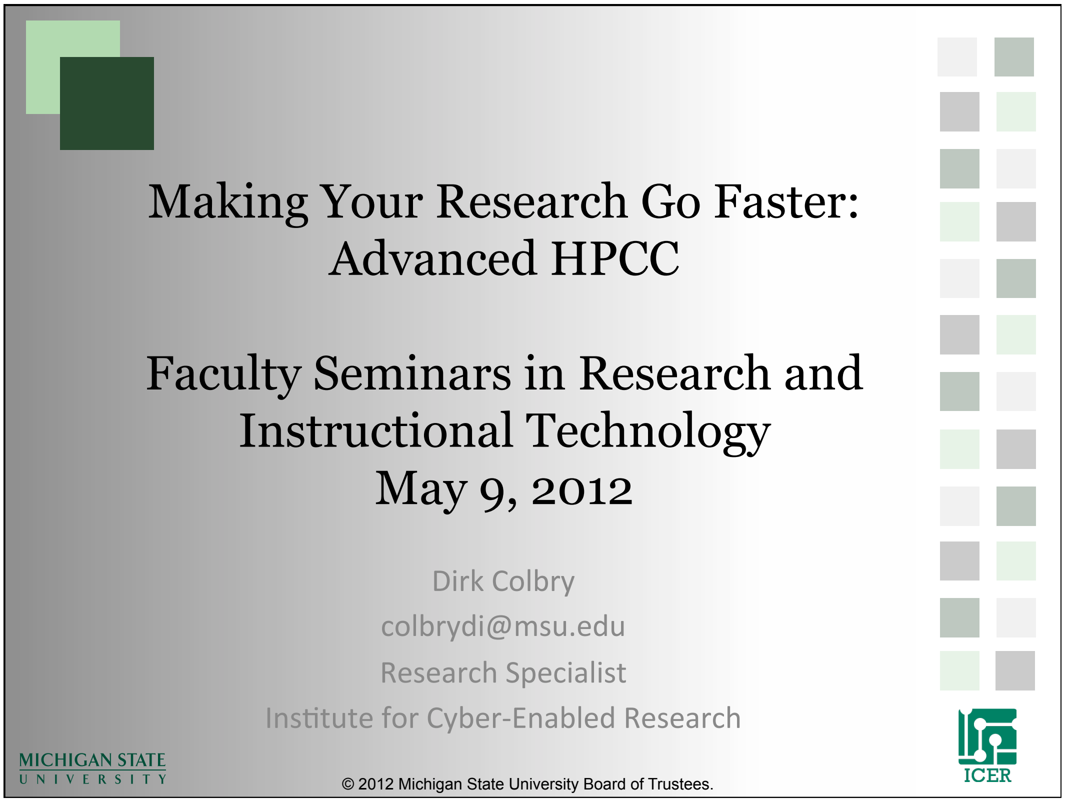


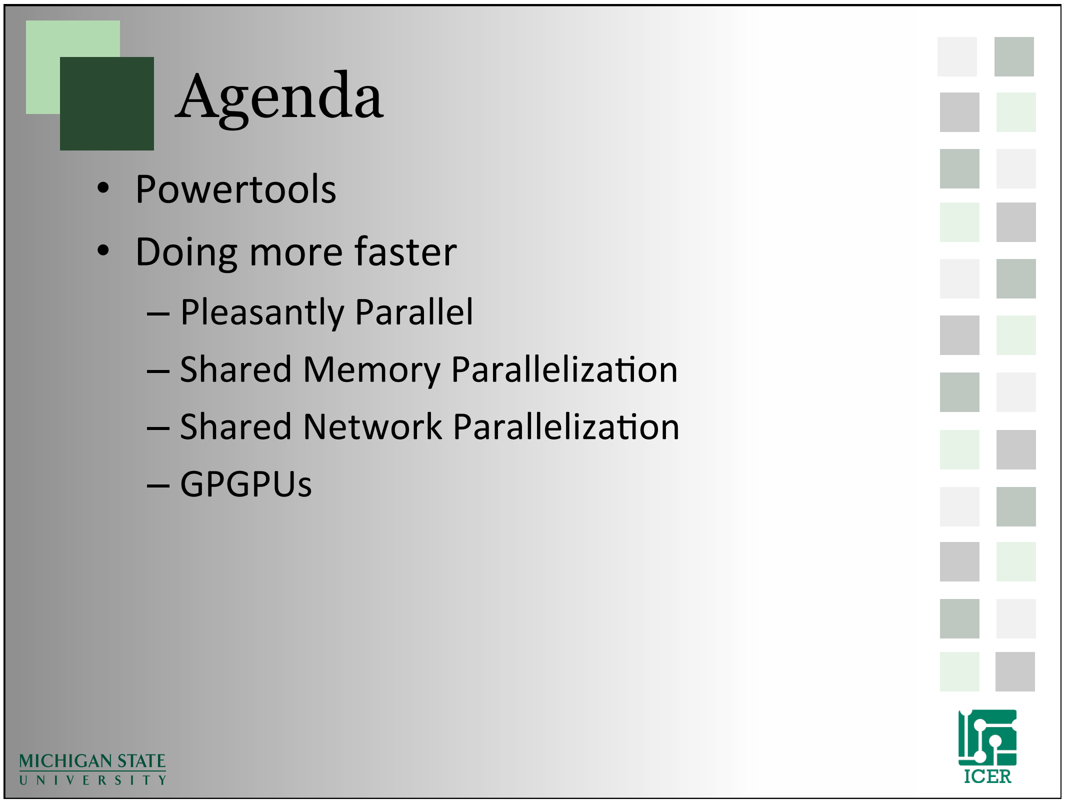
Making Your Research Go Faster: Advanced HPCC

Faculty Seminars in Research and Instructional Technology
May 9, 2012

Dirk Colbry
colbrydi@msu.edu
Research Specialist
Institute for Cyber-Enabled Research

MICHIGAN STATE
UNIVERSITY

ICER



Agenda

- Powertools
- Doing more faster
  - Pleasantly Parallel
  - Shared Memory Parallelization
  - Shared Network Parallelization
  - GPGPUs

MICHIGAN STATE
UNIVERSITY

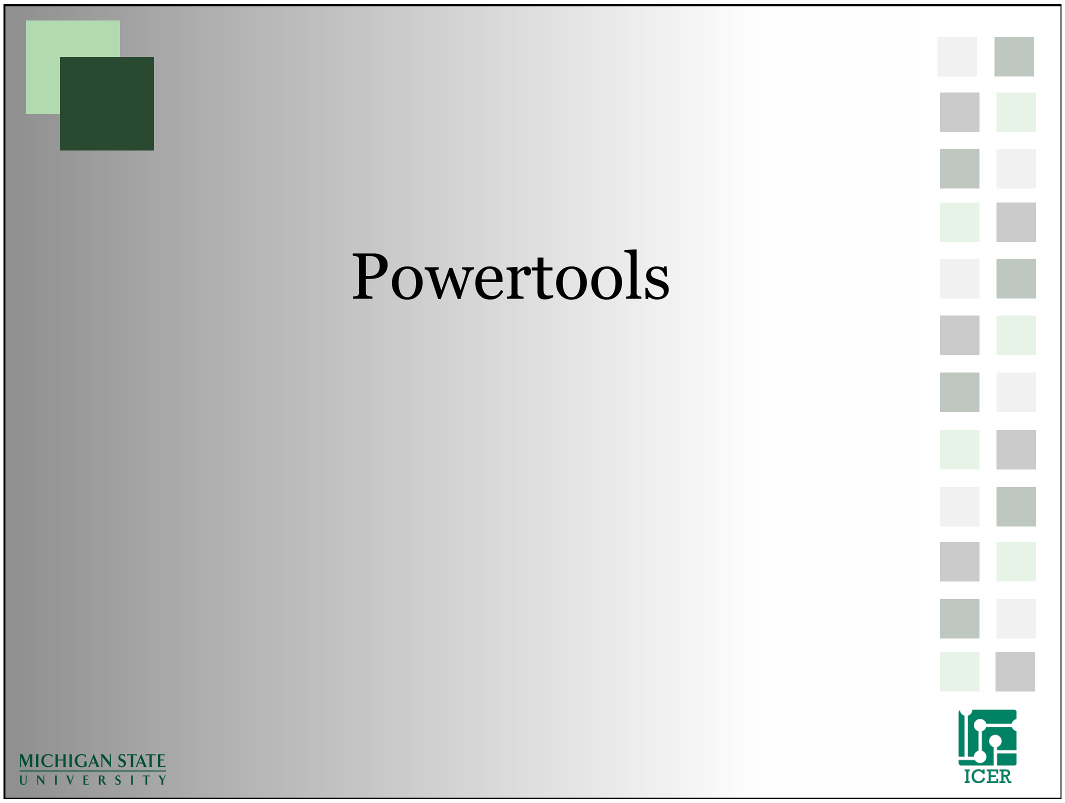
ICER

Submission Scripts

- Design Goals
  - One script does everything
  - Easy to read
  - Easily given to others
  - Easily moved to different directories

MICHIGAN STATE UNIVERSITY

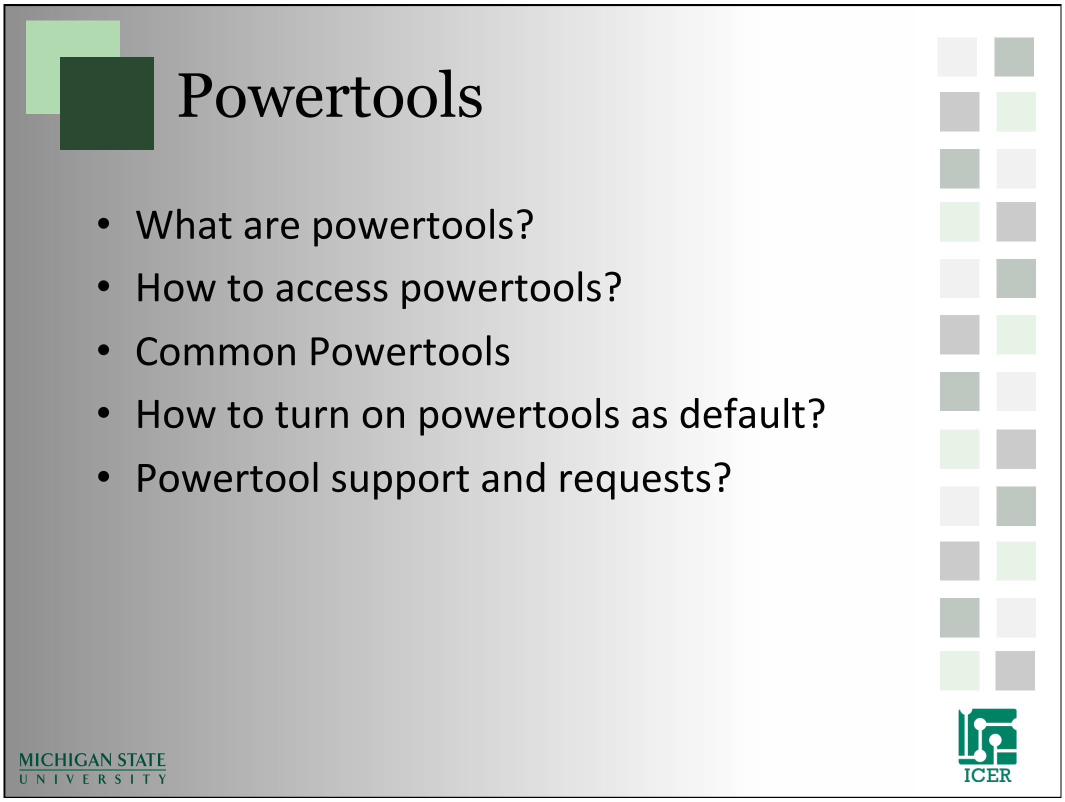
ICER



Powertools

MICHIGAN STATE UNIVERSITY

ICER

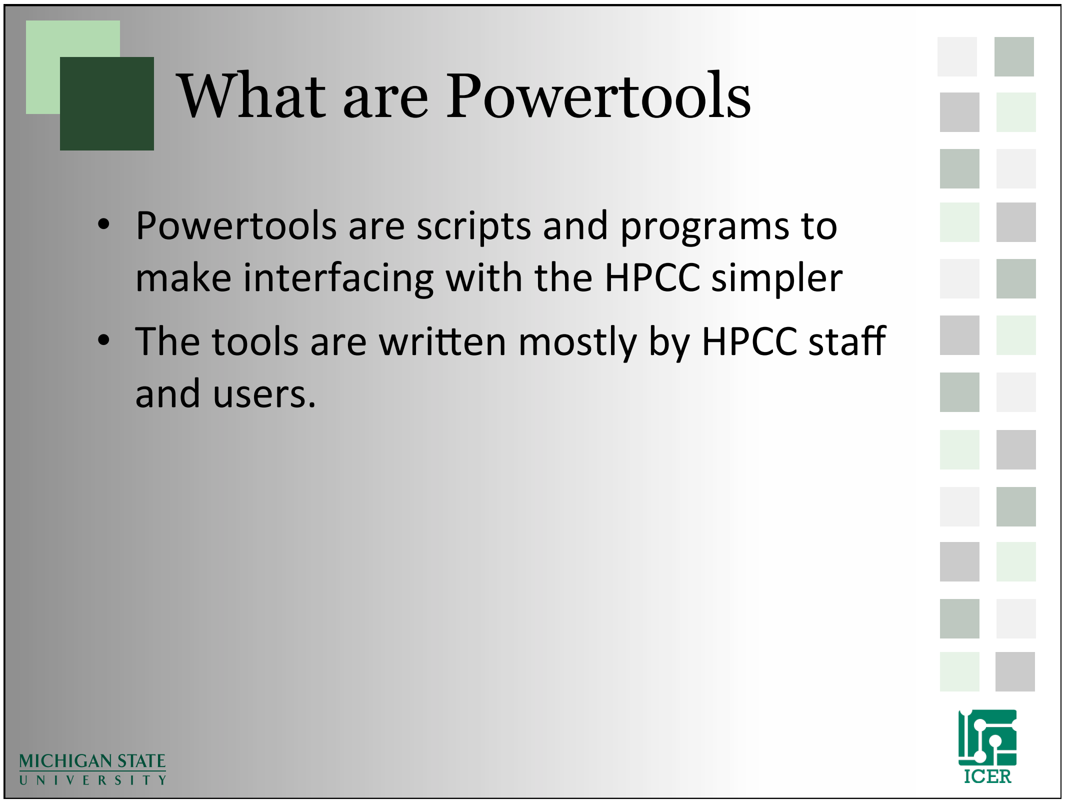


Powertools

- What are powertools?
- How to access powertools?
- Common Powertools
- How to turn on powertools as default?
- Powertool support and requests?

