue
Cluster
File System
High Speed Network
Key

Command Line Interface

- Command Line Interface (CLI)
- Shell
  - Program to run Programs

More information, I recommend:
http://www.softwarecarpentry.org/

- Bash (Bourne Again Shell)
- Use it because:
  - many tools only have command-line interfaces
  - allows you to combine tools in powerful new ways

Module System

- To maximize the different types of software and system configurations that are available to the users, HPCC uses a Module system
- Key Commands
  - **module avail** – show available modules
  - **module list** – list currently loaded modules
  - **module load** modulename – load a module
  - **module unload** modulename – unload a module
  - **module spider keyword** – Search modules for a keyword

Exercise – Module

- List loaded modules
  - `module list`
- Show available modules:
  - `module avail`
- Try an example (Shouldn’t work):
  - `powertools`

Exercise: getexample

- Load a module:

  -
    ```
    module load powertools
    ```
- Show powertools (should work now):

  -
    ```
    powertools
    ```
- Run the “getexample” powertool

  -
    ```
    getexample
    ```
- Download the helloworld example

  -
    ```
    getexample helloworld
    ```

Steps in Using the HPCC

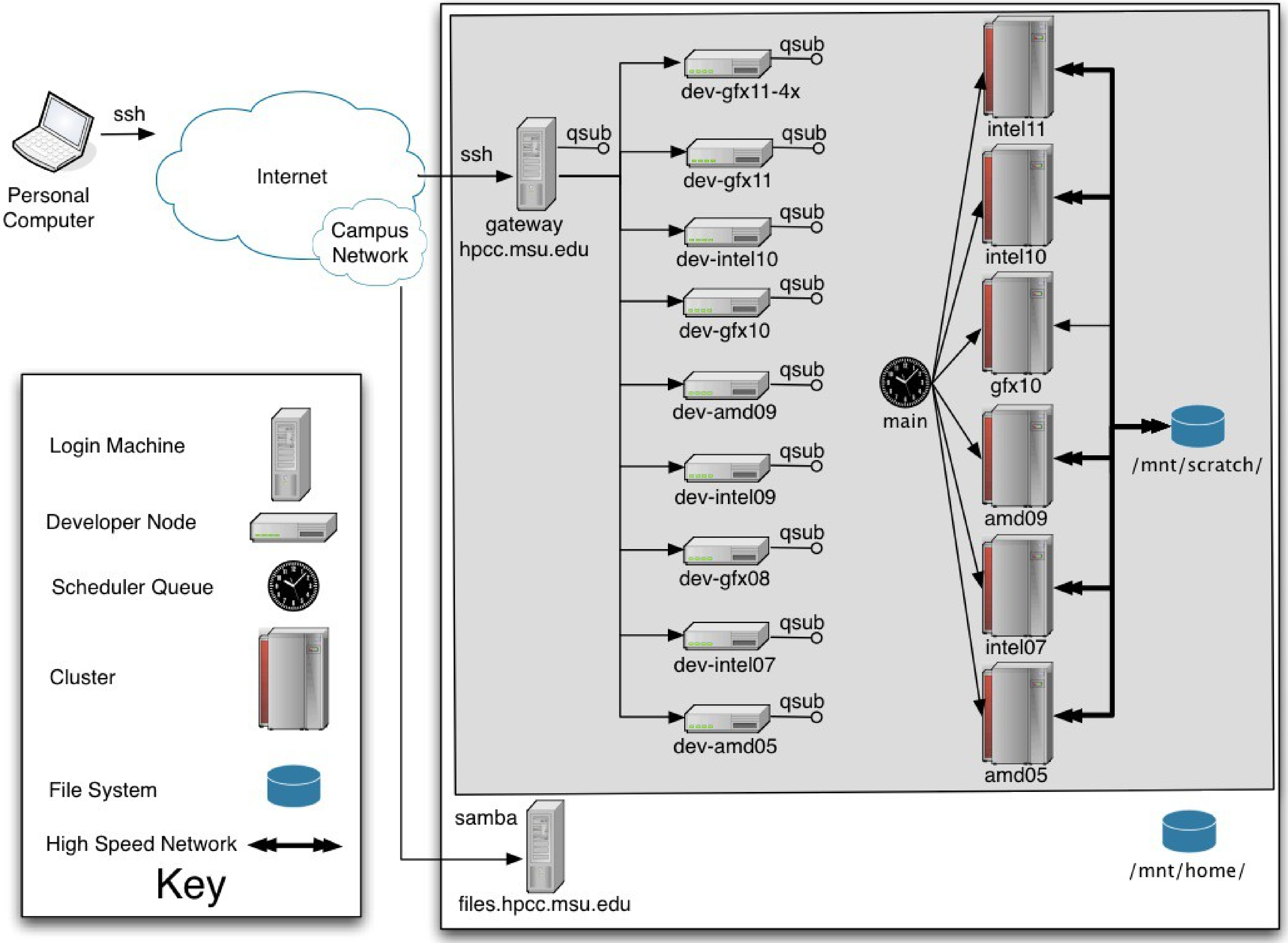
1. Get an account
2. Install needed software (SSH, SCP, X11)
3. **Transfer input files and source code**
4. Compile/Test programs on a developer node
5. Write a submission script
6. Submit the job
7. Get your results and write a paper!!

SCP/SFTP – Secure File transfer

- WinSCP for Windows
  - https://wiki.hpcc.msu.edu/x/Y4nh
- Command-line “scp” and “sftp”on Apple/Linux
- Many other scp and sftp clients out there as well
- Functions over SSHv2 protocol, very secure

Steps in Using the HPCC

1. Get an account
2. Install needed software (SSH, SCP, X11)
3. Transfer input files and source code
4. **Compile/Test programs on a developer node**
5. Write a submission script
6. Submit the job
7. Get your results and write a paper!!


Personal Computer
ssh
Internet
Campus Network
ssh
gateway
hpcc.msu.edu
qsub
dev-gfx11-4x
qsub
dev-gfx11
qsub
dev-intel10
qsub
dev-gfx10
qsub
dev-amd09
qsub
dev-intel09
qsub
dev-gfx08
qsub
dev-intel07
qsub
dev-amd05
qsub
main
intel11
intel10
gfx10
amd09
intel07
amd05
/mnt/scratch/
samba
files.hpcc.msu.edu
/mnt/home/
Login Machine
Developer Node
Scheduler Queue
Cluster
File System
High Speed Network
Key

Advantages of running Interactively

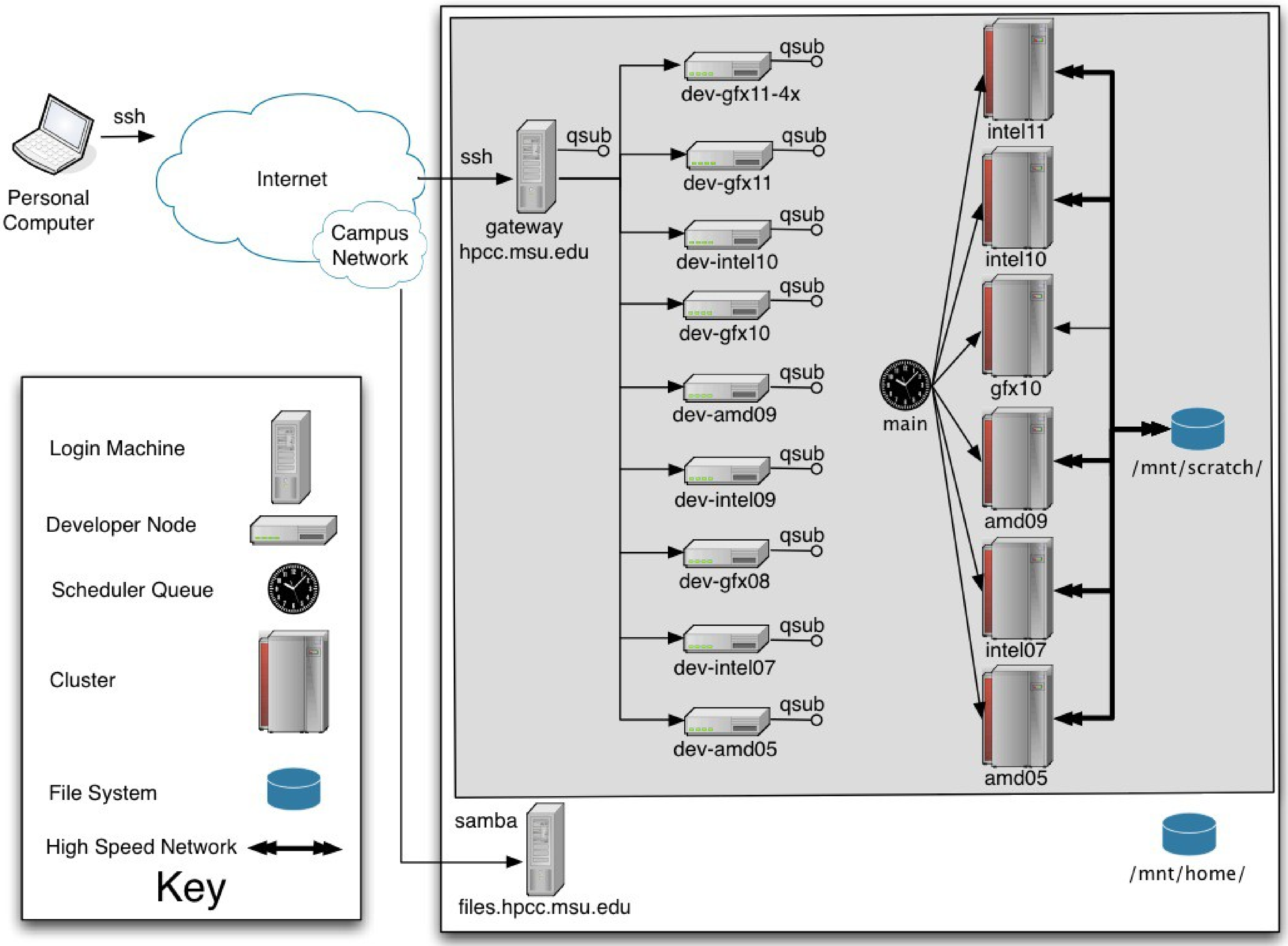
- You do not need to write a submission script
- You do not need to wait in the queue
- You can provide input to and get feedback from your programs as they are running

Disadvantages of running Interactively

- All the resources on Interactive nodes are shared between all users.
- Any single process is limited to 2 hours of cpu time. If a process runs longer than 2 hours it will be killed.
- Programs that overutilize the resources on an integrative node (preventing other to use the system) can be killed without warning.