MICHIGAN STATE UNIVERSITY

ICER

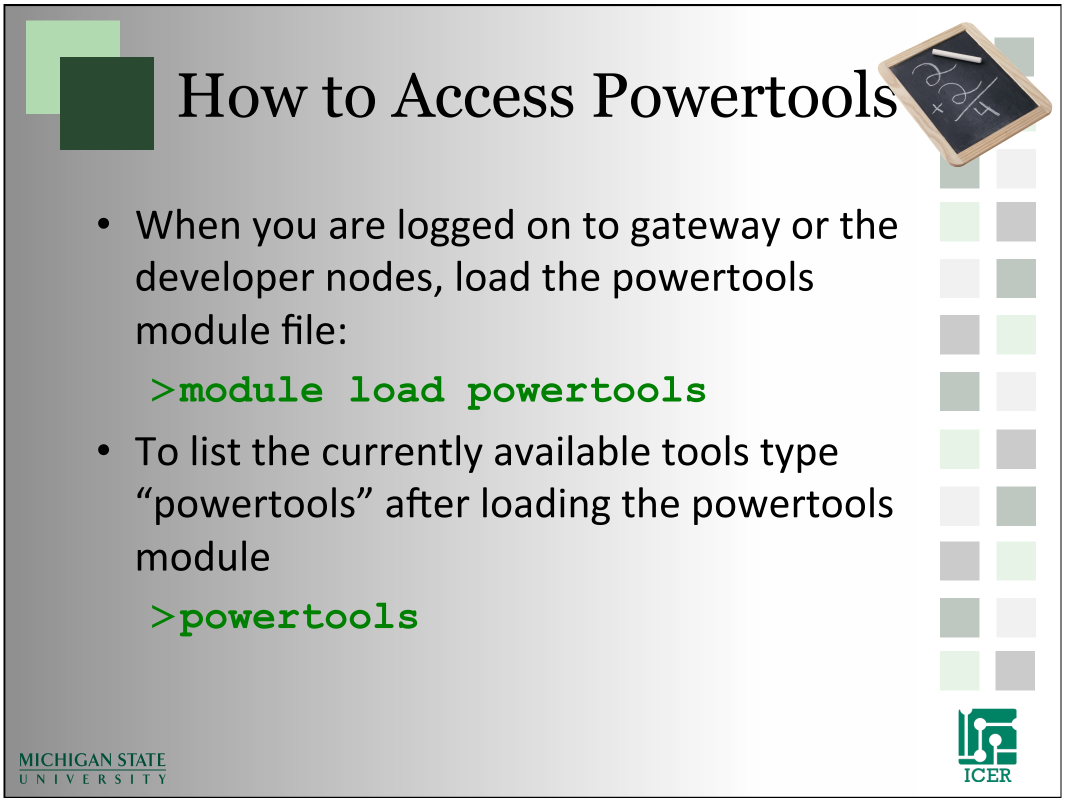


What are Powertools

- Powertools are scripts and programs to make interfacing with the HPCC simpler
- The tools are written mostly by HPCC staff and users.

MICHIGAN STATE UNIVERSITY

ICER



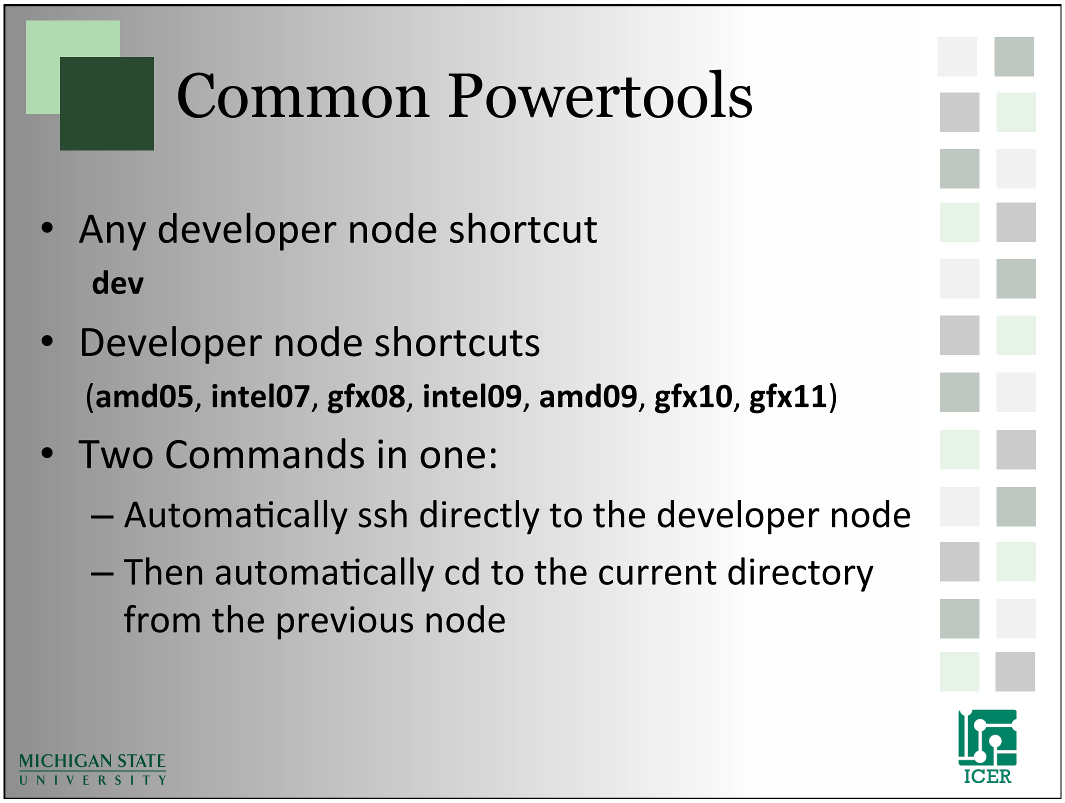
How to Access Powertools

- When you are logged on to gateway or the developer nodes, load the powertools module file:

```
>module load powertools
```

- To list the currently available tools type "powertools" after loading the powertools module

```
>powertools
```



Common Powertools

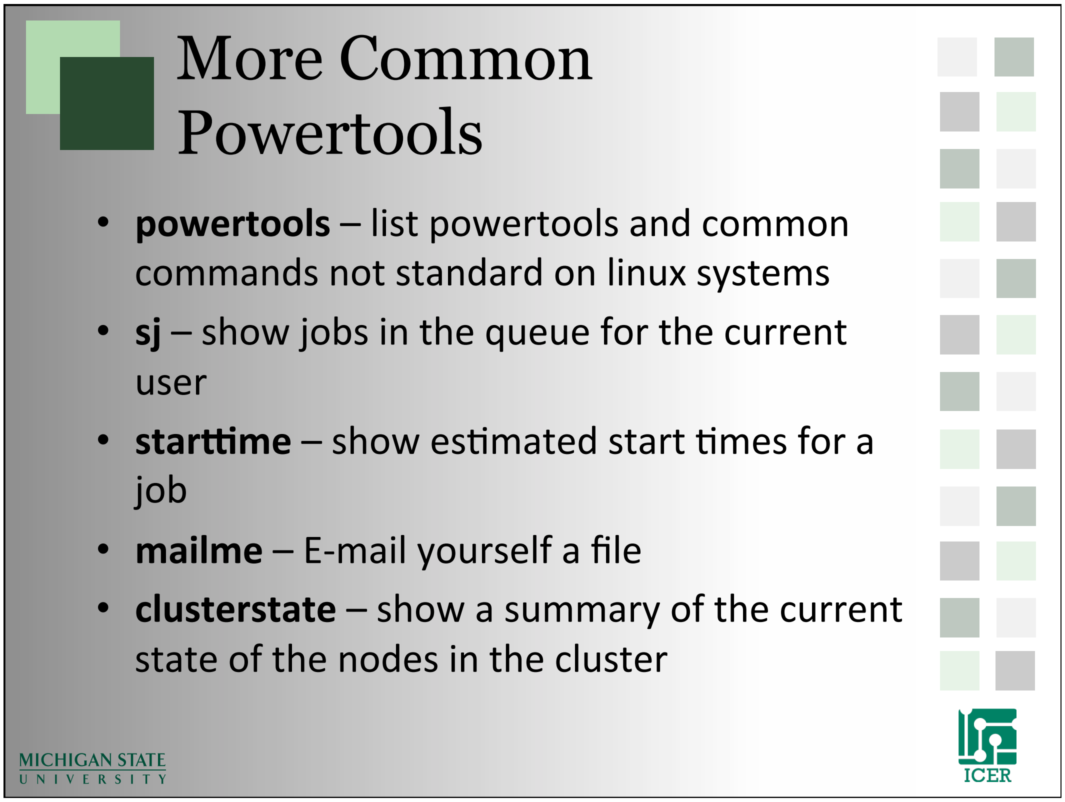
- Any developer node shortcut

  dev

- Developer node shortcuts

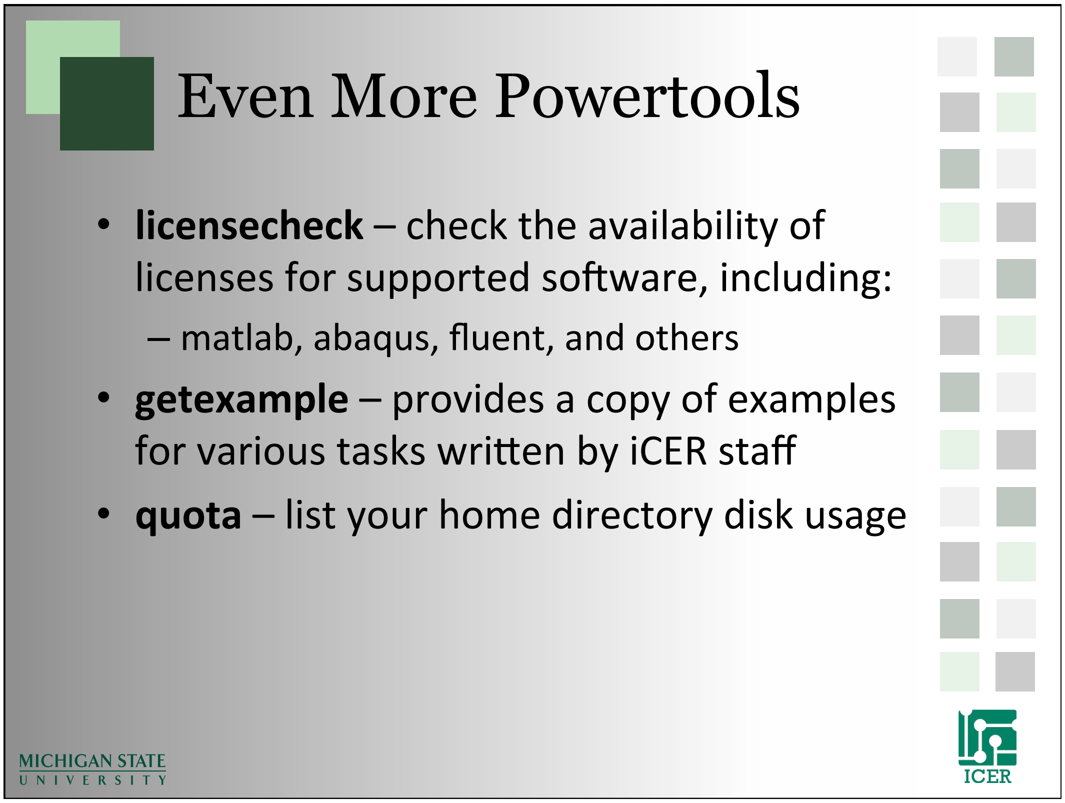
  (**amd05**, **intel07**, **gfx08**, **intel09**, **amd09**, **gfx10**, **gfx11**)

- Two Commands in one:
  - Automatically ssh directly to the developer node
  - Then automatically cd to the current directory from the previous node



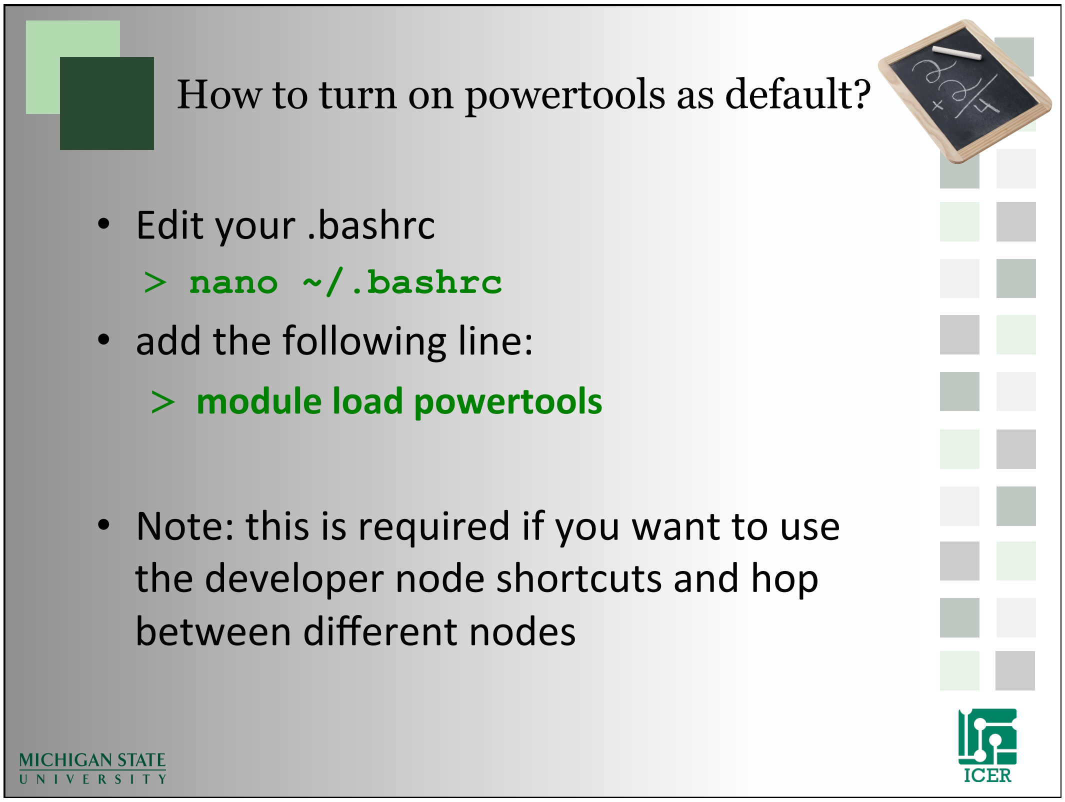
More Common Powertools

- **powertools** – list powertools and common commands not standard on linux systems
- **sj** – show jobs in the queue for the current user
- **starttime** – show estimated start times for a job
- **mailme** – E-mail yourself a file
- **clusterstate** – show a summary of the current state of the nodes in the cluster



Even More Powertools

- **licensecheck** – check the availability of licenses for supported software, including:
  - matlab, abaqus, fluent, and others
- **getexample** – provides a copy of examples for various tasks written by iCER staff
- **quota** – list your home directory disk usage



How to turn on powertools as default?

- Edit your .bashrc

```
> nano ~/.bashrc
```

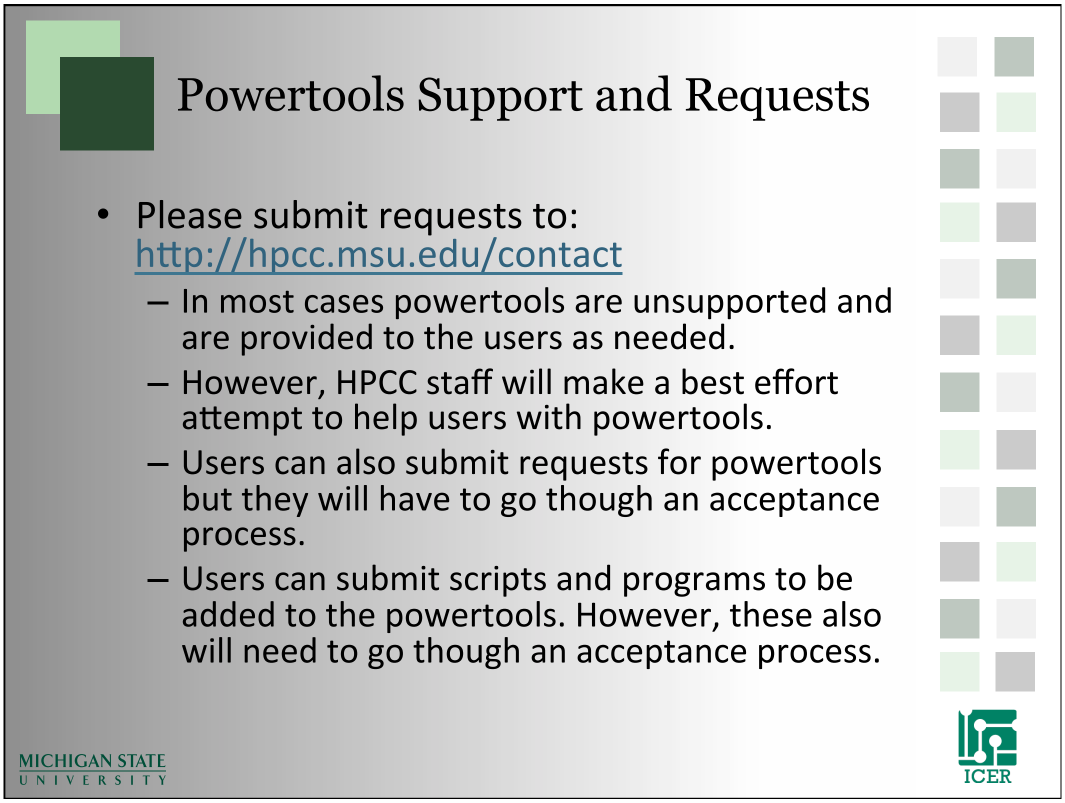
- add the following line:

```
> module load powertools
```

- Note: this is required if you want to use the developer node shortcuts and hop between different nodes

MICHIGAN STATE UNIVERSITY

ICER

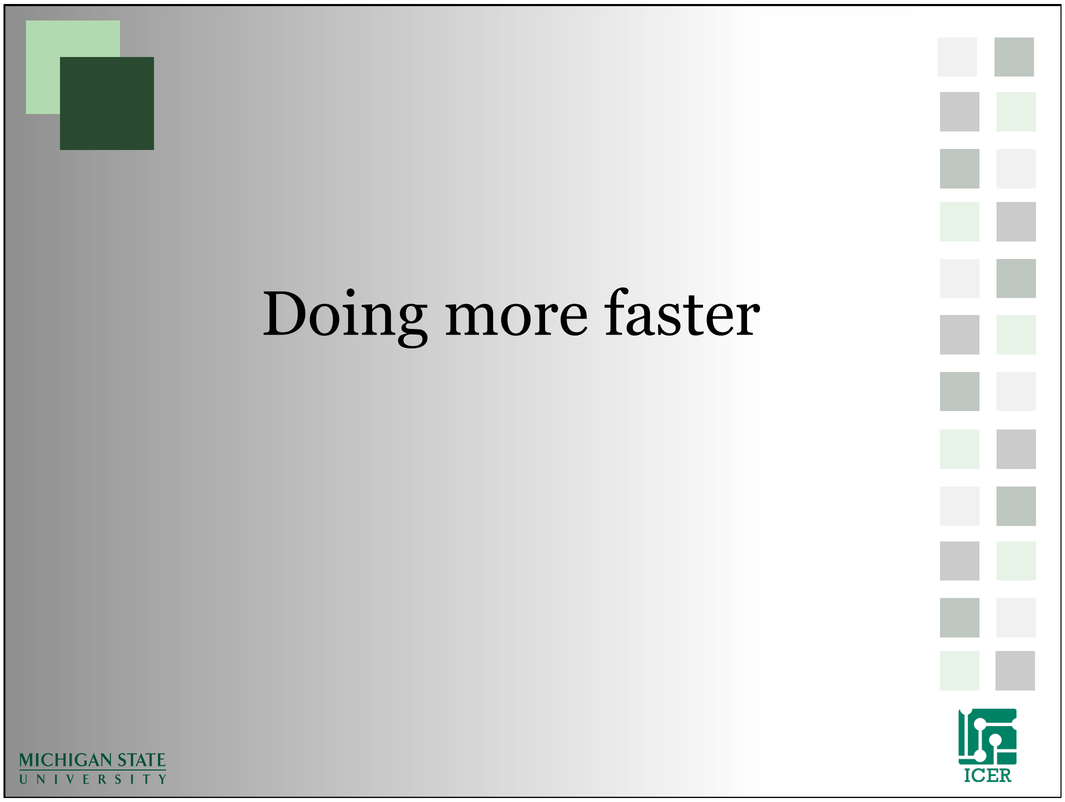


Powertools Support and Requests

- Please submit requests to: http://hpcc.msu.edu/contact
  - In most cases powertools are unsupported and are provided to the users as needed.
  - However, HPCC staff will make a best effort attempt to help users with powertools.
  - Users can also submit requests for powertools but they will have to go though an acceptance process.
  - Users can submit scripts and programs to be added to the powertools. However, these also will need to go though an acceptance process.

MICHIGAN STATE UNIVERSITY

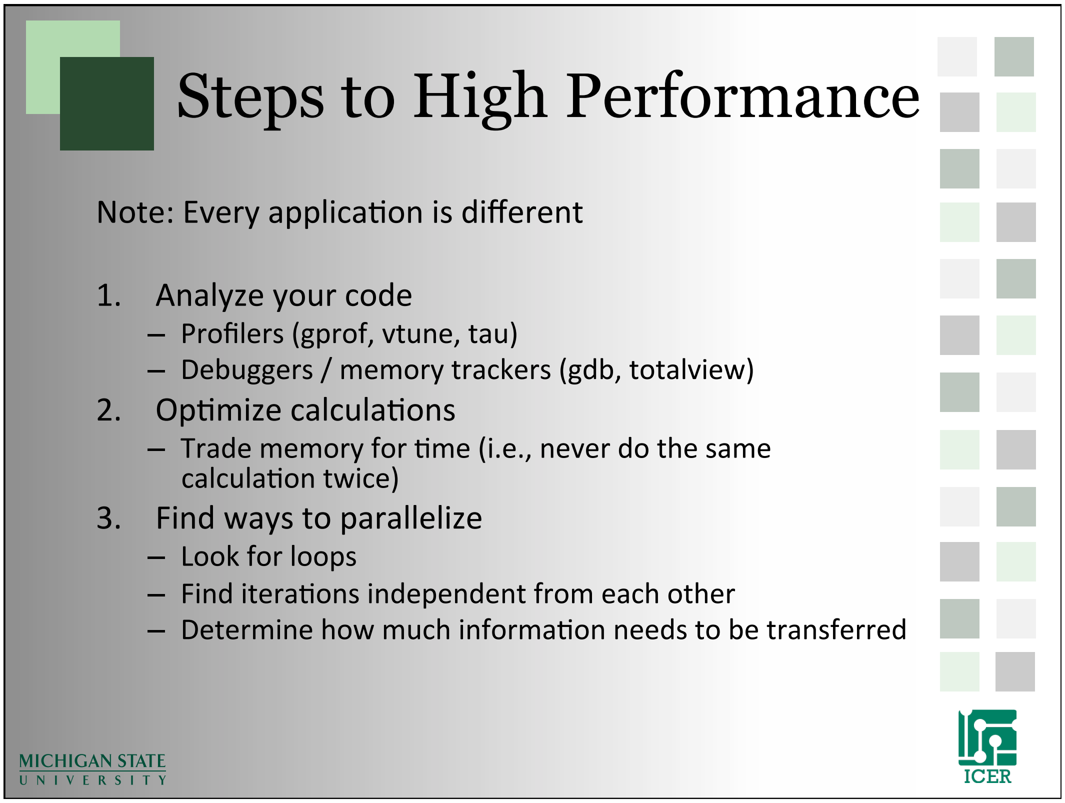
ICER



Doing more faster

MICHIGAN STATE UNIVERSITY

ICER



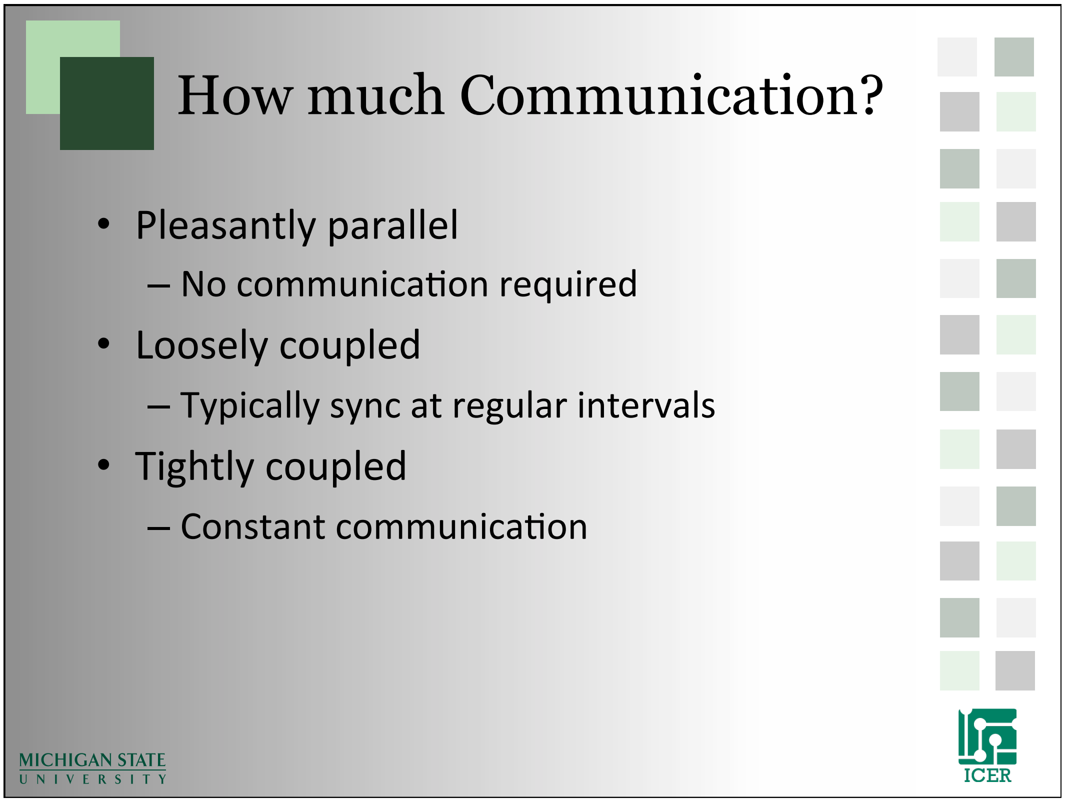
Steps to High Performance

Note: Every application is different

1. Analyze your code
   - Profilers (gprof, vtune, tau)
   - Debuggers / memory trackers (gdb, 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

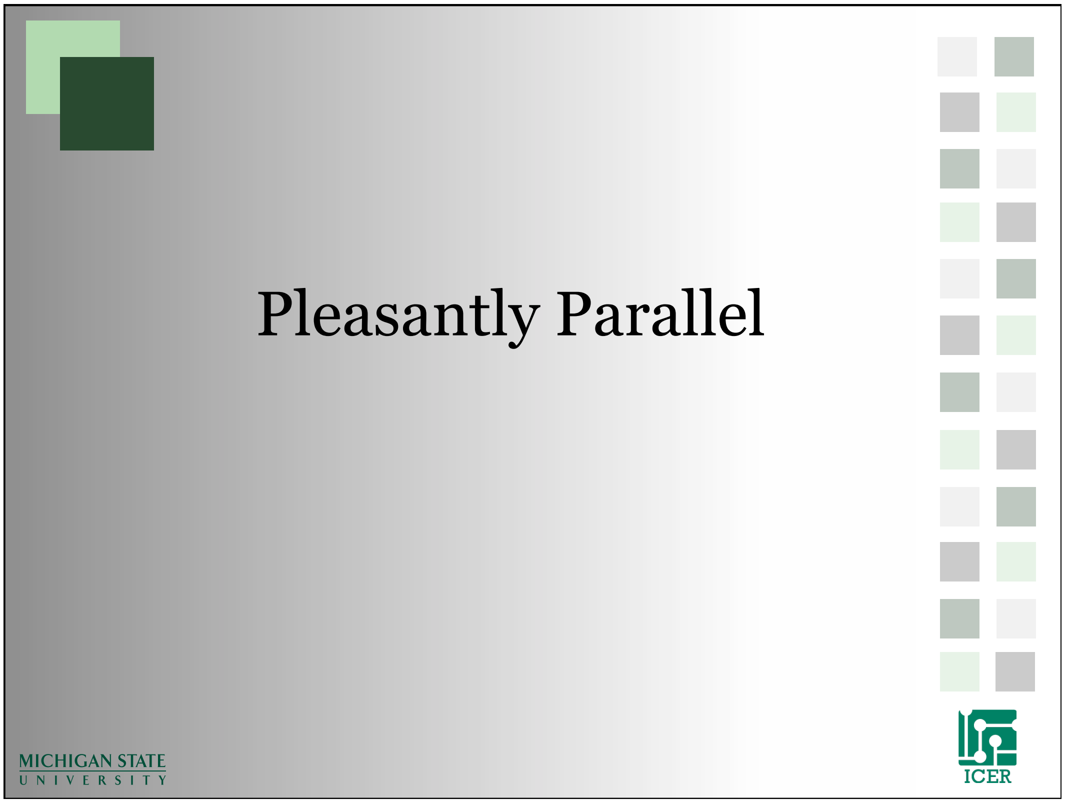
MICHIGAN STATE UNIVERSITY

ICER

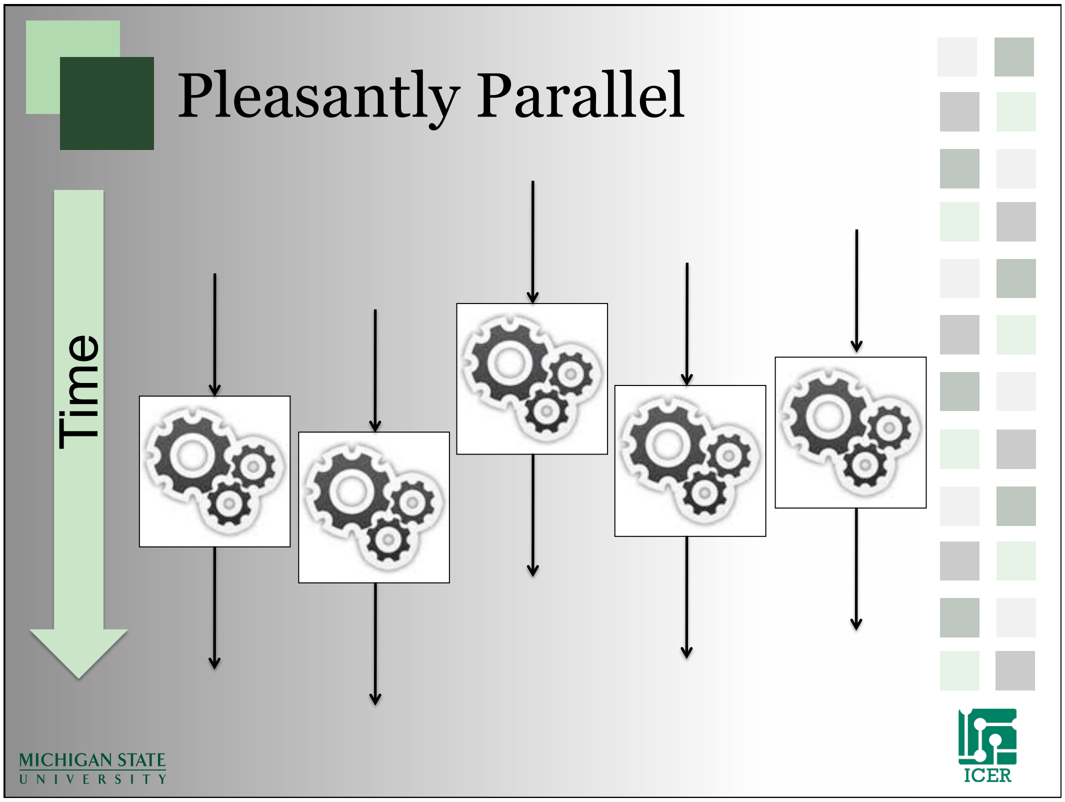


How much Communication?