Steps in Using the HPCC

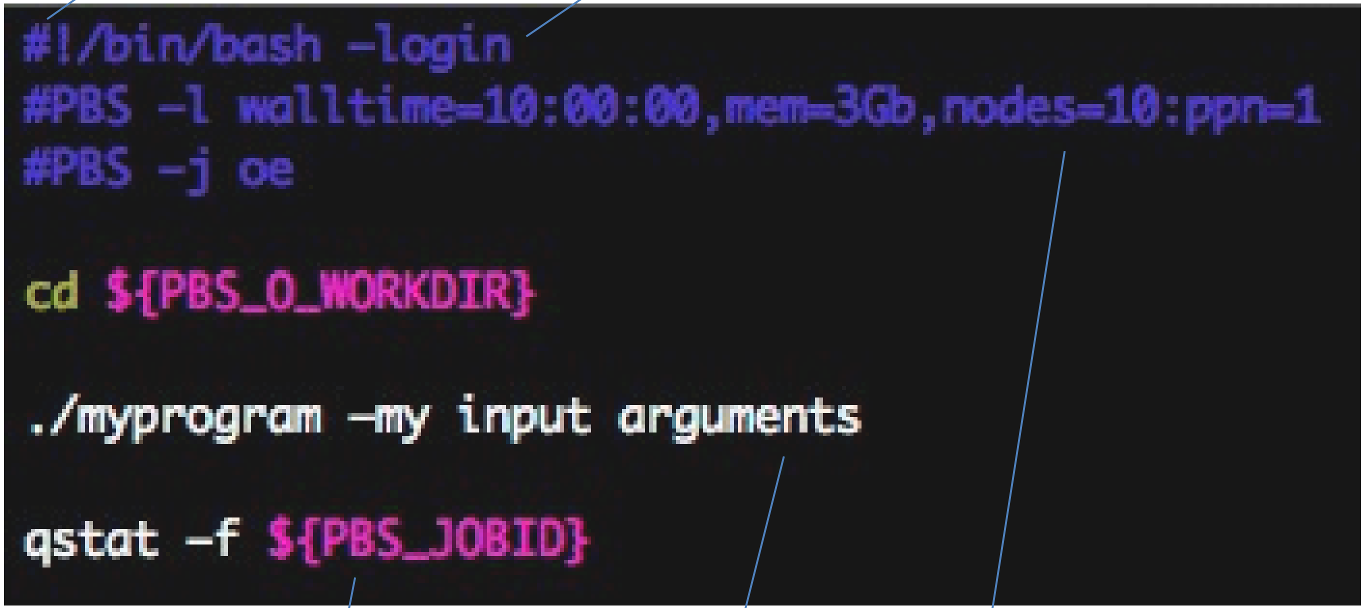
1. Get an account
2. Install needed software (SSH, SCP, X11)
3. Transfer input files and source code
4. Compile/Test programs on a developer node
5. **Write a submission script**
6. Submit the job
7. Get your results and write a paper!!


Personal Computer
ssh
Internet
Campus Network
ssh
gateway
hpcc.msu.edu
qsub
qsub
dev-gfx11-4x
qsub
dev-gfx11
qsub
dev-intel10
qsub
dev-gfx10
qsub
dev-amd09
qsub
dev-intel09
qsub
dev-gfx08
qsub
dev-intel07
qsub
dev-amd05
main
intel11
intel10
gfx10
amd09
intel07
amd05
/mnt/scratch/
samba
files.hpcc.msu.edu
/mnt/home/
Login Machine
Developer Node
Scheduler Queue
Cluster
File System
High Speed Network
Key

Submission Script

1. List of required resources
2. All command line instructions needed to run the computation

Typical Submission Script



```
#!/bin/bash -login
#PBS -l walltime=10:00:00,mem=3Gb,nodes=10:ppn=1
#PBS -j oe

cd ${PBS_O_WORKDIR}

./myprogram -my input arguments

qstat -f ${PBS_JOBID}
```

Shell Comment

Define Shell

Resource Requests

Shell Commands

Special Environment Variables

Example: Submit a job

- Go to the top helloworld directory
  - `cd ~/helloworld`
- Look at the simple submission script
  - `nano hello.qsub`
- Nano is a simple program you can use to edit text files on the HPCC
- See bottom line of nano for commands the "^" character indicates the "control" key

hello.qsub

```
#!/bin/bash -login
#PBS -l walltime=00:10:00
#PBS -l nodes=1:ppn=1,feature=gbe

cd ${PBS_O_WORKDIR}

./hello

qstat -f ${PBS_JOBID}
```

MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6QFiAQ

ICER

Steps in Using the HPCC

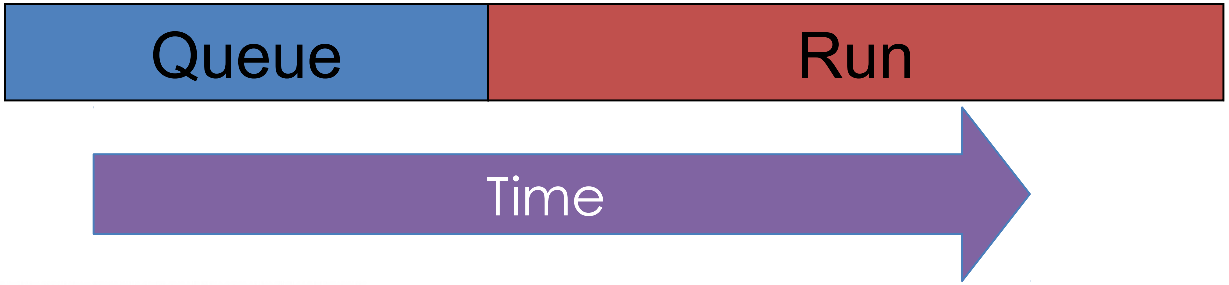
1. Get an account
2. Install needed software (SSH, SCP, X11)
3. Transfer input files and source code
4. Compile/Test programs on a developer node
5. Write a submission script
6. **Submit the job**
7. Get your results and write a paper!!

Common Queue Commands

- **qsub** <Submission script>
  - Submit a job to the queue
- **qdel** <JOB ID>
  - Delete a job from the queue
- **showq** –u <USERNAME>
  - Show the current job queue
- **checkjob** <JOB ID>
  - Check the status of the current job
- **showstart** –e all <JOB ID>
  - Show the estimated start time of the job.

Submitting a job

- qsub –arguments <Submission Script>
  - Returns the job ID. Typically looks like the following:
    - 5945571.cmgr01
- Time to job completion



https://wiki.hpcc.msu.edu/x/6QFiAQ

Example: Submit a job, cont.

- Submit the file to the queue

  -
    ```
    qsub hello.qsub
    ```
- Record jobid number (######) and wait at most 30 seconds
- Check the status of the queue

  -
    ```
    showq -u ${USER}
    ```

Example: Monitor a job

- Get the status of the job:
  - `qstat -f ######`
- When will a job start:
  - `showstart -e all ######`

Scheduling Priorities

- Jobs that use more resources get higher priority (because these are hard to schedule)
- Smaller jobs are backfilled to fit in the holes created by the bigger jobs
- Eligible jobs acquire more priority as they sit in the queue
- Jobs can be in three basic states:
  - Blocked, eligible or running

Current Cluster Resources

Year	Name	Description	ppn	Memory	Nodes	Total Cores
2007	intel07	Quad-core 2.3GHz Intel Xeon E5345	8	8GB	126	1008
2010	gfx10	NVIDIA CUDA Node (no IB)	8	18GB	32	256
2010	intel10	Intel Xeon E5620 (2.40 GHz)	8	24GB	191	1528
2011	intel11	Intel Xeon 2.66 GHz E7-8837	32	512GB	2	64
			32	1TB	1	32
			64	2TB	2	128
2014	intel14	Intel Xeon E5-2670 v2 (2.6 GHz)	20	64GB	128	2560
			20	256GB	24	480
		2 NVIDIA K20 GPUs	20	128GB	40	800
		2 Xeon Phi 5110P	20	128GB	28	560
2014	Intel14-XL	Intel Xeon E7-8857 v2 (3 GHz)	48	1-3TB	5	240
			96	6 TB	1	96
	Total				580	7752

Job completion

- By default the job will automatically generate two files when it completes:
  - Standard Output:
    - Ex: jobname.o5945571
  - Standard Error:
    - Ex: jobname.e5945571
- You can combine these files if you add the join option in your submission script:
  - “#PBS -j oe”
- You can change the output file name
  - #PBS -o /mnt/home/netid/myoutputfile.txt

https://wiki.hpcc.msu.edu/x/6QFiAQ

Other Job Properties

- resources (-l)
  - Walltime, memory, nodes, processor, network, etc.
- #PBS –l feature=gpgpu,gbe
- #PBS –l nodes=2:ppn=8:gpu=2
- #PBS –l mem=16gb
- Email address (-M)
  - Ex: #PBS –M colbrydi@msu.edu

- Email Options (-m)
  - Ex: #PBS –m abe

Many others, see the wiki: http://wiki.hpcc.msu.edu/

Steps in Using the HPCC

1. Get an account
2. Install needed software (SSH, SCP, X11)
3. Transfer input files and source code
4. Compile/Test programs on a developer node
5. Write a submission script
6. Submit the job
7. **Get your results and write a paper!!**

Agenda

- What is iCER / HPCC
- Common classes problems
- Overview of Hardware
- Getting started, Seven Steps to High Performance
- Running in parallel
- Tips and tricks

What is the Bottleneck

- Not enough Memory
  - Solution: use a bigger node (6tb 96 cores)
- Slow File I/O
  - Solution: use scratch
  - Solution: use a ram disk
- Too many calculations
  - Solution: run your code in parallel