- Pleasantly parallel
  - No communication required
- Loosely coupled
  - Typically sync at regular intervals
- Tightly coupled
  - Constant communication



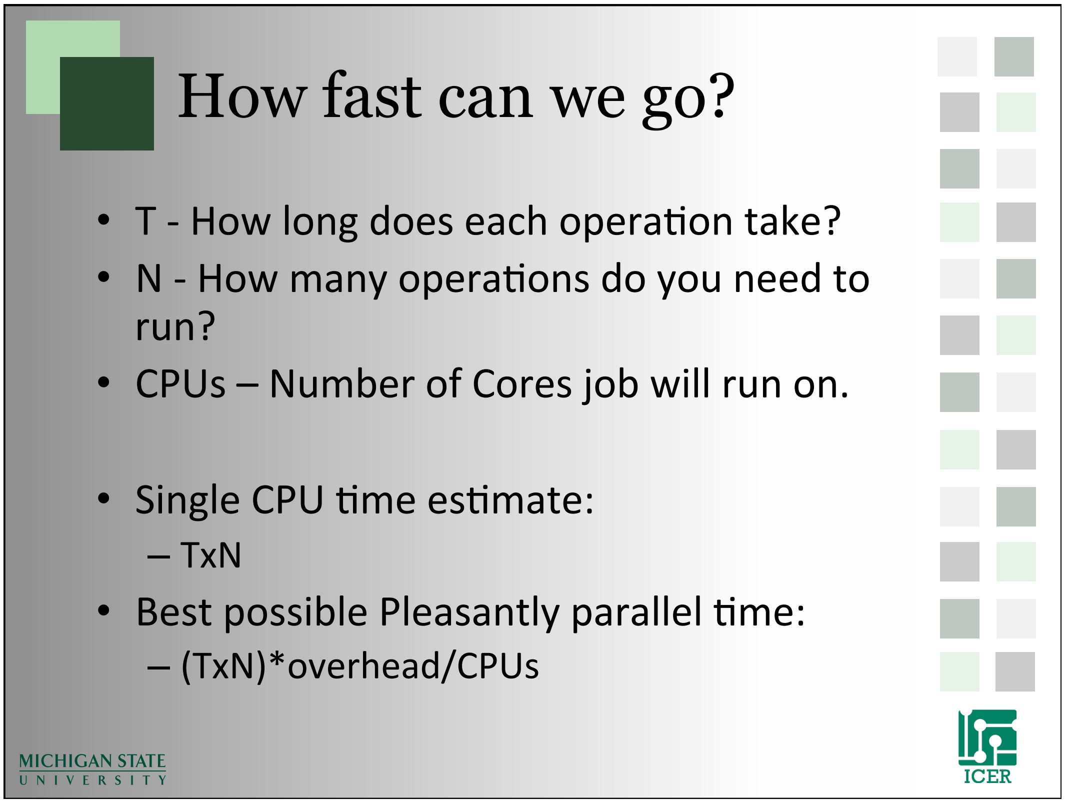
Pleasantly Parallel



Pleasantly Parallel

MICHIGAN STATE UNIVERSITY

ICER



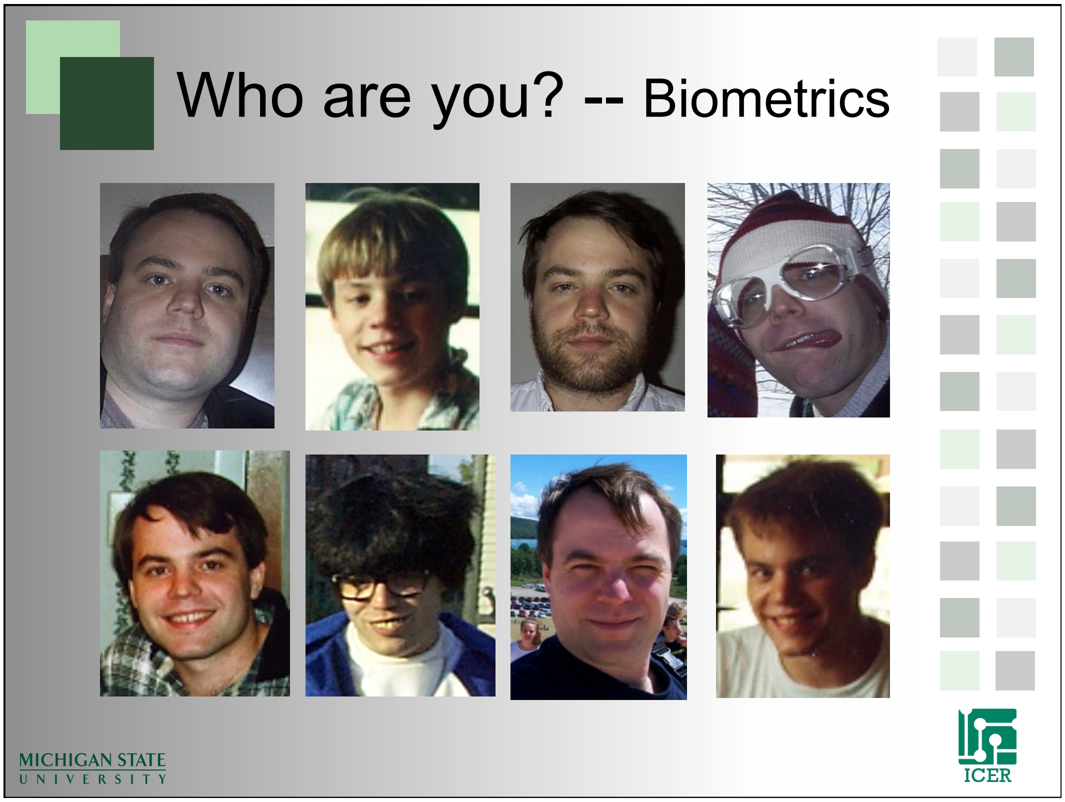
How fast can we go?

- 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

MICHIGAN STATE UNIVERSITY

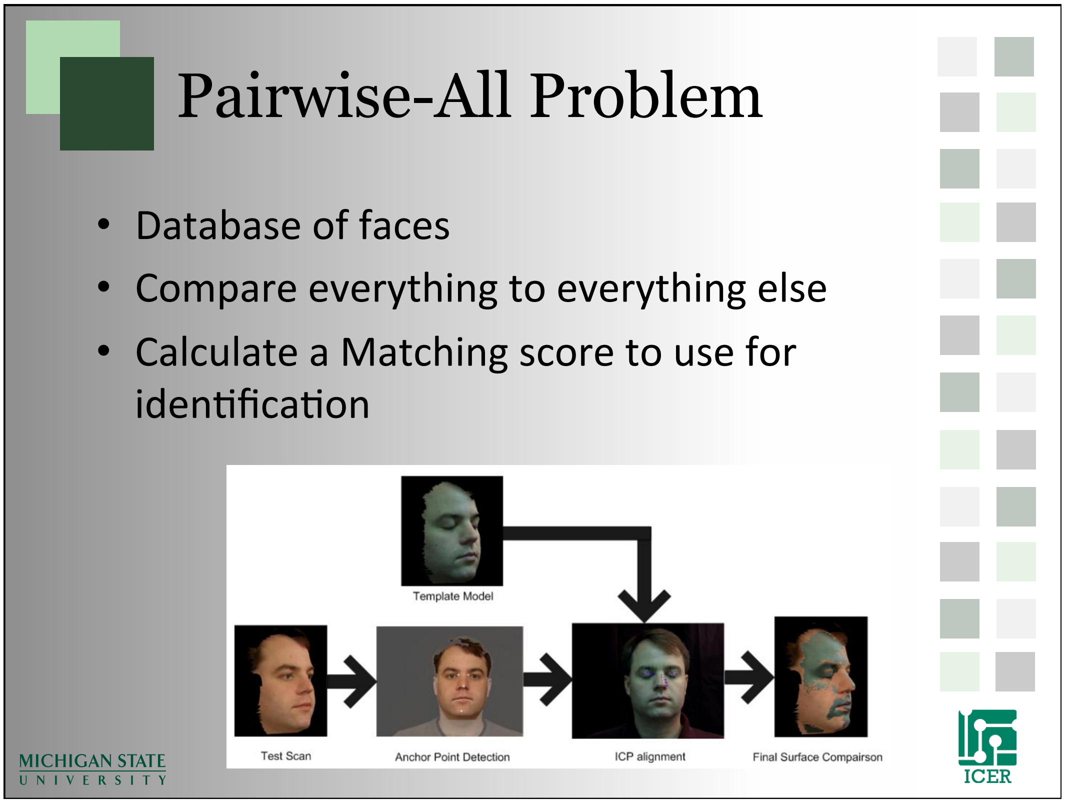
ICER



Who are you? -- Biometrics

MICHIGAN STATE UNIVERSITY

ICER

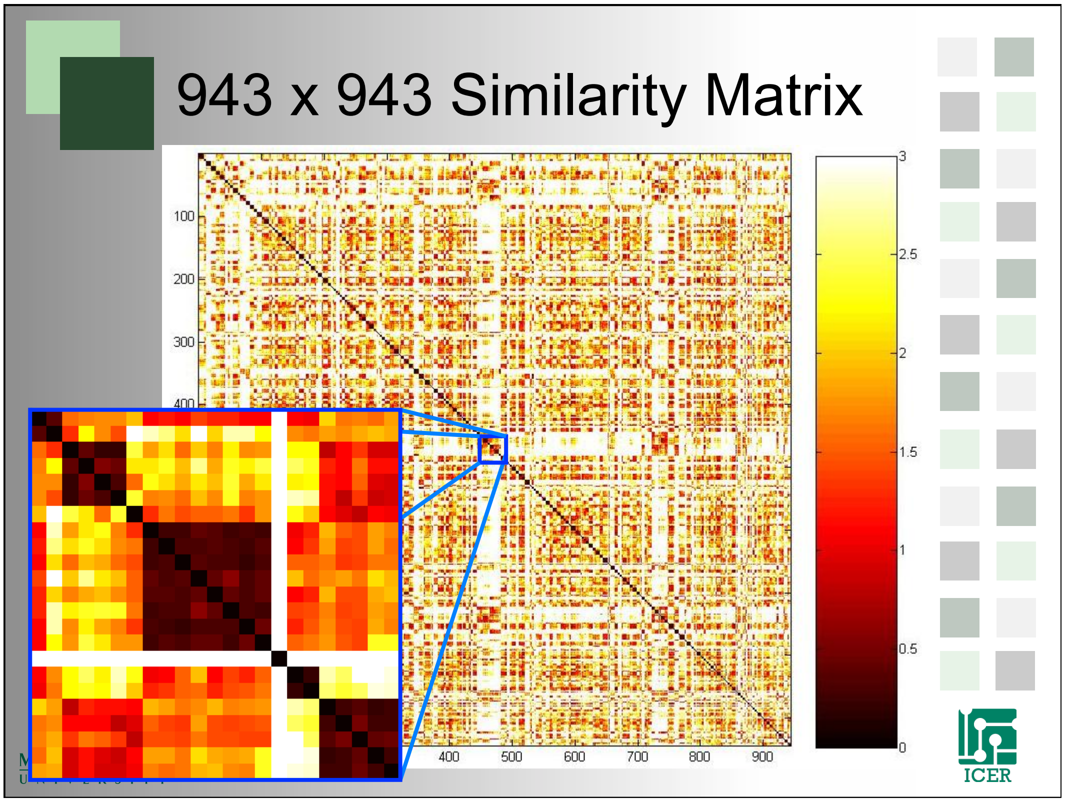


Pairwise-All Problem

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

MICHIGAN STATE UNIVERSITY

ICER



943 x 943 Similarity Matrix

Estimated Calculation Times

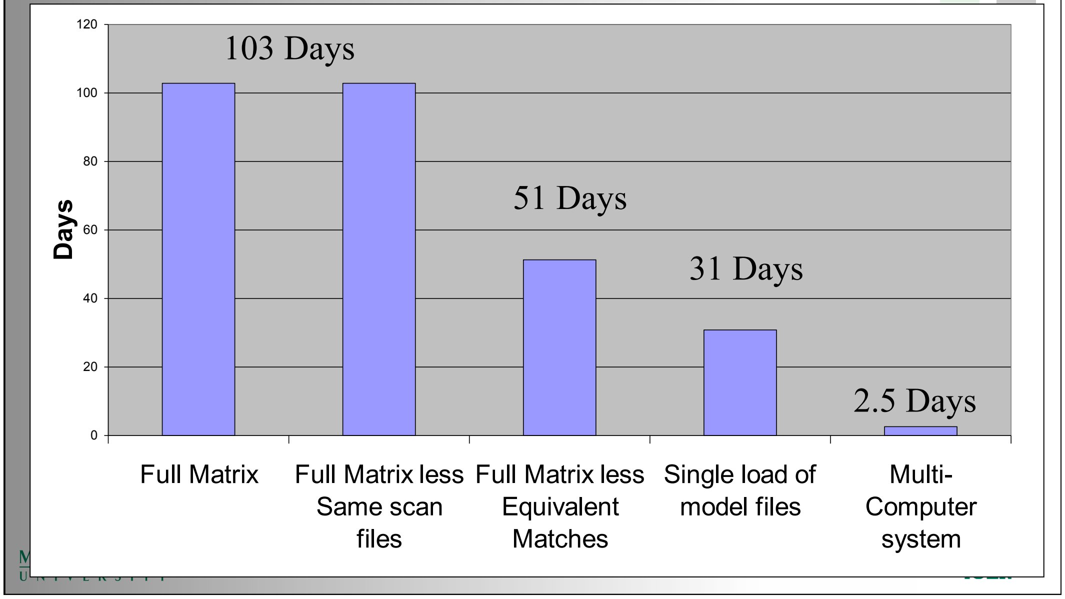
- Preprocessing
  - 943 * 12 (seconds) ≈ 189 Minutes
- Matching
  - 943 * 943 * 5 (seconds) ≈ 103 Days
- Scans matched to themselves always result in 0 mm
  - (943 * 943 – 943) * 5 (seconds) ≈ 103 Days
- The Proposed Alignment Algorithm is symmetric.
  - (943 * 943 – 943)/2 * 5 (seconds) ≈ 51.5 Days