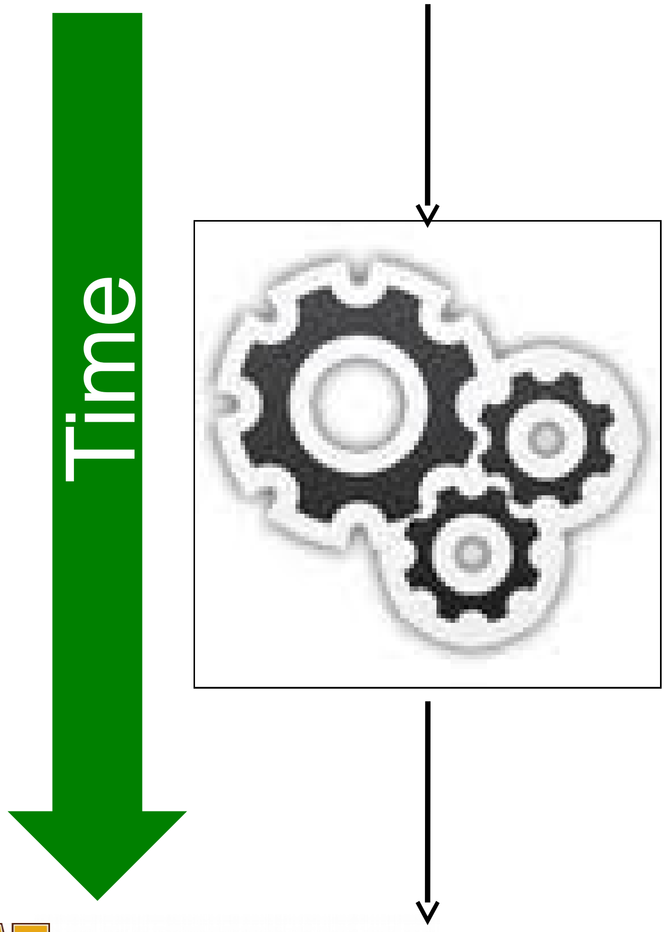
Steps to parallel code

Note: Every application is different

1. Analyze your code
   - Profilers (gprof, vtune, map, perfreport, tau)
   - Debuggers / memory trackers (gdb, ddt, totalview)
2. Optimize calculations
   - Trade memory for time (i.e., never do the same calculation twice)
3. Find ways to parallelize
   - Look for loops
   - Find iterations independent from each other
   - Determine how much information needs to be transferred

https://wiki.hpcc.msu.edu/x/6QFiAQ

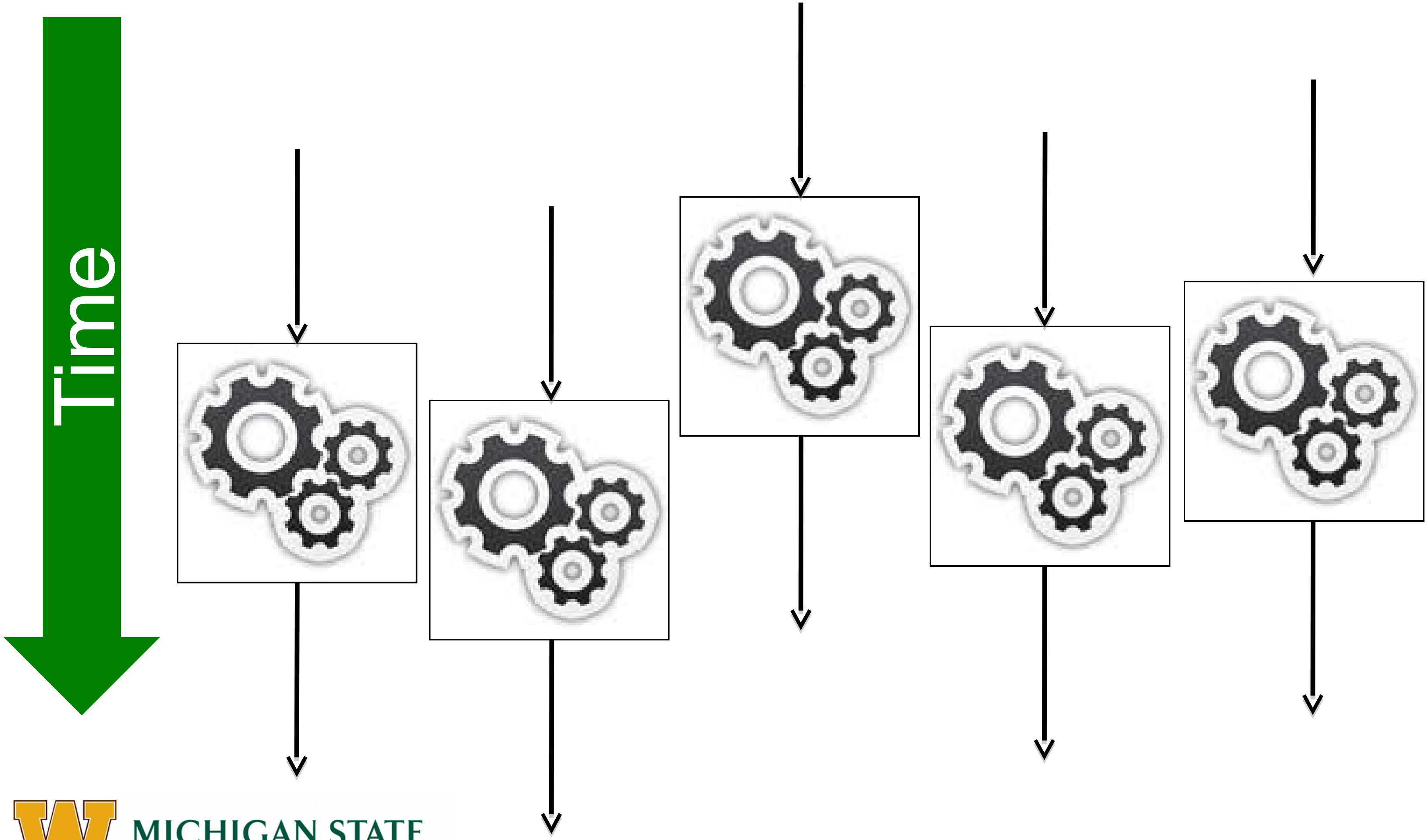
Single Thread Jobs



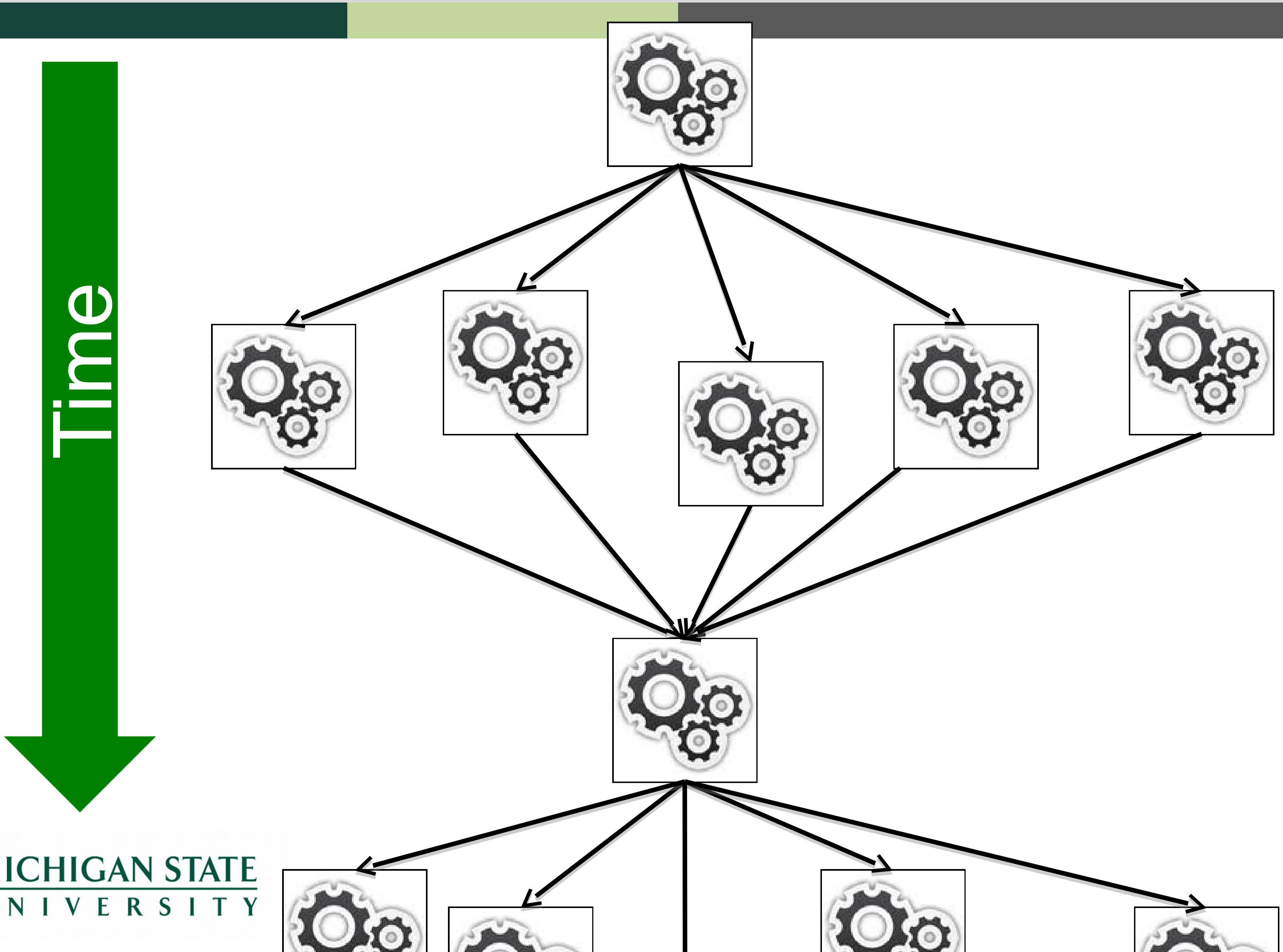
One CPU can only run one thing at a time. (sort of)