- We also load models once per row instead of every time
  - (943*943-943)/2 * 3 (seconds) + 943 * 2 (seconds) ≈ 31 Days

MICHIGAN STATE UNIVERSITY

ICER

Calculation Time for Full Similarity Matrix

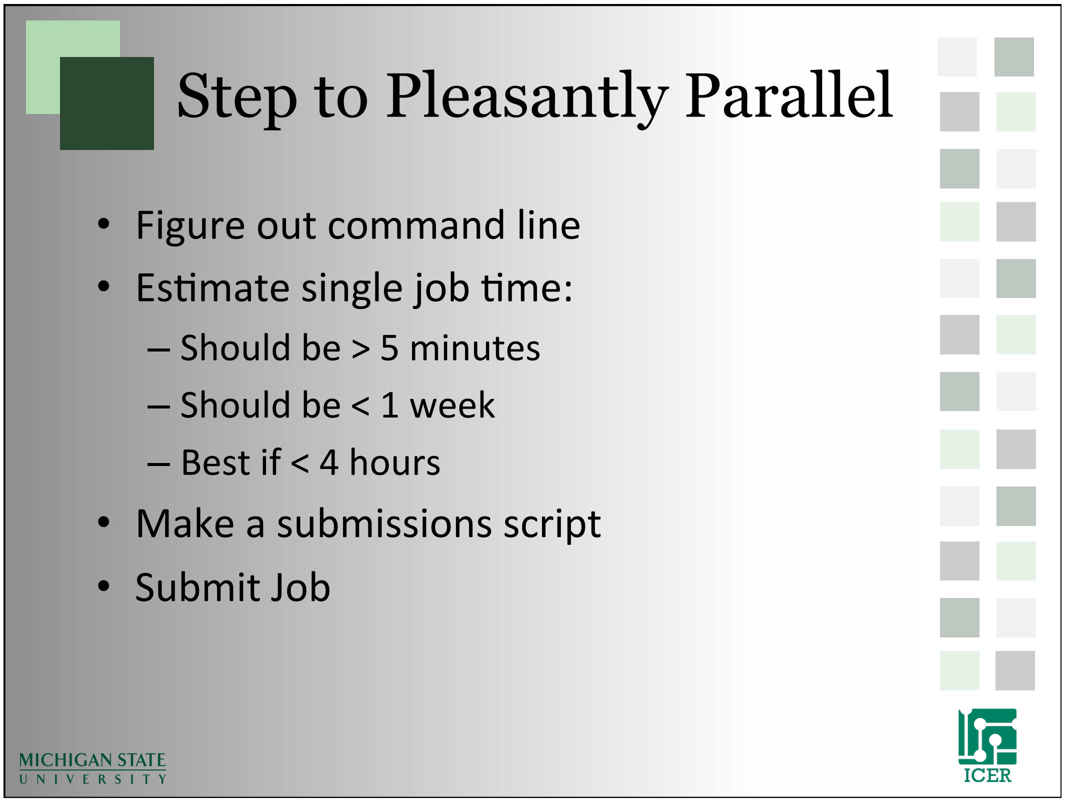


How do we go even bigger?

- 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

MICHIGAN STATE
UNIVERSITY

ICER

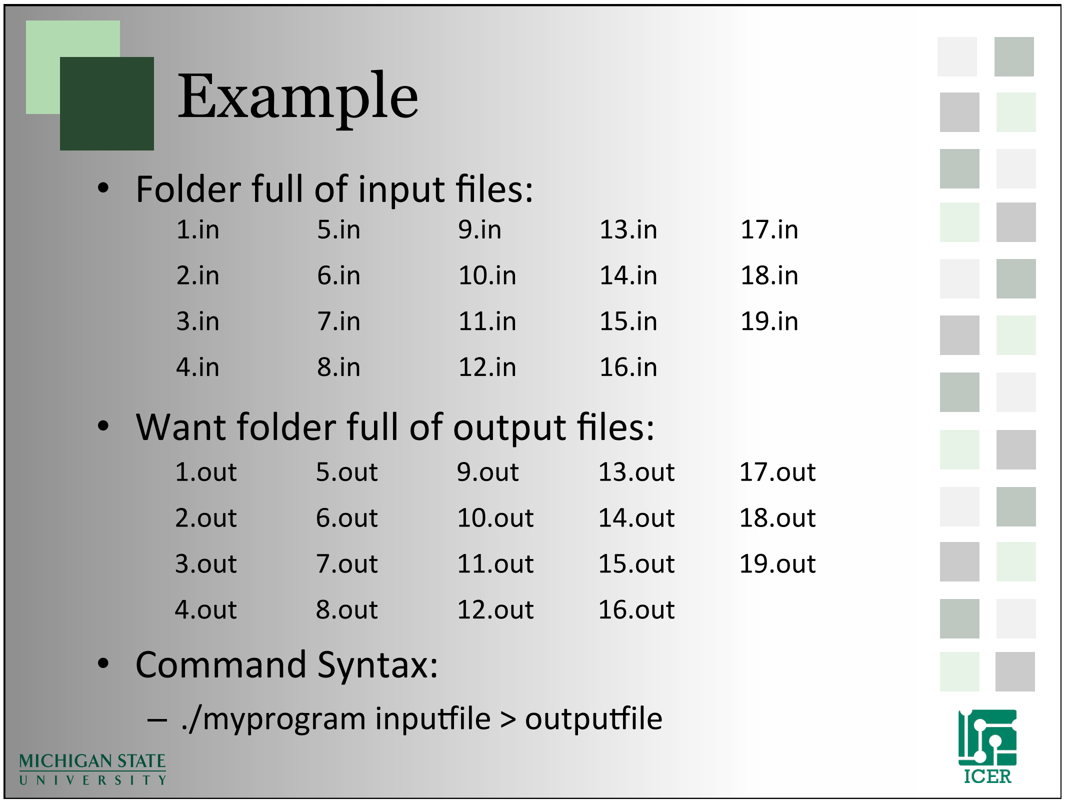


Step to Pleasantly Parallel

- 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

MICHIGAN STATE UNIVERSITY

ICER



Example

- Folder full of input files:

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

- Want folder full of output files:

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

- Command Syntax:
  - ./myprogram inputfile > outputfile

MICHIGAN STATE UNIVERSITY

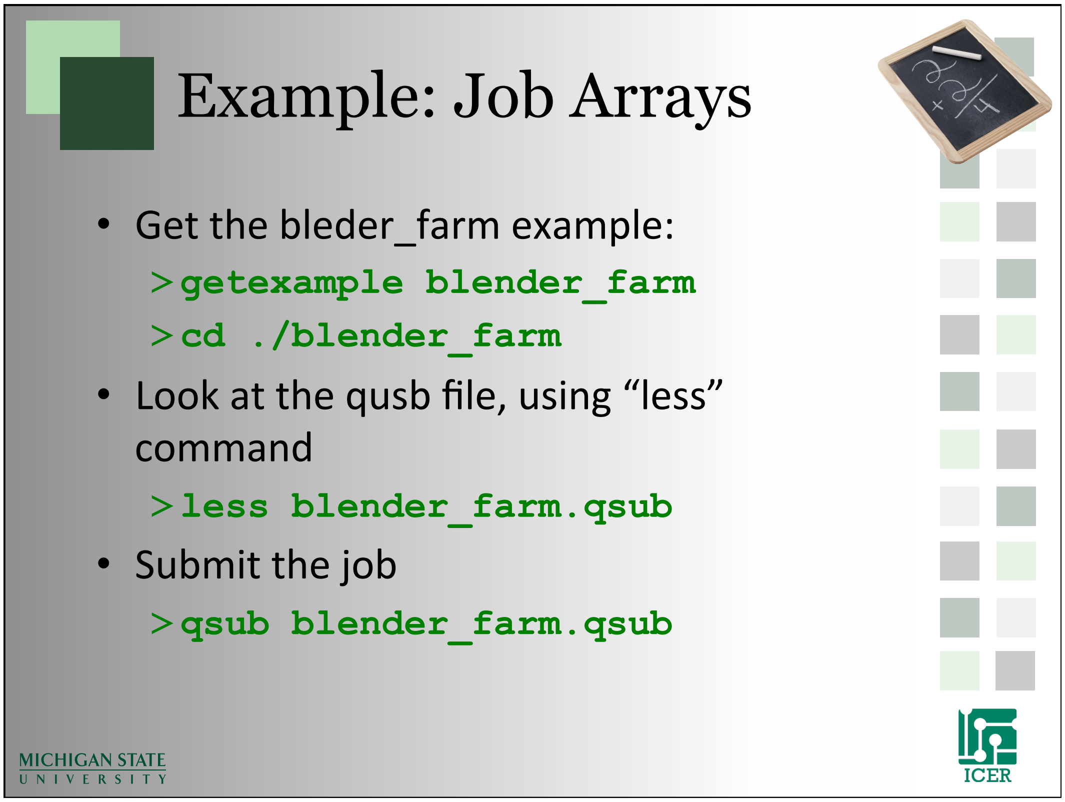
ICER

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

Simple Job Array

```
#!/bin/bash -login
#PBS -l walltime=00:05:00,mem=2gb
#PBS -l nodes=1:ppn=1,feature=gbe
#PBS -t 1-100

cd ${PBS_O_WORKID}

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

qstat -f ${PBS_JOBID}
```



Example: Job Arrays

- Get the bleder_farm example:

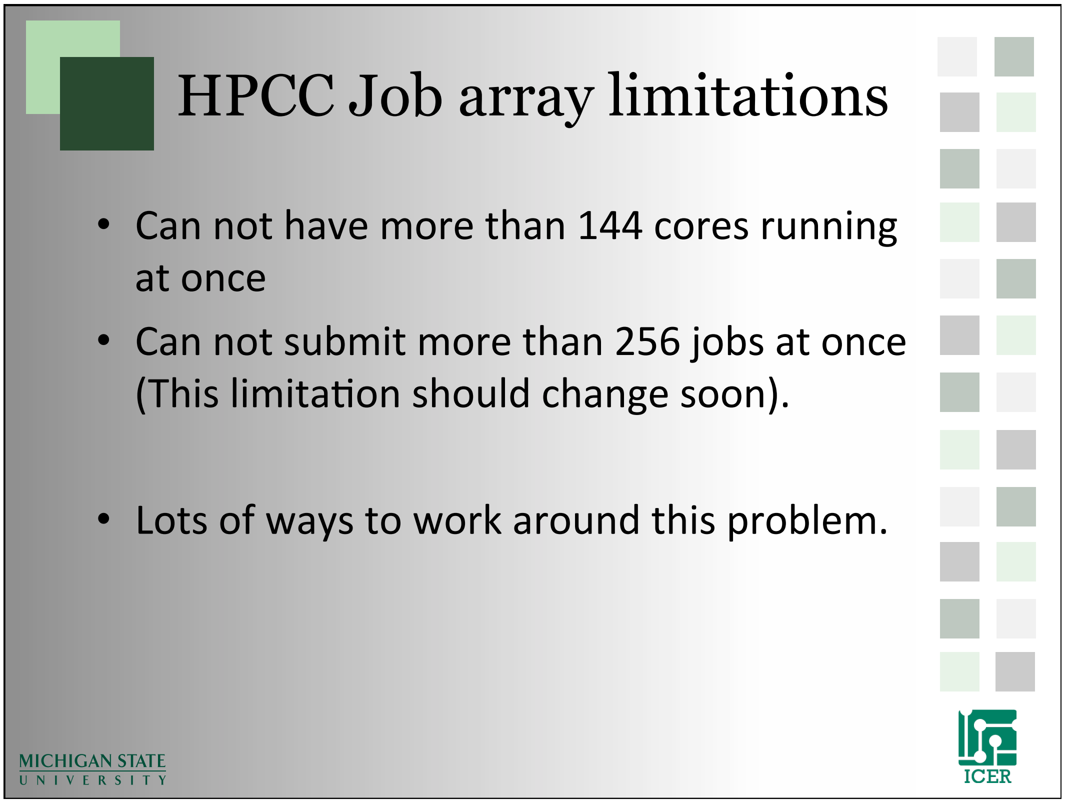
```
>getexample blender_farm
>cd ./blender_farm
```

- Look at the qusb file, using “less” command

```
>less blender_farm.qsub
```

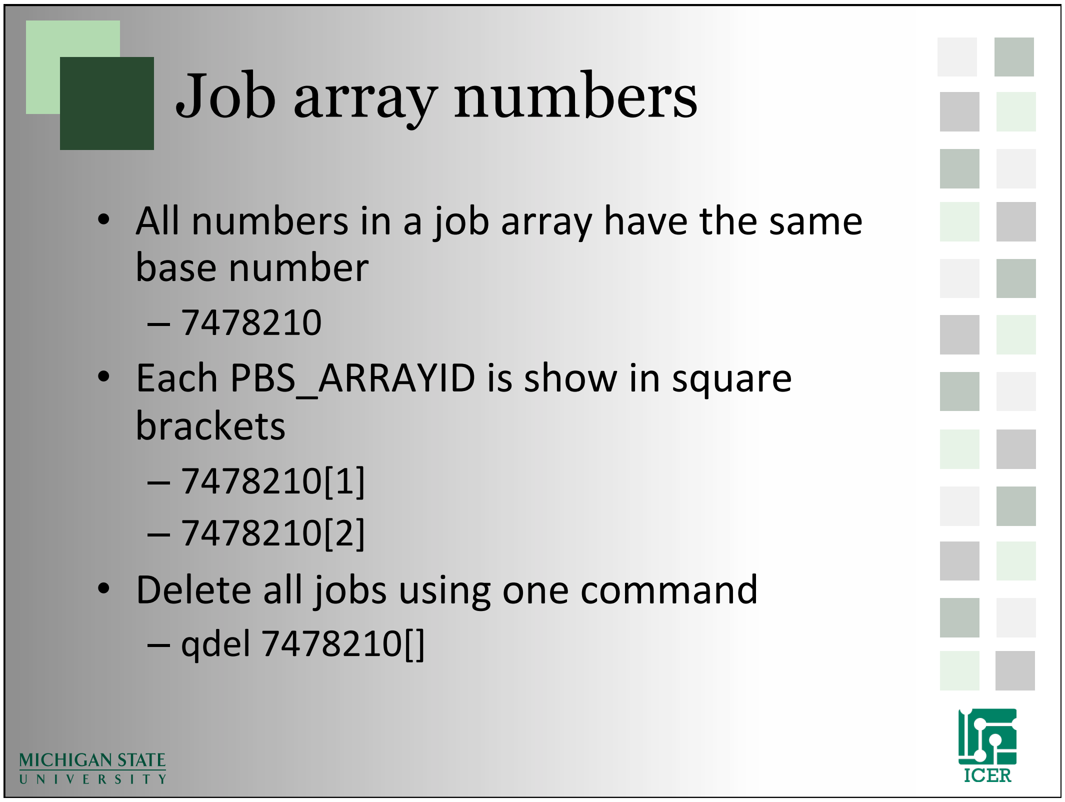
- Submit the job

```
>qsub blender_farm.qsub
```