https://wiki.hpcc.msu.edu/x/6QFiAQ

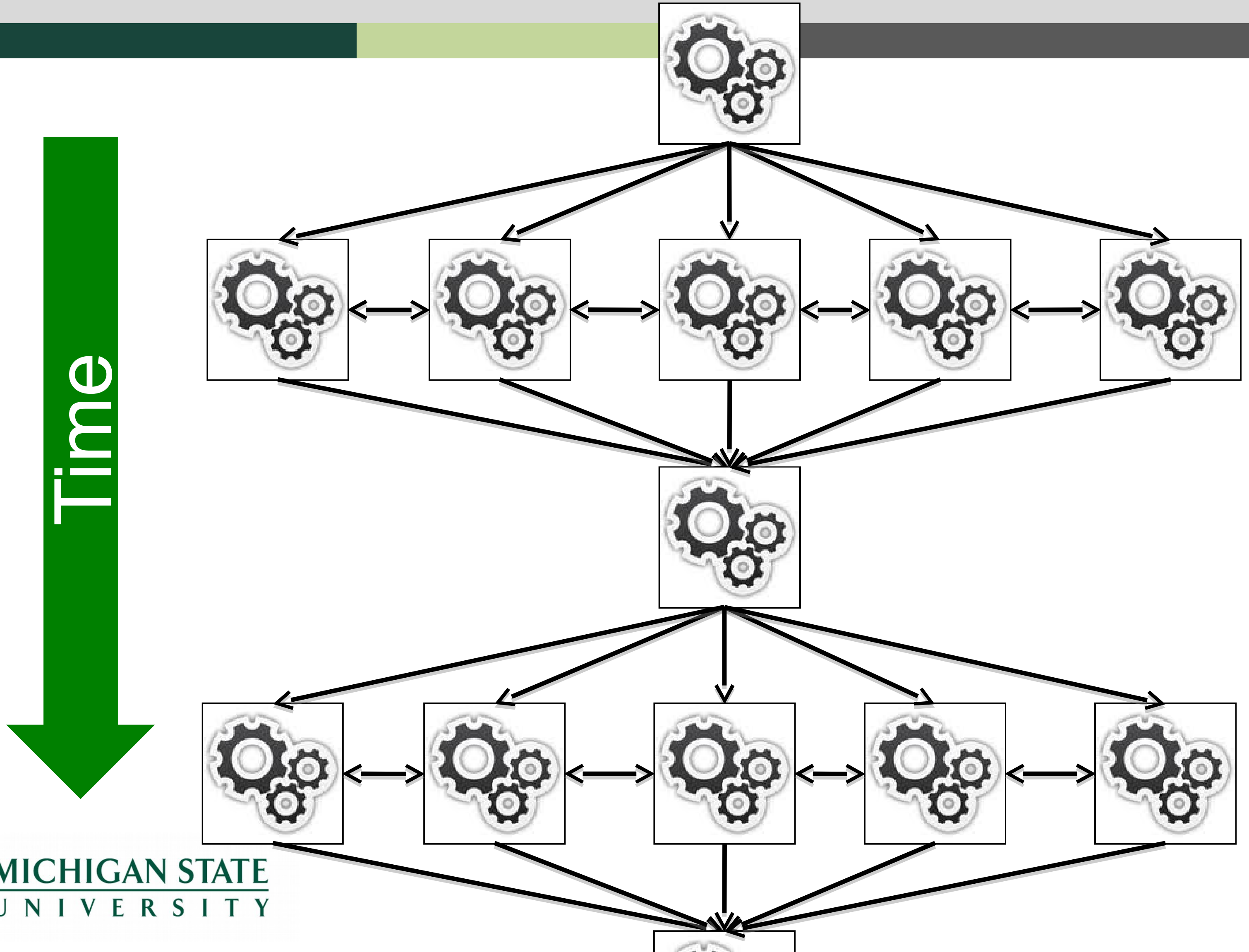
Pleasantly Parallel



Loosely Coupled



Tightly Coupled

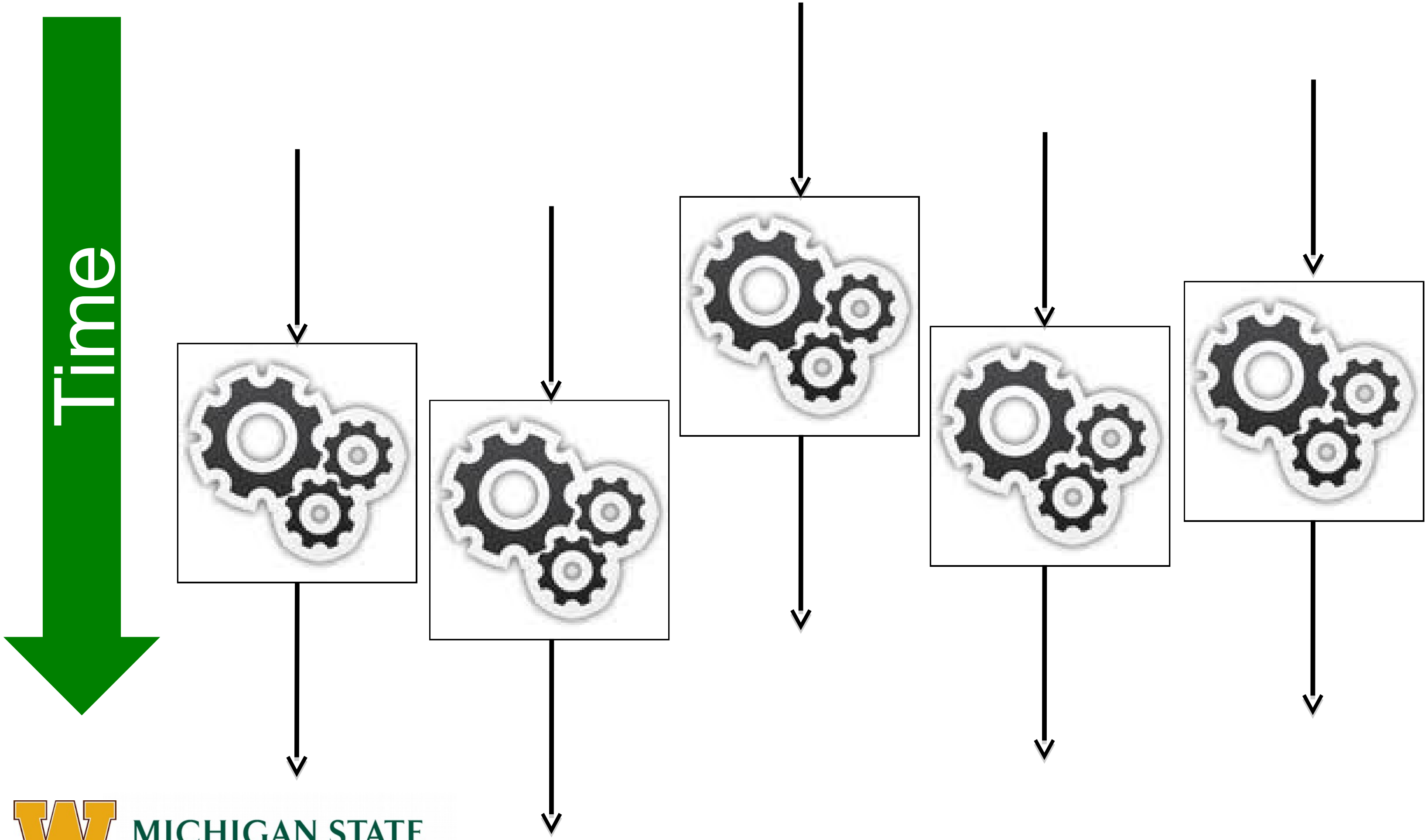


Communication

- Shared Memory
- Shared Network
- Distributed Network
- Dedicated Acceleratc
- Hybrid Systems

Pleasantly Parallel

Pleasantly Parallel



https://wiki.hpcc.msu.edu/x/6QFiAQ

How fast can we go?

- T - How long does each operation take?
- N - How many operations do you need to run?
- CPUs – Number of Cores job will run on.

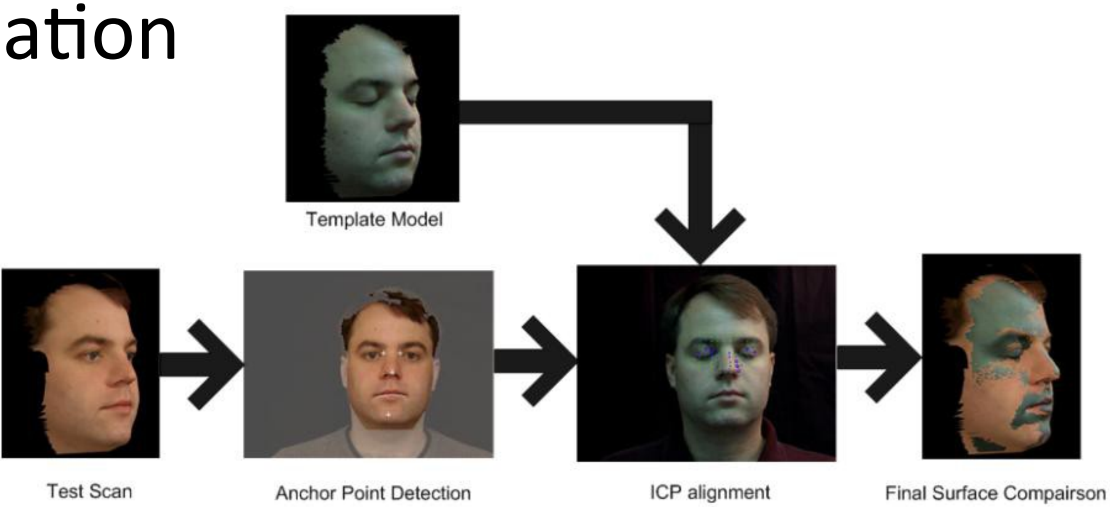
- Single CPU time estimate:
  - TxN
- Best possible Pleasantly parallel time:
  - (TxN)*overhead/CPUs

Who are you? -- Biometrics

https://wiki.hpcc.msu.edu/x/6QFiAQ

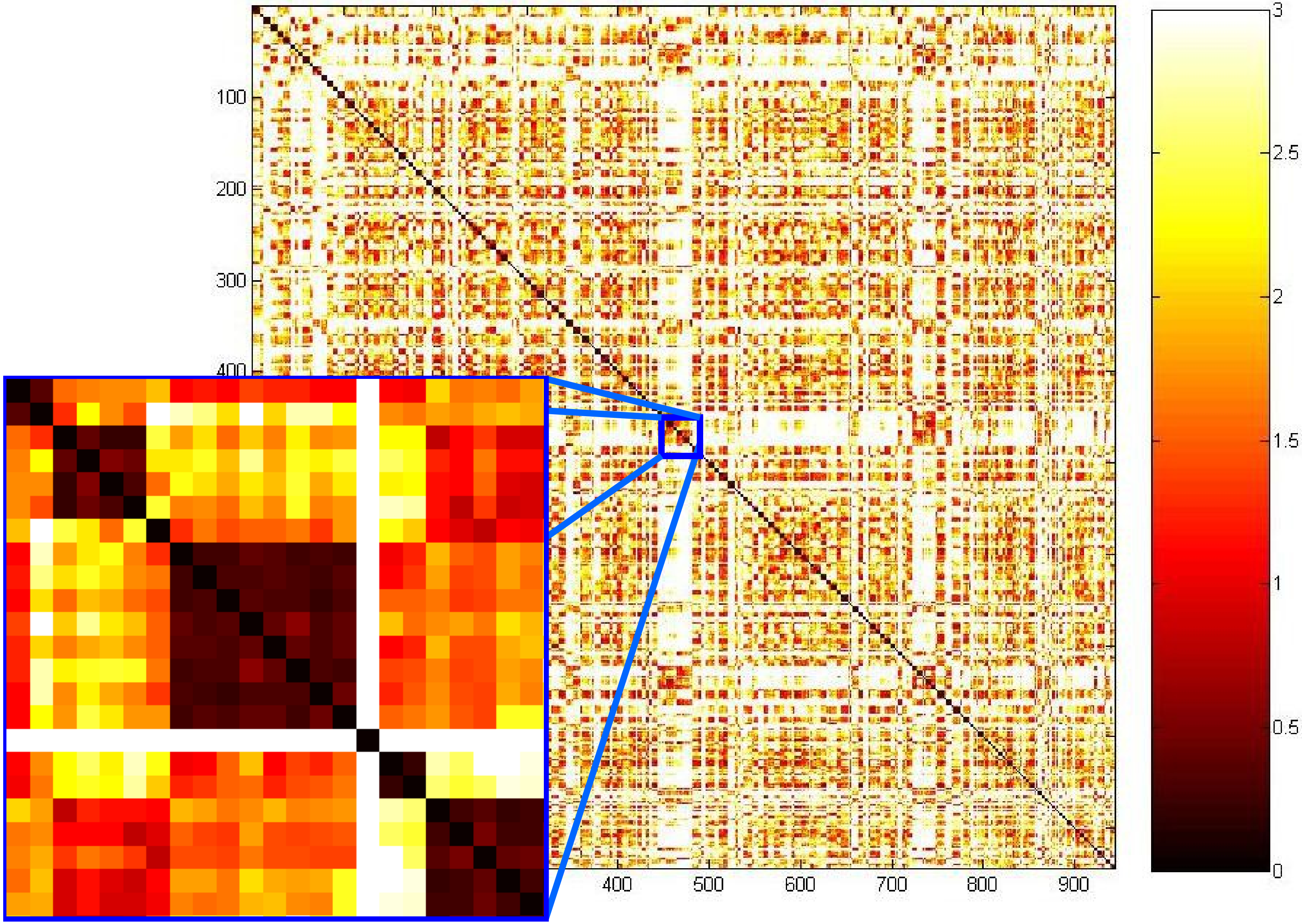
Pairwise-All Problem

- Database of faces
- Compare everything to everything else
- Calculate a Matching score to use for identification



https://wiki.hpcc.msu.edu/x/6QFiAQ

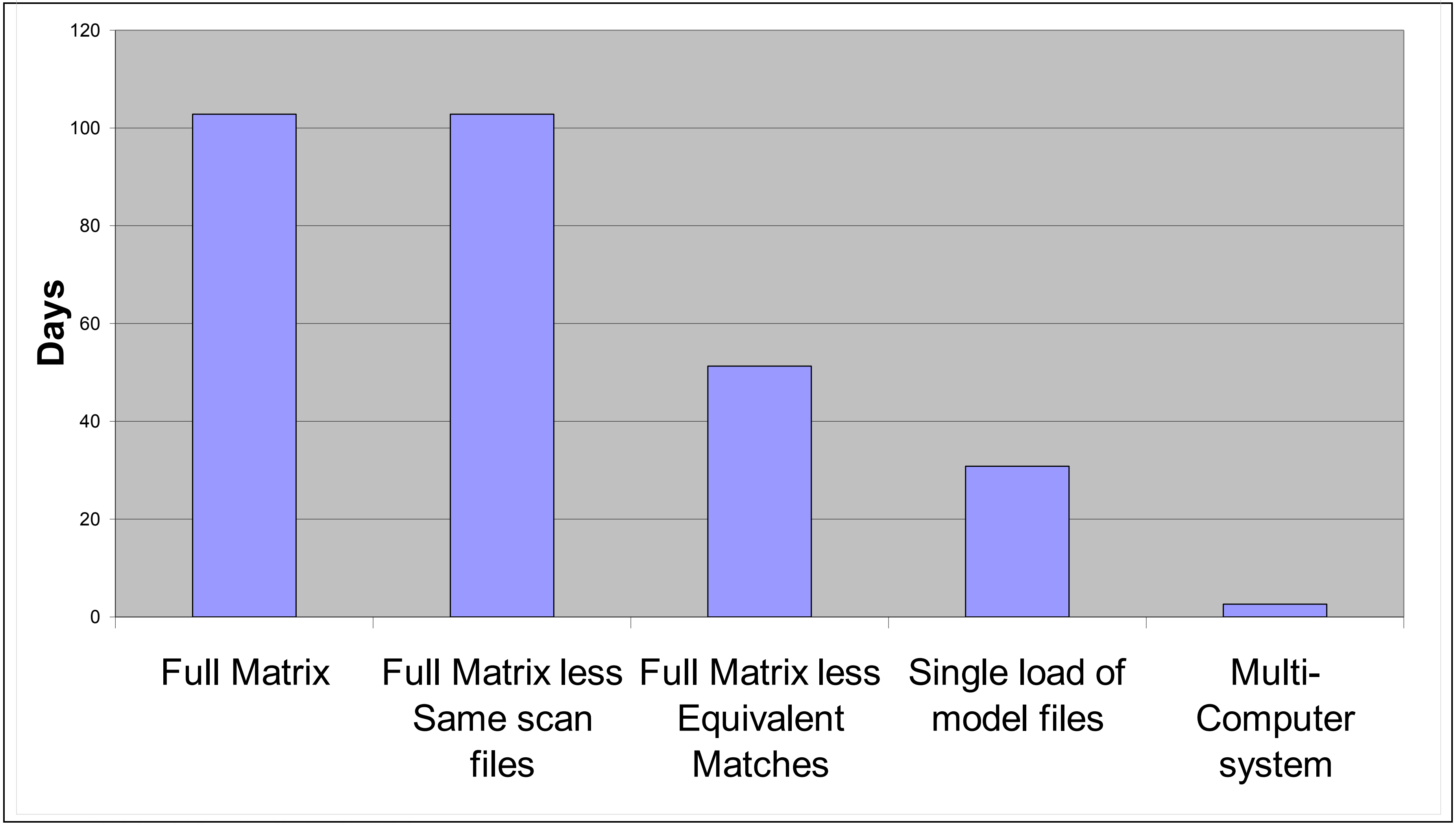
943 x 943 Similarity Matrix



Estimated Calculation Times

- Preprocessing
  - 943 * 12 (seconds) ≈ 189 Minutes
- Matching
  - 943 * 943 * 5 (seconds) ≈ 103 Days
- Scans matched to themselves always result in 0 mm
  - (943 * 943 – 943) * 5 (seconds) ≈ 103 Days
- The Proposed Alignment Algorithm is symmetric.
  - (943 * 943 – 943)/2 * 5 (seconds) ≈ 51.5 Days
- We also load models once per row instead of every time
  - (943*943-943)/2 * 3 (seconds) + 943 * 2 (seconds) ≈ 31 Days

Calculation Time for Full Similarity Matrix



How do we go even bigger?

- 5000 scans.
  - 1.5 years on a single processor computer
  - 13 days on our ad-hoc cluster.
  - 1.5 days a commodity cluster at MSU

Steps to Pleasantly Parallel

- Figure out command line
- Estimate single job time:
  - Should be > 5 minutes
  - Should be < 1 week
  - Best if < 4 hours
- Make a submissions script
- Submit Job

Pleasantly Parallel Example

- Folder full of input files:

1.in	5.in	9.in	13.in	17.in
2.in	6.in	10.in	14.in	18.in
3.in	7.in	11.in	15.in	19.in
4.in	8.in	12.in	16.in	

- Want folder full of output files:

1.out	5.out	9.out	13.out	17.out
2.out	6.out	10.out	14.out	18.out
3.out	7.out	11.out	15.out	19.out
4.out	8.out	12.out	16.out	

- Command Syntax:

```
./myprogram inputfile > outputfile
```

https://wiki.hpcc.msu.edu/x/6QFiAQ

MICHIGAN STATE UNIVERSITY

ICER

PBS Job Arrays

- One submission script copied many times
- Uses the PBS –t option
  - Ranges: 1-10
  - Lists: 2,4,100,3
  - Combination: 1-10,20,50,100
- Distinguish between jobs by using the PBS_ARRAYID environment variable

https://wiki.hpcc.msu.edu/x/6QFiAQ

Simple Job Array

```
#!/bin/bash -login
#PBS -l walltime=00:05:00,mem=2gb
#PBS -l nodes=1:ppn=1,feature=gbe
#PBS -t 1-19

cd ${PBS_O_WORKDIR}

mkdir ${PBS_ARRAYID}
Cd ${PBS_ARRAYID}


../myprogram ../${PBS_ARRAYID}.in > ${PBS_ARRAYID}.out

qstat -f ${PBS_JOBID}
```

Example: Job Arrays

- Get the bleder_farm example:
  - `getexample`
  - `getexample blender_farm`
  - `cd ./blender_farm`
- Look at the qusb file, using “less” command
  - `less blender_farm.qsub`
- Submit the job
  - `qsub blender_farm.qsub`

https://wiki.hpcc.msu.edu/x/6QFiAQ

HPCC Job array limitations

- Can not have more than 520 cores running at once
- Can not submit more than 1000 jobs at once
- Each job can not run longer than one week