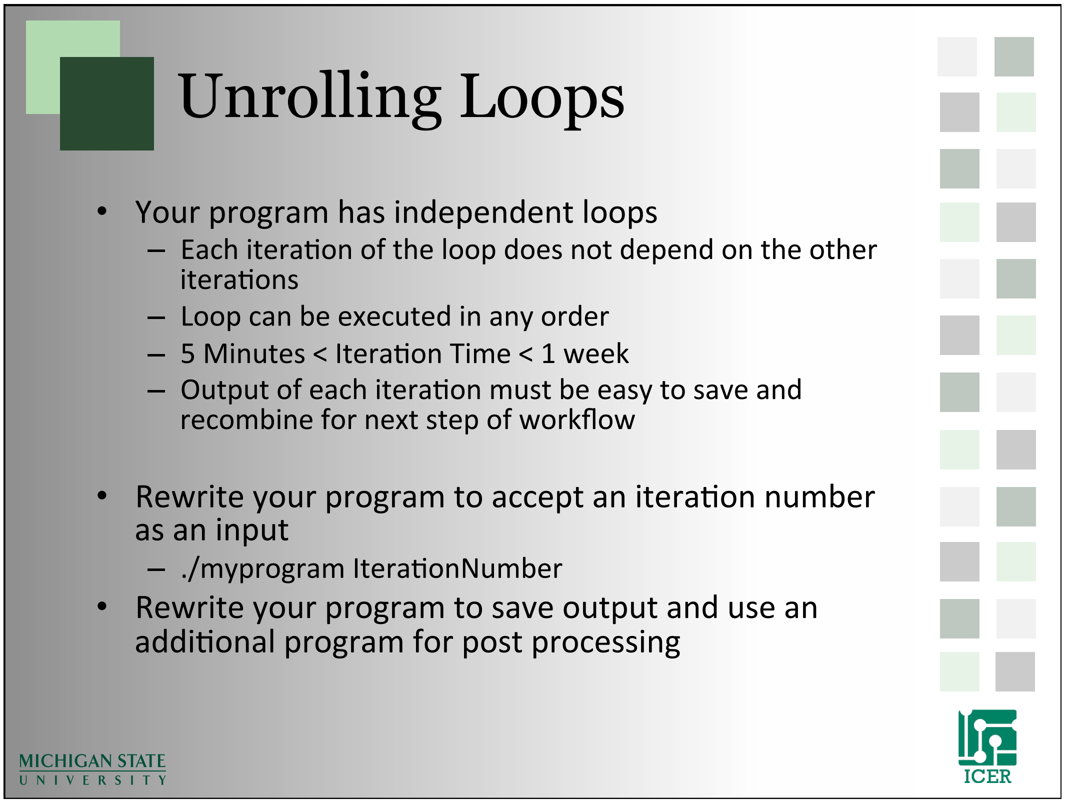
HPCC Job array limitations

- Can not have more than 144 cores running at once
- Can not submit more than 256 jobs at once (This limitation should change soon).
- Lots of ways to work around this problem.



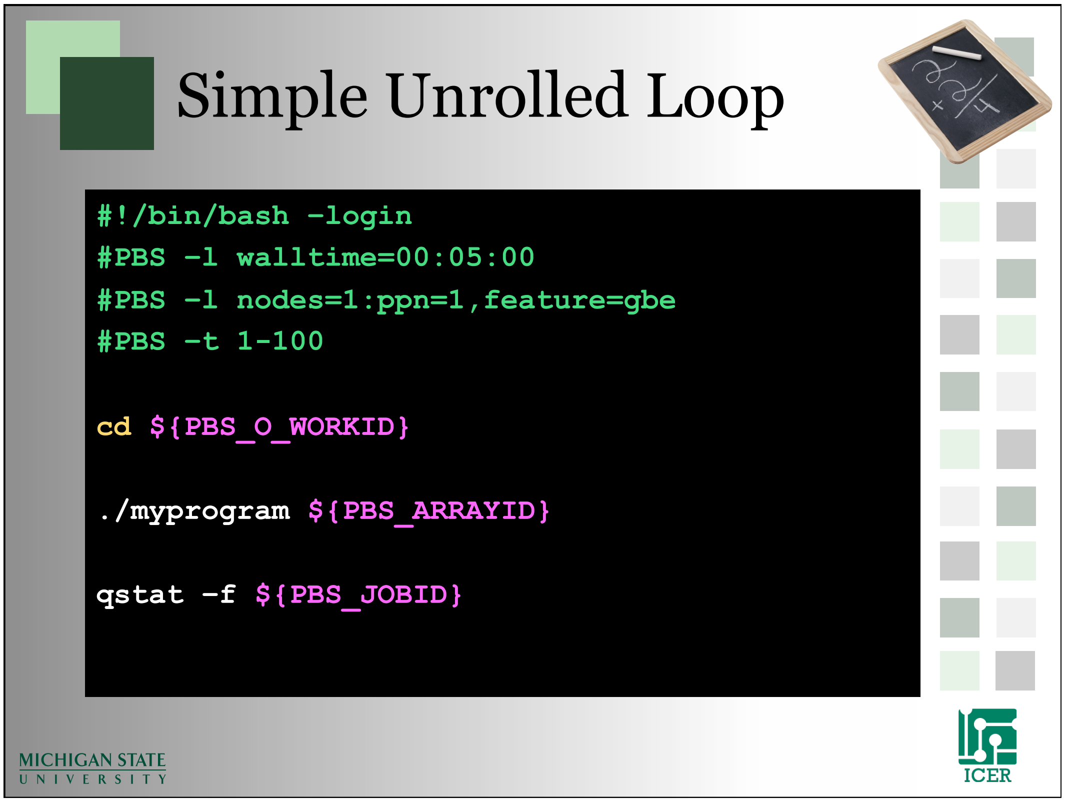
Job array numbers

- 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[]



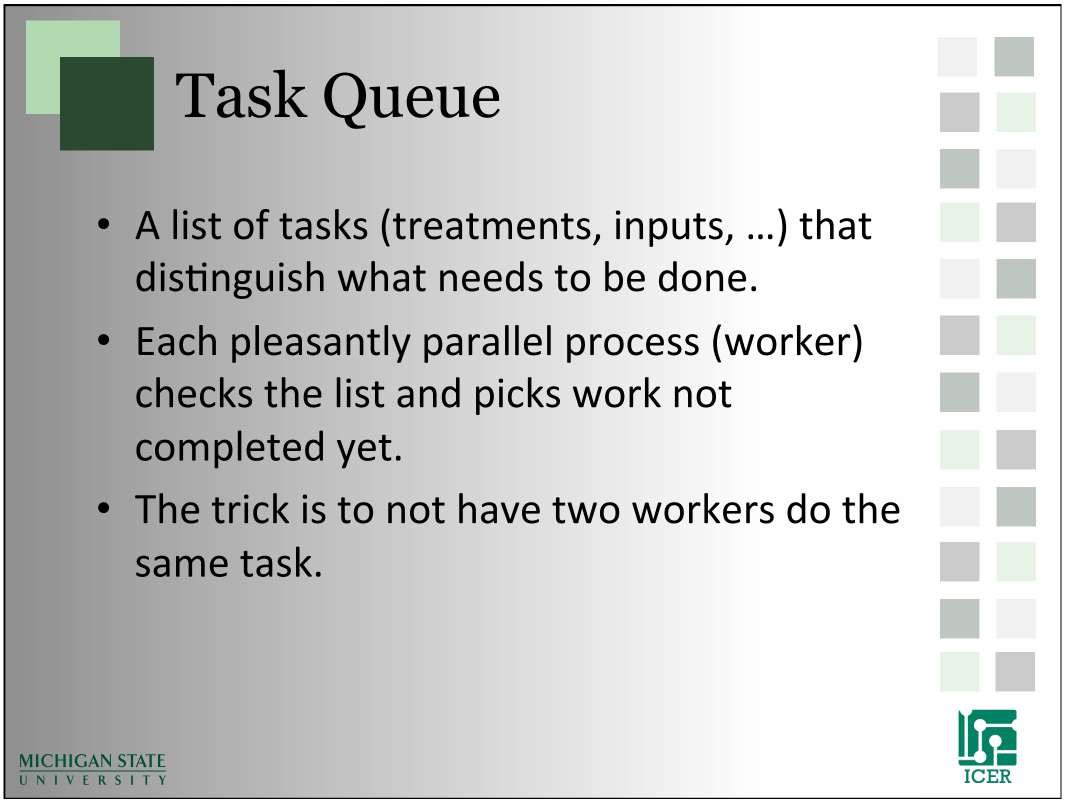
Unrolling Loops

- Your program has independent loops
  - Each iteration of the loop does not depend on the other iterations
  - Loop can be executed in any order
  - 5 Minutes < Iteration Time < 1 week
  - Output of each iteration must be easy to save and recombine for next step of workflow
- Rewrite your program to accept an iteration number as an input
  - ./myprogram IterationNumber
- Rewrite your program to save output and use an additional program for post processing



Simple Unrolled Loop

```
#!/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}

./myprogram ${PBS_ARRAYID}

qstat -f ${PBS_JOBID}
```



Task Queue

- A list of tasks (treatments, inputs, …) that distinguish what needs to be done.
- Each pleasantly parallel process (worker) checks the list and picks work not completed yet.
- The trick is to not have two workers do the same task.

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)

MICHIGAN STATE
UNIVERSITY

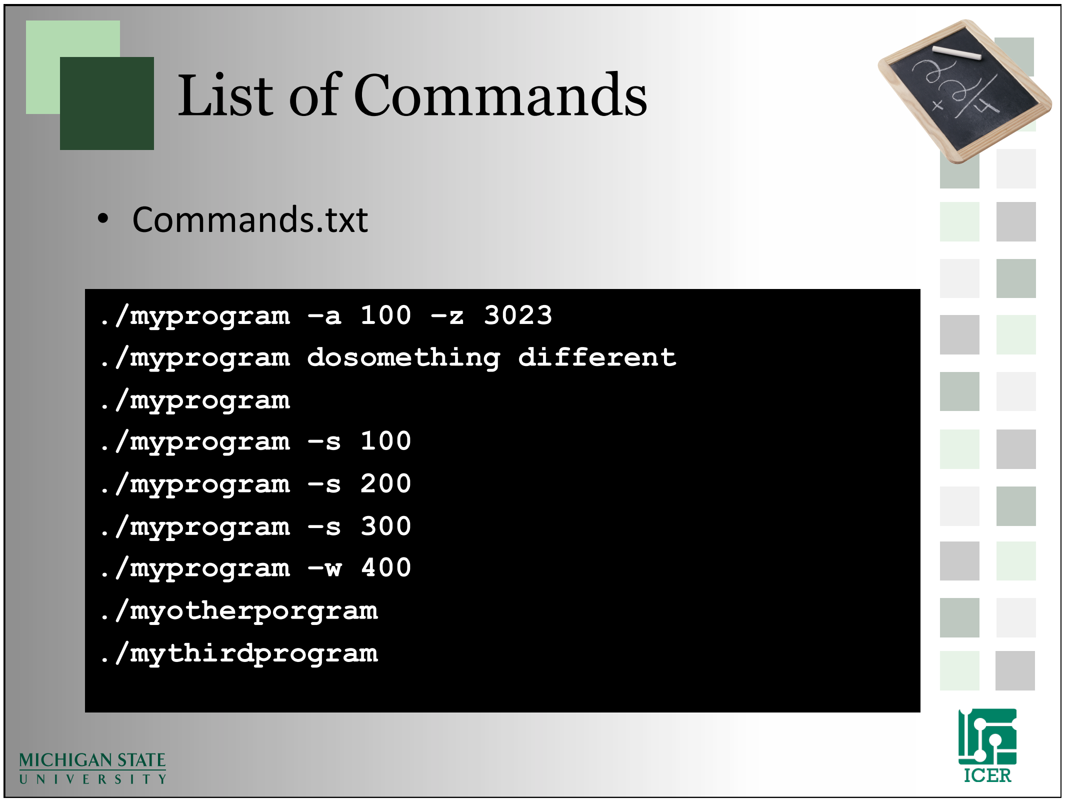
ICER

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
done
```

MICHIGAN STATE
UNIVERSITY

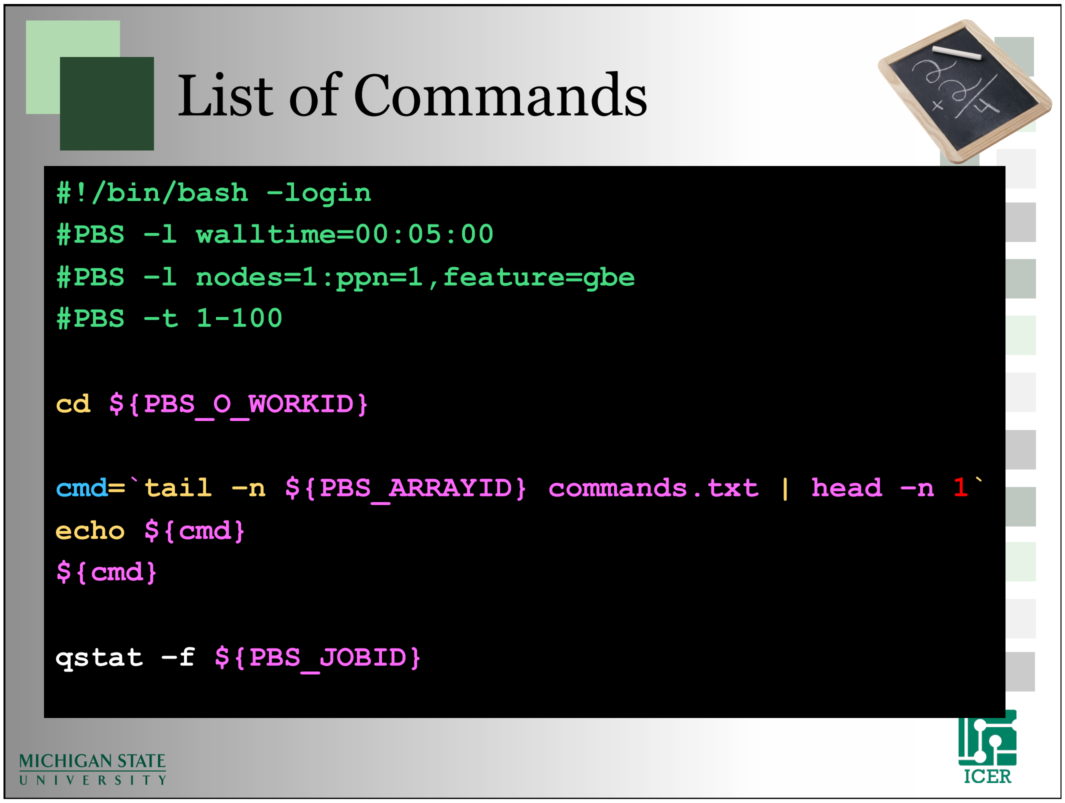
ICER



List of Commands

- Commands.txt