- Lots of ways to work around these limitations

https://wiki.hpcc.msu.edu/x/6QFiAQ

Job array numbers

- All numbers in a job array have the same base number
  - 7478210
- Each PBS_ARRAYID is show in square brackets
  - 7478210[1]
  - 7478210[2]
- Delete all jobs using one command
  - qdel 7478210[]

https://wiki.hpcc.msu.edu/x/6QFiAQ

Files as Semaphores (FAS)

- Use a list of input files as your task list
- Use a list of output files (or flag files) as your in-progress/complete list
- Rely on the file system to ensure that no two jobs are selected at the same time (not a great assumption but it works)

Simple FAS

```
#!/bin/bash -login
#PBS -l walltime=00:05:00
#PBS -l nodes=1:ppn=1,feature=gbe
#PBS -t 1-100
cd ${PBS_O_WORKID}
sleep $(( ${RANDOM} % 100 ))

for file in *.in; do
  output=“./${file%.*}.out”
  if [ ! -f ${output} ]; then
    touch ${output}
    ./myprogram ${file} > ${output}
    qsub -t 0 -N ${PBS_JOBNAME} ${0}
    exit 0
  fi
```

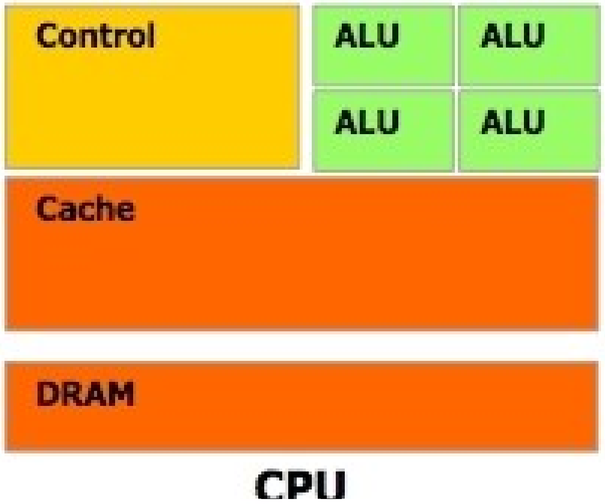
Shared Memory Parallelization

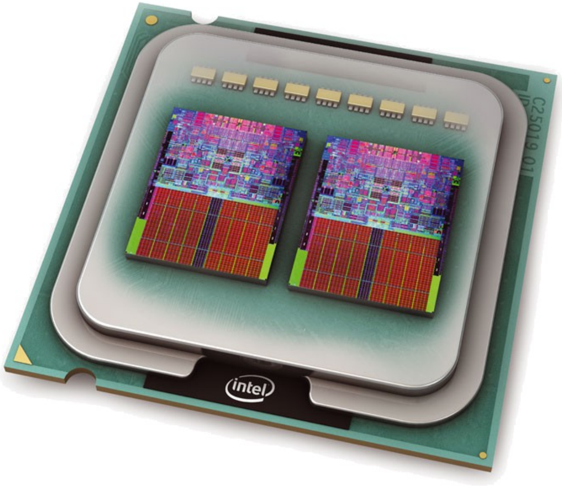
Shared Memory

- Different threads (cores, processes) communicate though pointers to the same memory location
- Problems can occur if different threads write the same memory at the same time
- Flags (also called locks and/or semaphores) are used to allow only one thread to access memory at the same time

Shared Memory Communication

- Cores on a processor share the same memory
- OpenMP
- Fat nodes
  - 96 cores
  - 6TB of memory

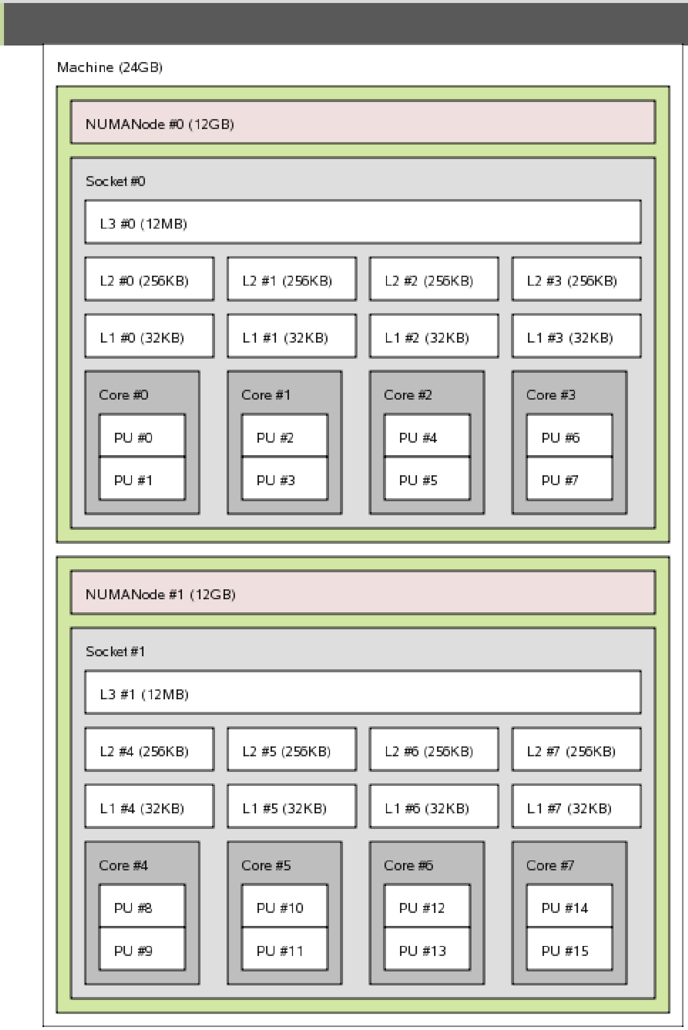




https://wiki.hpcc.msu.edu/x/

Intel10

- 8 cores
- 24 GB memory



Machine (24GB)

NUMANode #0 (12GB)

Socket #0

L3 #0 (12MB)

L2 #0 (256KB) | L2 #1 (256KB) | L2 #2 (256KB) | L2 #3 (256KB)

L1 #0 (32KB) | L1 #1 (32KB) | L1 #2 (32KB) | L1 #3 (32KB)

Core #0: PU #0, PU #1 | Core #1: PU #2, PU #3 | Core #2: PU #4, PU #5 | Core #3: PU #6, PU #7

NUMANode #1 (12GB)

Socket #1

L3 #1 (12MB)

L2 #4 (256KB) | L2 #5 (256KB) | L2 #6 (256KB) | L2 #7 (256KB)

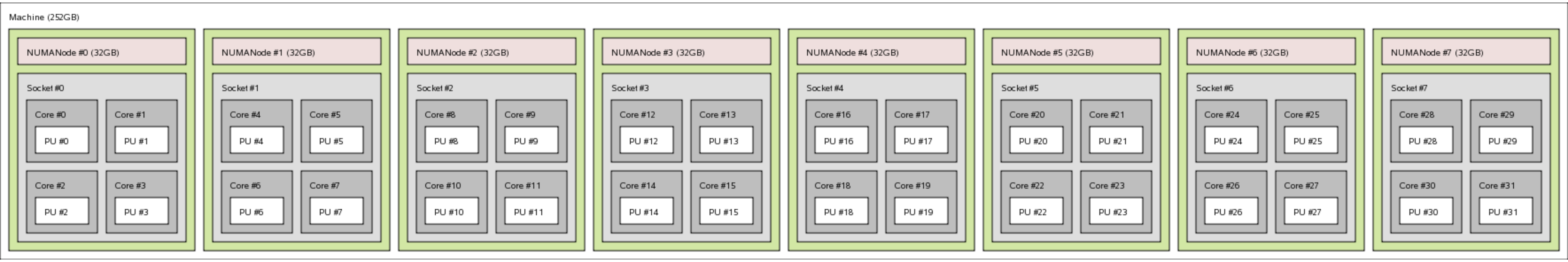
L1 #4 (32KB) | L1 #5 (32KB) | L1 #6 (32KB) | L1 #7 (32KB)

Core #4: PU #8, PU #9 | Core #5: PU #10, PU #11 | Core #6: PU #12, PU #13 | Core #7: PU #14, PU #15