```
./myprogram -a 100 -z 3023
./myprogram dosomething different
./myprogram
./myprogram -s 100
./myprogram -s 200
./myprogram -s 300
./myprogram -w 400
./myotherporgram
./mythirdprogram
```

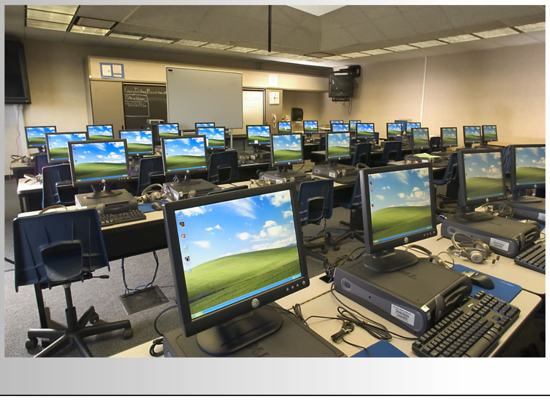


List of Commands

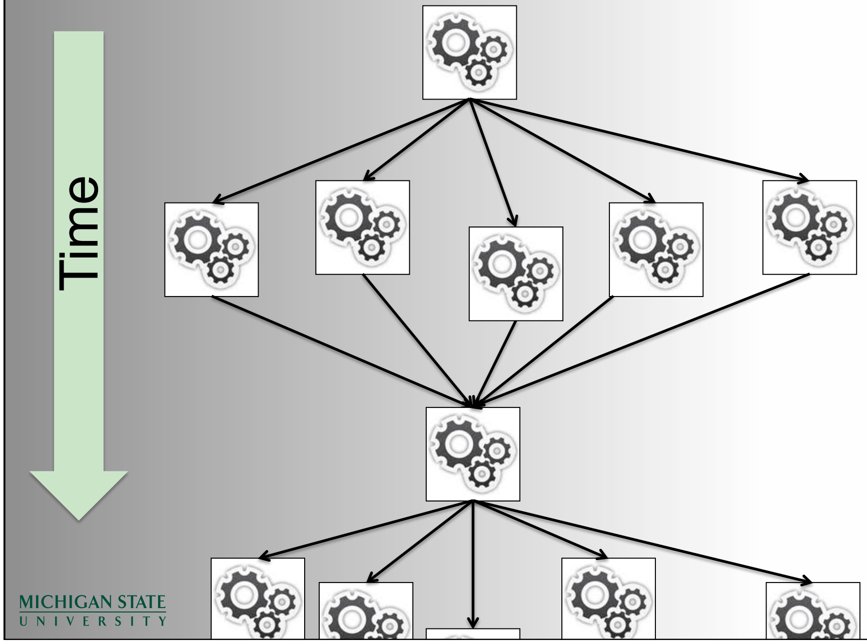
```
#!/bin/bash -login
#PBS -l walltime=00:05:00
#PBS -l nodes=1:ppn=1,feature=gbe
#PBS -t 1-100

cd ${PBS_O_WORKID}

cmd=`tail -n ${PBS_ARRAYID} commands.txt | head -n 1`
echo ${cmd}
${cmd}

qstat -f ${PBS_JOBID}
```

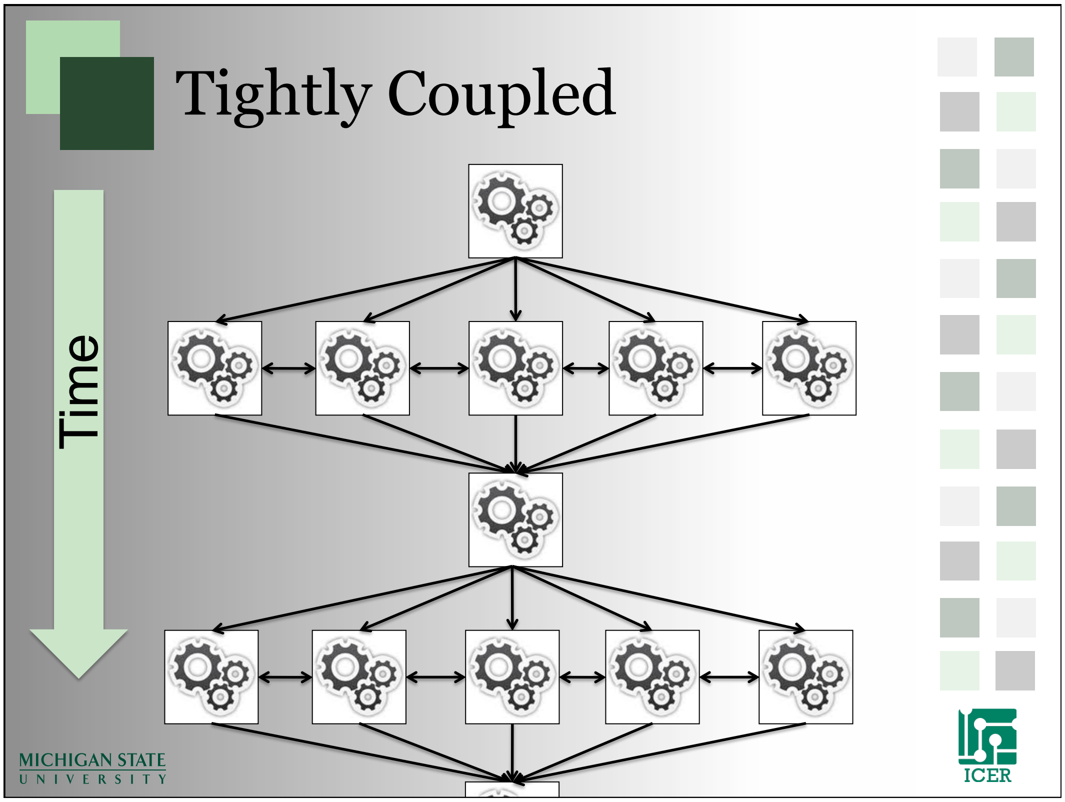
Condor High Throughput Computing

- Job submission system
- Runs like a screen saver
- Steals CPU Cycles

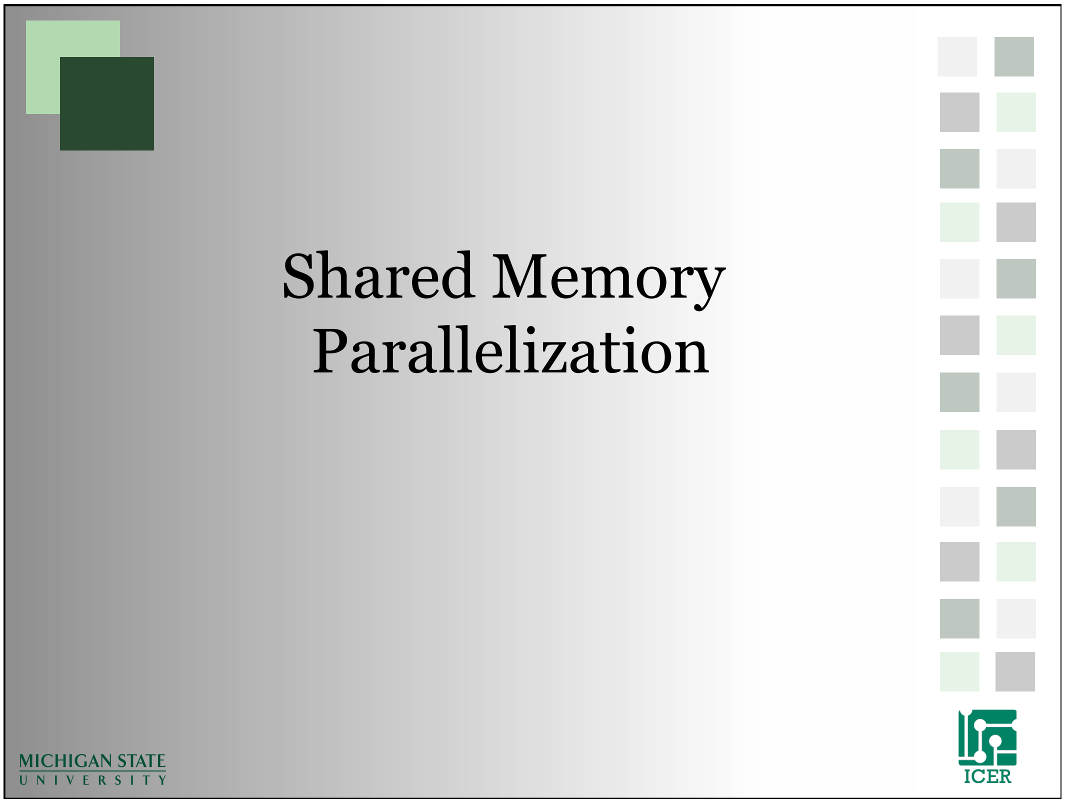


Loosely Coupled

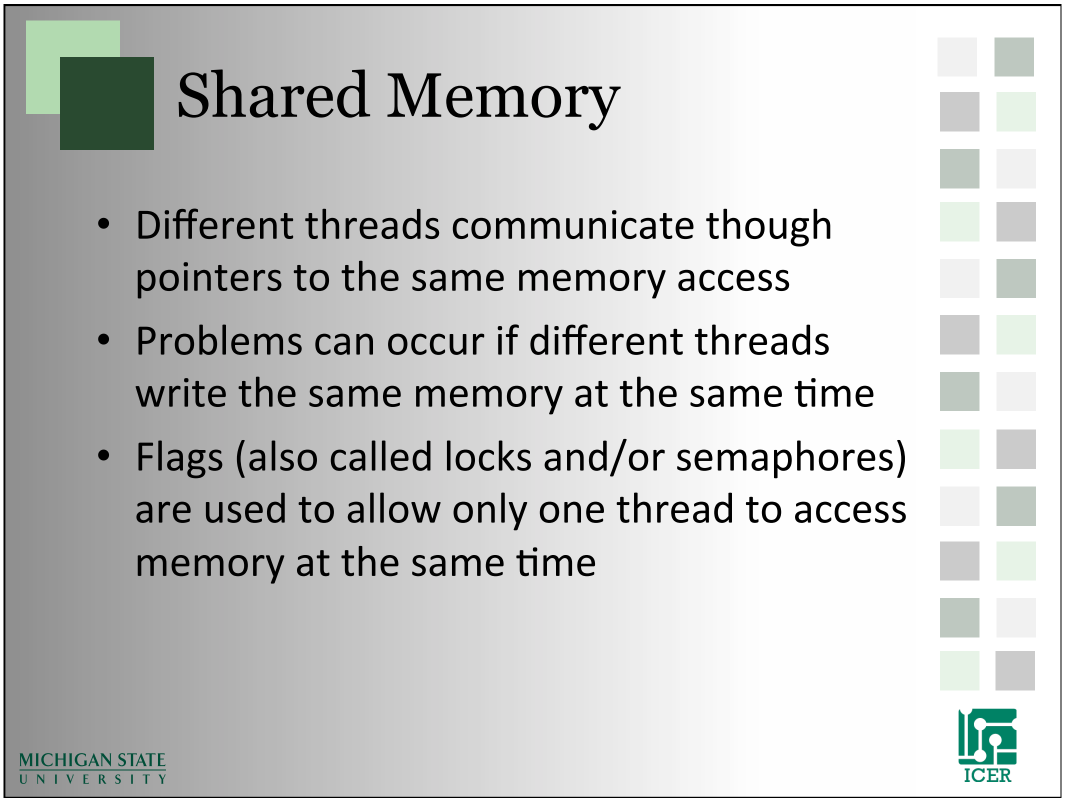




Tightly Coupled



Shared Memory Parallelization

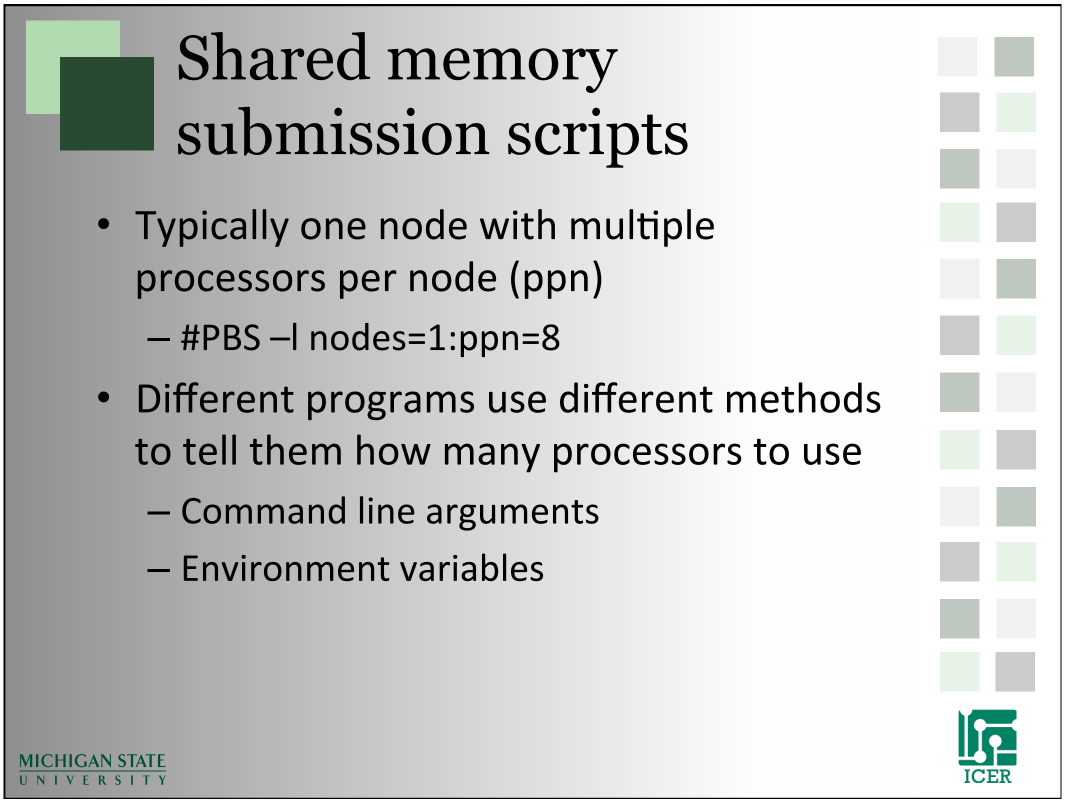


Shared Memory

- Different threads communicate though pointers to the same memory access
- 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

MICHIGAN STATE UNIVERSITY

ICER

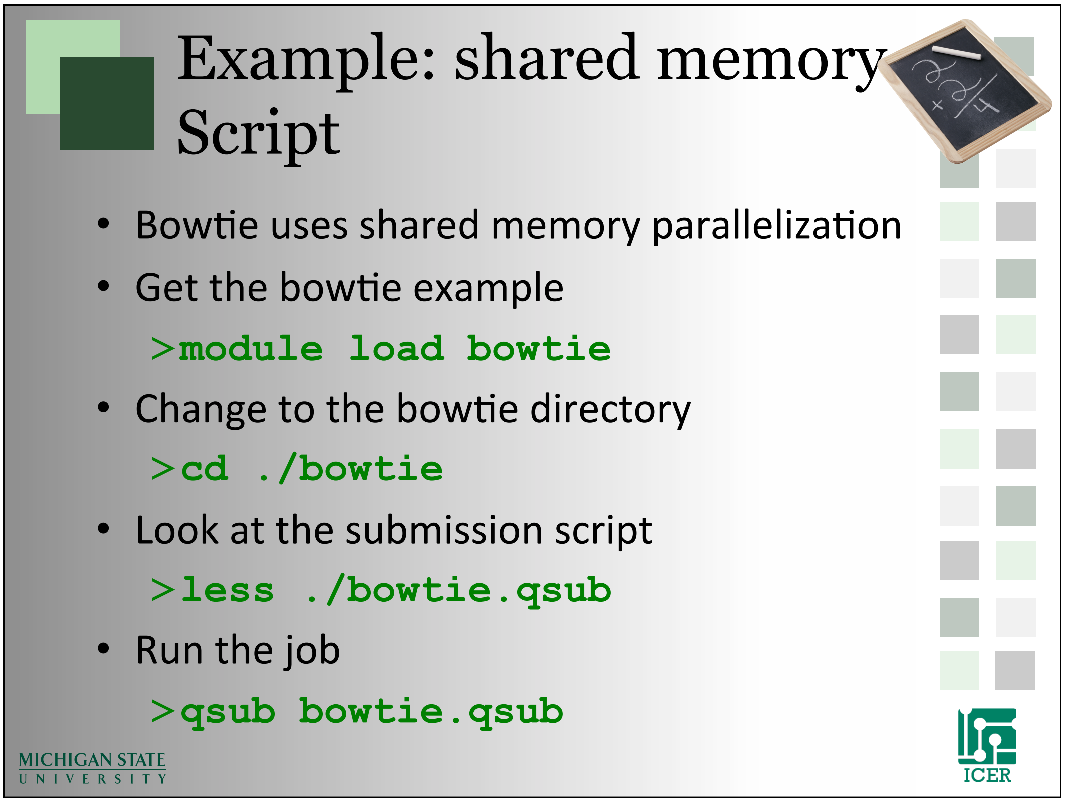


Shared memory submission scripts

- 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

MICHIGAN STATE UNIVERSITY

ICER



Example: shared memory Script

- Bowtie uses shared memory parallelization
- Get the bowtie example