https://wiki.hpcc.msu.edu/x/6QFiAQ

Large Memory Example

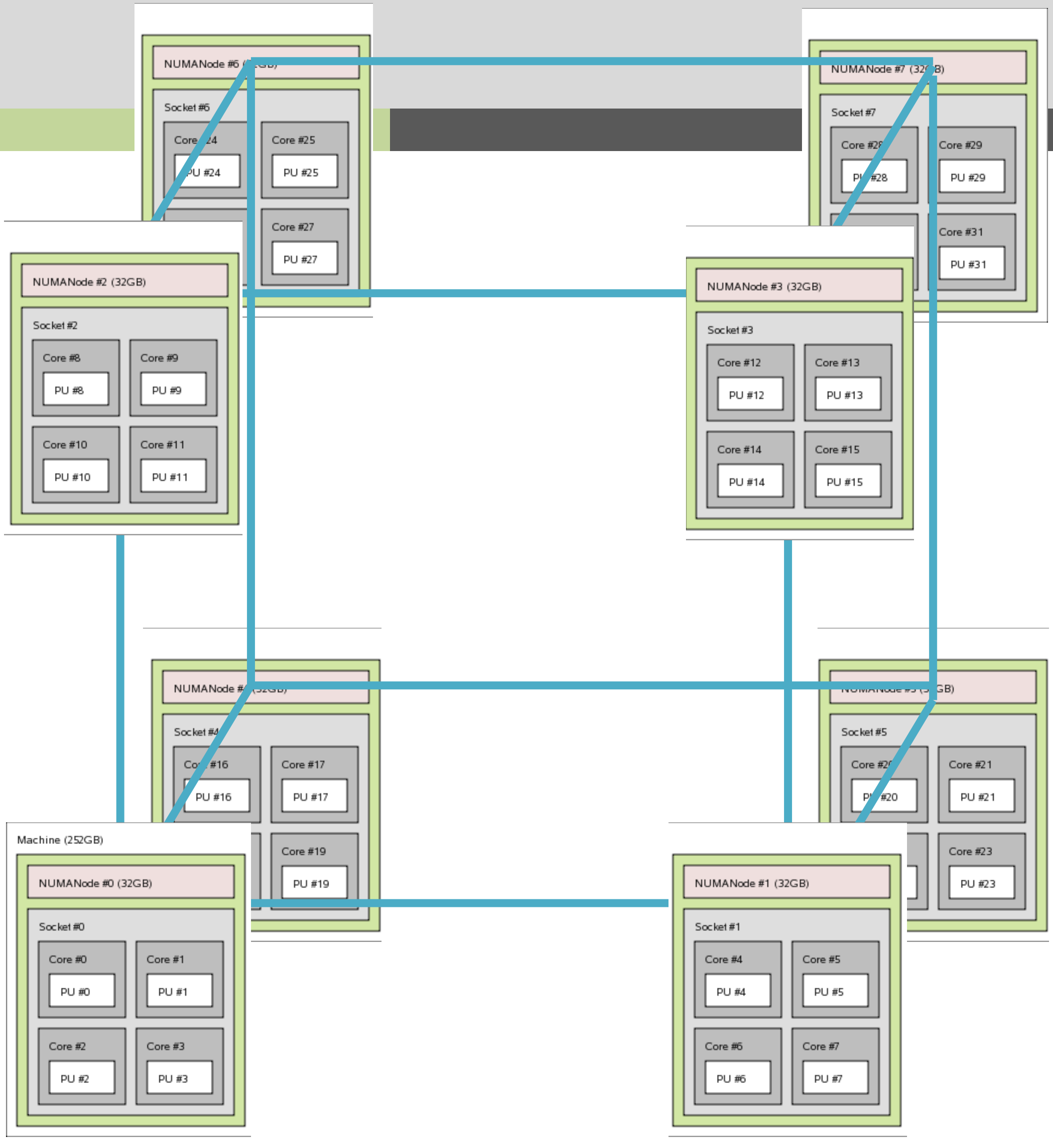
- 32 cores
- 256 GB memory



- We also have nodes with up to 64 cores and 2TB of memory

https://wiki.hpcc.msu.edu/x/6QFiAQ

NUMA



https://wiki.hpcc.msu.edu/x/6QFiAQ

Shared memory submission scripts

- Typically one node with multiple processors per node (ppn)
  - #PBS –l nodes=1:ppn=**8**
- Different programs use different methods to tell them how many processors to use
  - Command line arguments
  - Environment variables

Example: shared memory Script

- Bowtie uses shared memory parallelization
- Get the bowtie example
  - `getexample bowtie`
- Change to the bowtie directory
  - `cd ./bowtie`
- Look at the submission script
  - `less ./bowtie.qsub`
- Run the job
  - `qsub bowtie.qsub`

OpenMP

- Common Shared Memory parallelizaiton
- Single program runs in many threads
- Really easy to pick loops that are parallel and split them into multi threads
- Minor modifications to code that can be written not to affect single

OpenMP is easy

```
#include <omp.h>

...
  #pragma omp parallel for
  for (i=0;i<100;++i) {
    A(i) = A(i) + B
  }

...
```

Compile OpenMP Jobs

- Use compiler option fopenmp.
  - fopenmp
- Example:

```
gcc –fopenmp mycode.cc –o mycode
```

simpleOMP.qsub example

```
#!/bin/bash -login
#PBS -l walltime=00:01:00
#PBS -l nodes=1:ppn=5,feature=gbe

cd ${PBS_O_WORKDIR}
export OMP_NUM_THREADS=${PBS_NUM_PPN}

./simpleOMP

qstat -f ${PBS_JOBID}
```

Try another getexample

```
getexample helloOpenMP
getexample OpenMP_profiling
```

Shared Network Parallelization

https://wiki.hpcc.msu.edu/x/6QFiAQ

MPI program (1 of 4)

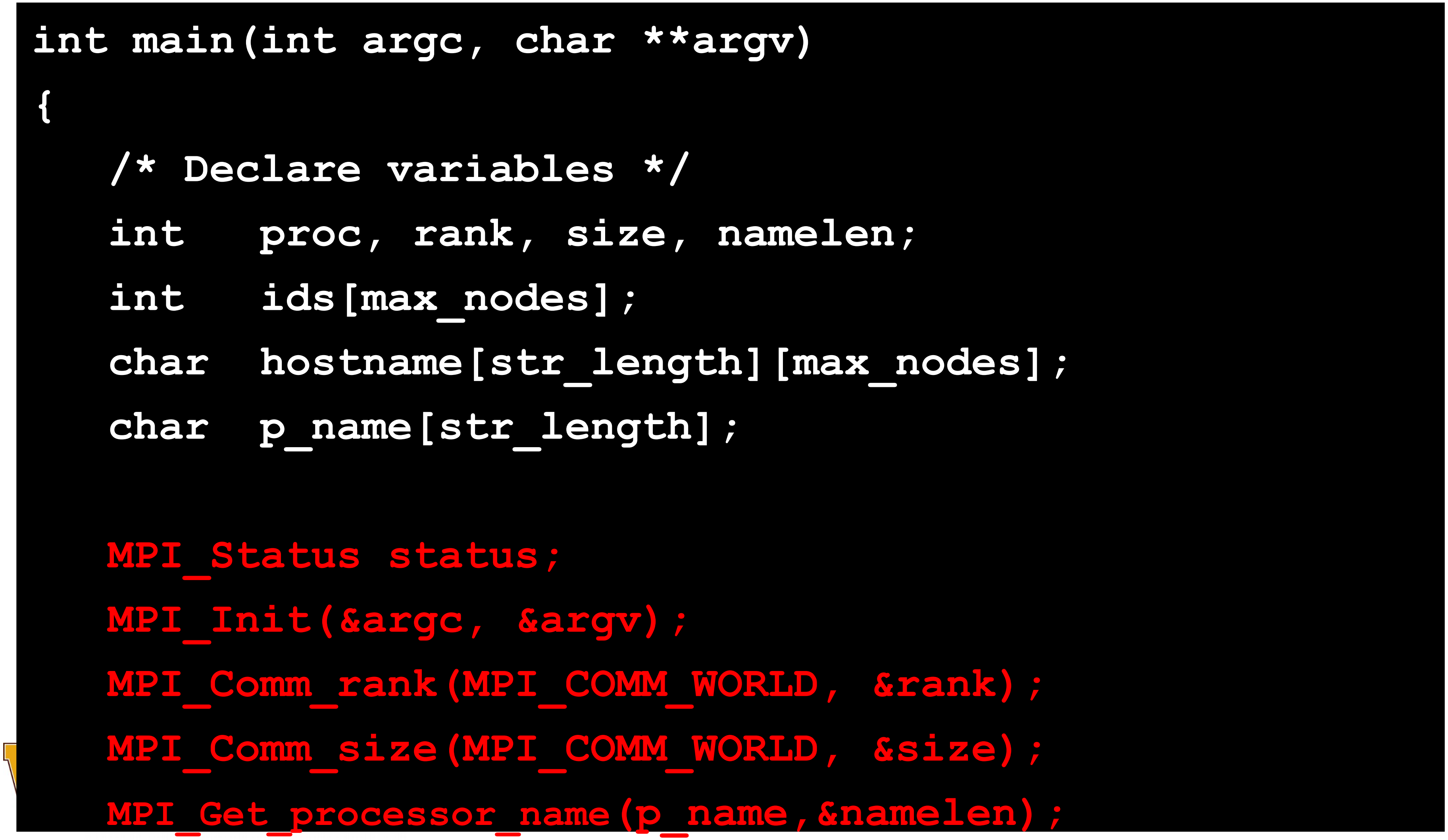
```
/* Needed for printf'ing */
#include <stdio.h>
#include <stdlib.h>

/* Get the MPI header file */
#include <mpi.h>

/* Max number of nodes to test */
#define max_nodes 264

/* Largest hostname string hostnames */
#define str_length 50
```

MPI program (2 of 4)



```
int main(int argc, char **argv)
{
   /* Declare variables */
   int    proc, rank, size, namelen;
   int    ids[max_nodes];
   char   hostname[str_length][max_nodes];
   char   p_name[str_length];

   MPI_Status status;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Get_processor_name(p_name,&namelen);
```

MPI program (3 of 4)

```
if (rank==0) {
   printf("Hello From: %s I am the receiving processor
%d of %d\n",p_name, rank+1, size);
   for (proc=1;proc<size;proc++) {
      MPI_Recv(&hostname[0][proc], \\
               str_length,MPI_INT,proc, \\
               1,MPI_COMM_WORLD,&status);
      MPI_Recv(&ids[proc], \\
               str_length,MPI_INT,proc, \\
               2,MPI_COMM_WORLD,&status);
      printf("Hello From: %-20s I am processor %d of
%d\n",&hostname[0][proc], ids[proc]+1, size);
   }
```

MPI program (4 of 4)

```
} else { // NOT Rank 0
      srand(rank);
      int t = rand()%10+1;
      sleep(t);
      MPI_Send(&p_name,str_length, \\
               MPI_INT,0,1,MPI_COMM_WORLD);
      MPI_Send(&rank,str_length, \\
               MPI_INT,0,2,MPI_COMM_WORLD);
   }
   MPI_Finalize();

   return(0);
}
```

Trying out an example

1. Log on to one of the developer nodes
2. Load the powertools module:

   -
     ```
     module load powertools
     ```
3. Run the getexample program. This will create a folder called helloMPI:

   -
     ```
     getexample helloMPI
     ```
4. Change to the helloMPI directory and read the readme files
5. Or just type the following on the command line:

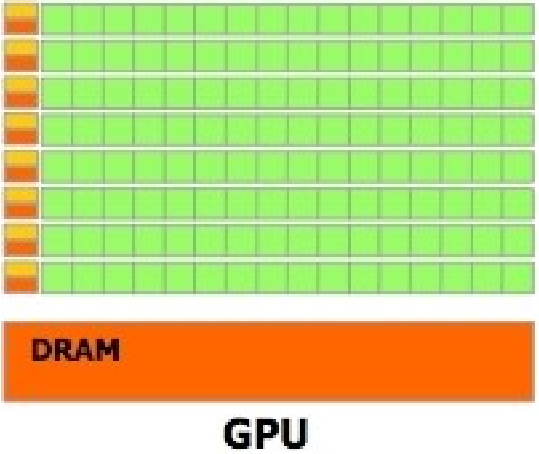
   -
     ```
     ./README
     ```

https://wiki.hpcc.msu.edu/x/6QFiAQ

Accelerator Cards

GPU

- Cards used to render graphics on a computer
- Hundreds of cores
- Not very smart cores
- But, if you can make your research look like graphics rendering you may be able to run really fast!



MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6(

Running on the GPU

- Program Starts on the CPU
  - Copy data to GPU (slow-ish)
  - Run kernel threads on GPU (very fast)
  - Copy results back to CPU (slow-ish)

- There are a lot of clever ways to fully utilize both the GPU and CPU.

Pros and Cons

- Benefits
  - Lots of processing cores.
  - Works with the CPU as a co-processor
  - Very fast local memory bandwidth
  - Large online community of developers

Drawbacks

- Can be difficult to program.
- Memory Transfers between GPU and CPU are costly (time).
- Cores typically run the same code.
- Errors are not detected (on older cards)
- Double precision calculations are slow (On older cards)

CUDA program (1 of 5)

```
#include "cuda.h"
#include <iostream>

using namespace std;

void printGrid(float an_array[16][16]) {
  for (int i = 0; i < 16; i++){
      for (int j = 0; j < 16; j++) {
          cout << an_array[i][j];
      }
      cout << endl;
    }
}
```

https://wiki.hpcc.msu.edu/x/6QFiAQ

CUDA program (2 of 5)

```
__global__ void theKernel(float * our_array)
{
  // This is array flattening,
  //(Array Width * Y Index + X Index)
  our_array[(gridDim.x * blockDim.x) * \\
            (blockIdx.y * blockDim.y + threadIdx.y) + \\
            (blockIdx.x * blockDim.x + threadIdx.x)] = \\
            = 5;
}
```

https://wiki.hpcc.msu.edu/x/6QFiAQ

CUDA program (3 of 5)

```
int main()
{
  float our_array[16][16];

  for (int i = 0; i < 16; i++) {
    for (int j = 0; j < 16; j++) {
      our_array[i][j] = 0;
    }
  }
```

CUDA program (4 of 5)

```
//STEP 1: ALLOCATE
float * our_array_d;
int size = sizeof(float)*256;
cudaMalloc((void **) &our_array_d, size);

//STEP 2: TRANSFER
cudaMemcpy(our_array_d, our_array, size, \\
            cudaMemcpyHostToDevice);
```

MICHIGAN STATE UNIVERSITY

https://wiki.hpcc.msu.edu/x/6QFiAQ

ICER

CUDA program (5 of 5)

```
  //STEP 3: SET UP
  dim3 blockSize(8,8,1);
  dim3 gridSize(2,2,1);

  //STEP 4: RUN
  theKernel<<<gridSize, blockSize>>>(our_array_d);

  //STEP 5: TRANSFER
  printGrid(our_array);
  cudaMemcpy(our_array, our_array_d, size, \\
             cudaMemcpyDeviceToHost);
  cout << "--------------------" << endl;
  printGrid(our_array);


}
```

Compile CUDA Jobs

- Just like MPI, to compile an cuda program you need to use the cuda compiler wrappers:
  - nvcc simple.cu -o simple_cuda

Try a cuda getexample

getexample cuda

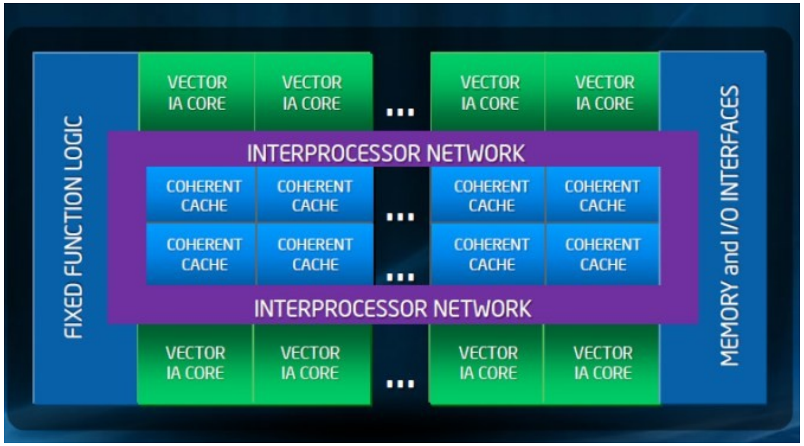
getexample cuda_clock

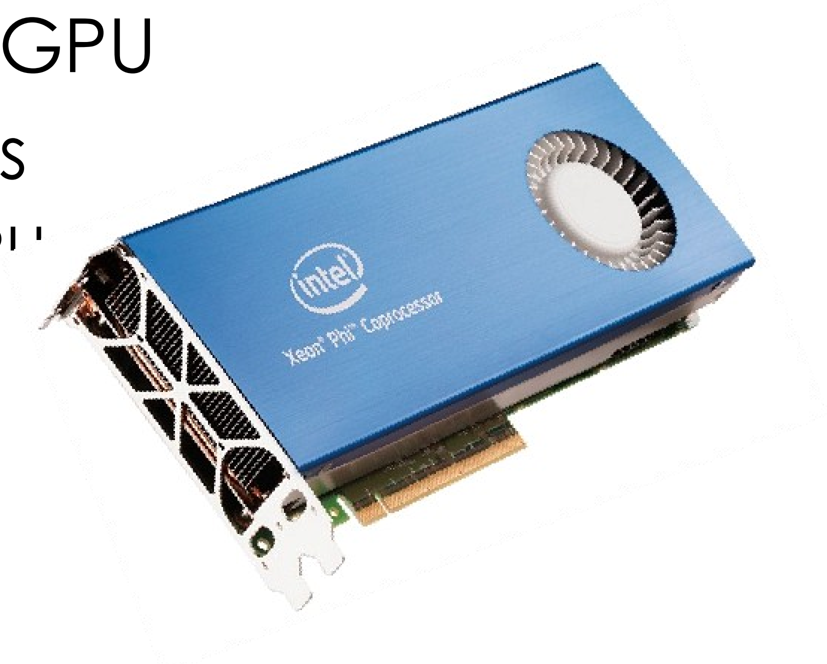
getexample cuda_hybrid

getexample NAMD_CUDA_example

Intel Xeon Phi

- Cross between CPU and GPU
- About 61 Pentium III cores
  - Less cores/slower than GP
  - Easier to use than GP





- MPI
- OpenMP

https://wiki.hpcc.msu.edu/x/6QFiAQ

Try a Phi Card example

```
getexample MIC_examples
getexample MKL_mic
```

Standard Libraries

Standard Libraries

- When possible take advantage of parallel libraries
  - Easy to use
  - Saves time
  - Takes care of the parallel coding for you
  - Tested and vetted by the community

Math Kernel Library

- getexample MKL_benchmark
- getexample MKL_c_eigenvalues
- getexample MKL_Example
- getexample MKL_mic
- getexample MKL_parallel

Other Libraries

- fftw
- BLAS
- ACML
- BLAS (Basic Linar Algibra)
- Lapak
- trilinos
- petsc
- Magma
- Cudatools
- Mumps

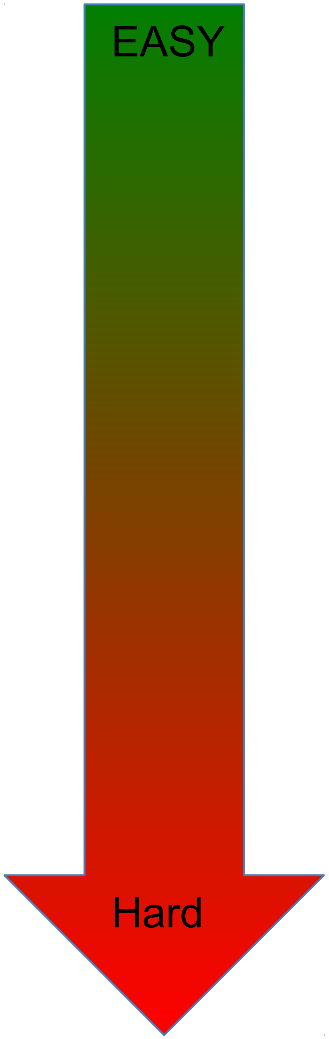
https://wiki.hpcc.msu.edu/x/6QFiAQ

Which approach is the best?

- Depends on what you are doing?
- Depends on how much communication you need.
- Depends on what hardware you have.
- Depends on how much time you have.

My Recommendations

- Pleasantly Parallel
- Standard Libraries
- OpenMP
- OpenACC
- OpenMP on Phi
- MPI
- MPI on Phi?
- GPGPU



https://wiki.hpcc.msu.edu/x/6QFiAQ

Agenda

- What is iCER / HPCC
- Common classes problems
- Overview of Hardware
- Getting started, Seven Steps to High Performance
- Running in parallel
- Tips and tricks

Tips and Tricks
Going beyond system Limits

- Going beyond system Limits
  - More than 520 jobs
  - Jobs longer than 1 week
  - Taking advantage of more nodes

Finding more Nodes

- Owners are guaranteed access to their buy-in node within 4 hours. If they are not using the node, others can use it:
    - #PBS –l walltime=04:00:00
- Some of the nodes do not have Infiniband. If you are not using scratch and do not need between node communication you can access these nodes:
    - #PBS feature=gbe

https://wiki.hpcc.msu.edu/x/6QFiAQ

Checkpoint / Restart

- What?
  - Save the state of your program
  - Restart your program from the saved point
- How?
  - Design into your program
  - BLCR (Berkley Lab Checkpoint Restart)
  - Condor Checkpoint Restart
  - Others
- Why?
  - Robust jobs
    - As HPC scales … hardware failures are guaranteed
  - Longer jobs
  - Better science

Questions?

- Software Carpentry (the basics):
  - http://www.softwarecarpentry.org/
- Announcements:
  - https://wiki.hpcc.msu.edu/
- Documentation and User Manual:
  - https://wiki.hpcc.msu.edu/x/A4AN
- Training Videos:
  - https://www.youtube.com/user/icermsu
- Western Michigan University contact:
  - donald.weber@wmich.edu
- Online Chat:
  - http://www.hipchat.com/gYlrQfgah
- Contact us:
  - wmichhelp@hpcc.msu.edu

https://wiki.hpcc.msu.edu/x/6QFiAQ