```
>module load bowtie
```

- Change to the bowtie directory

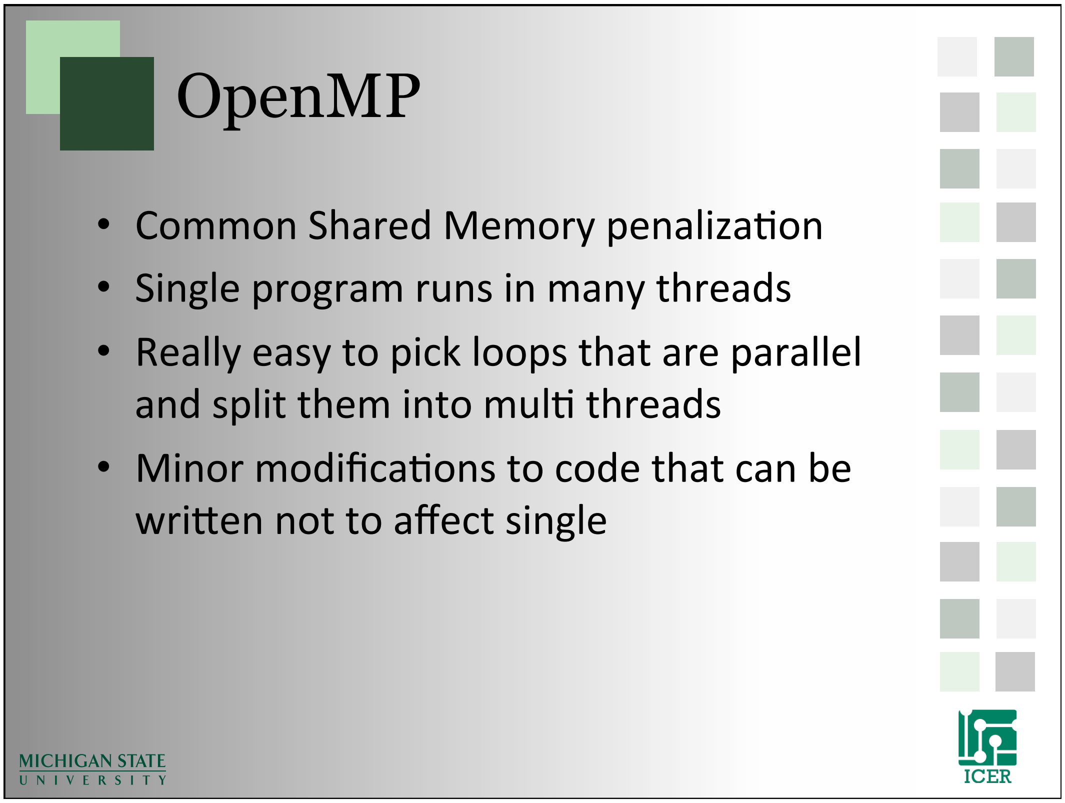
```
>cd ./bowtie
```

- Look at the submission script

```
>less ./bowtie.qsub
```

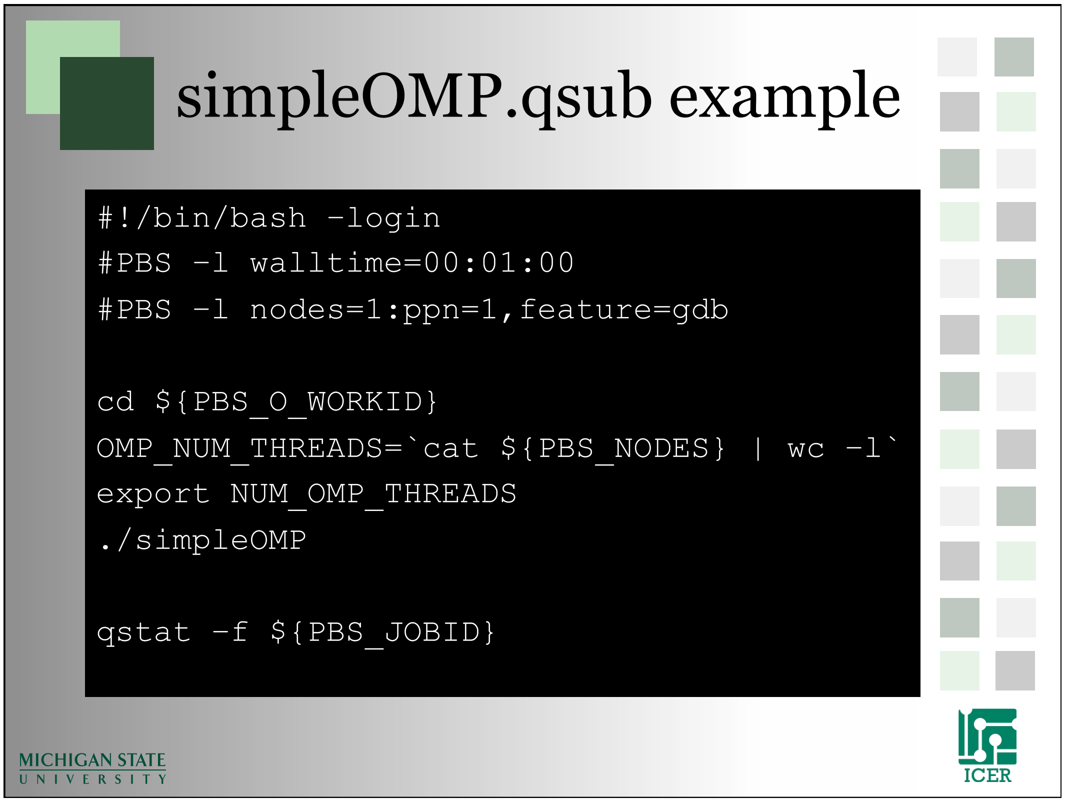
- Run the job

```
>qsub bowtie.qsub
```



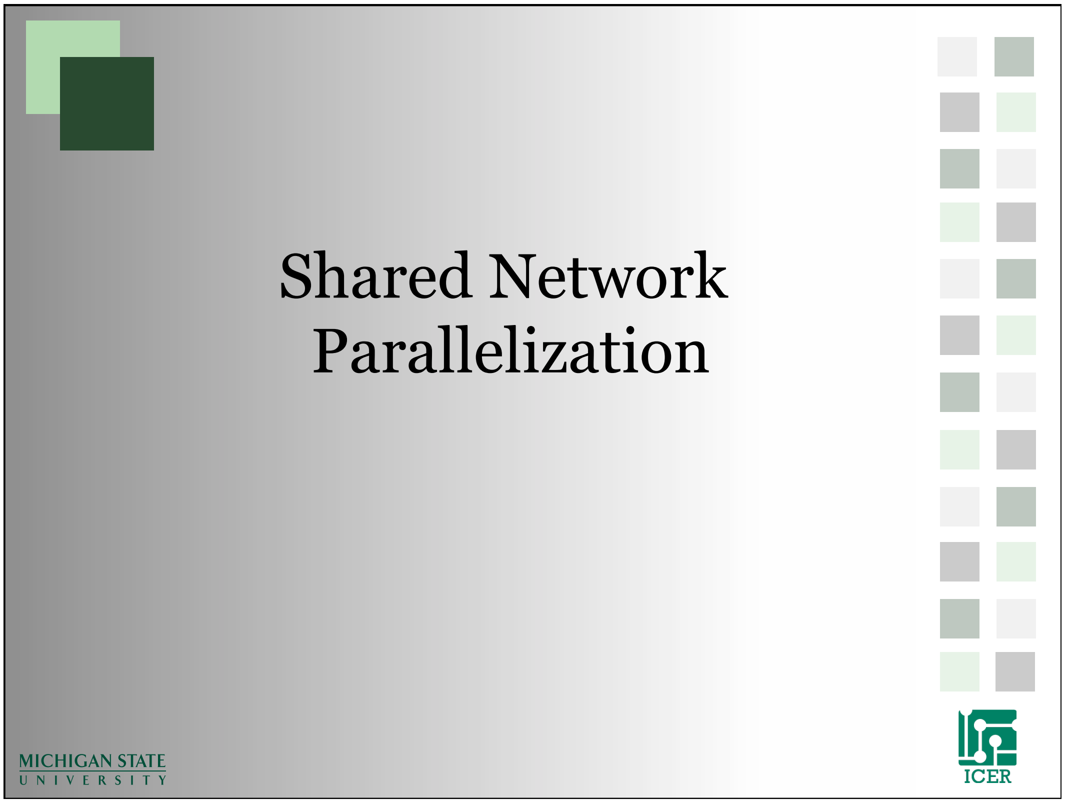
OpenMP

- Common Shared Memory penalization
- 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

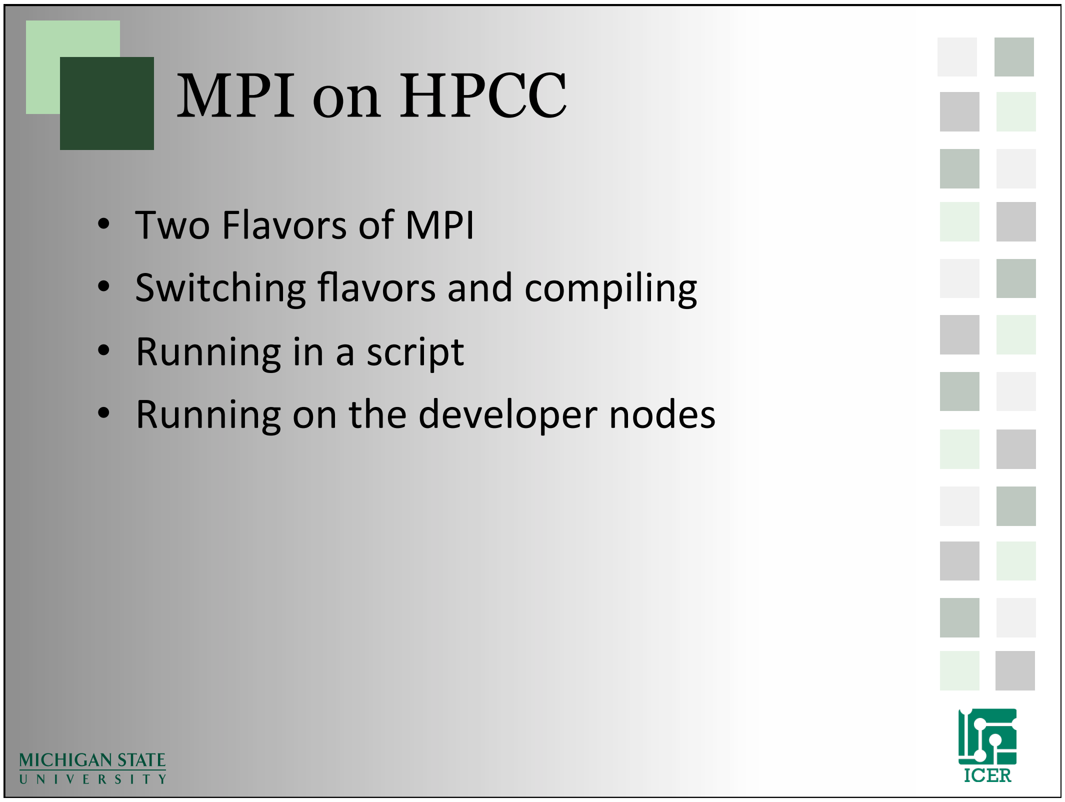


simpleOMP.qsub example

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

cd ${PBS_O_WORKID}
OMP_NUM_THREADS=`cat ${PBS_NODES} | wc -l`
export NUM_OMP_THREADS
./simpleOMP

qstat -f ${PBS_JOBID}
```

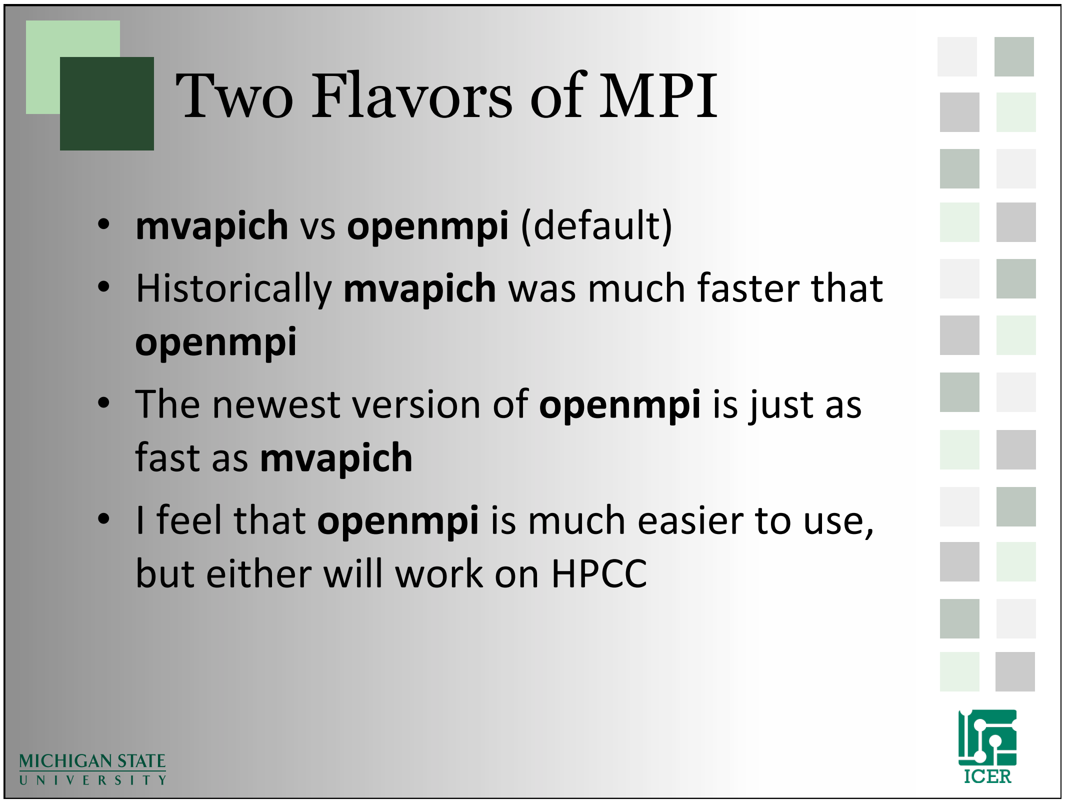


Shared Network Parallelization



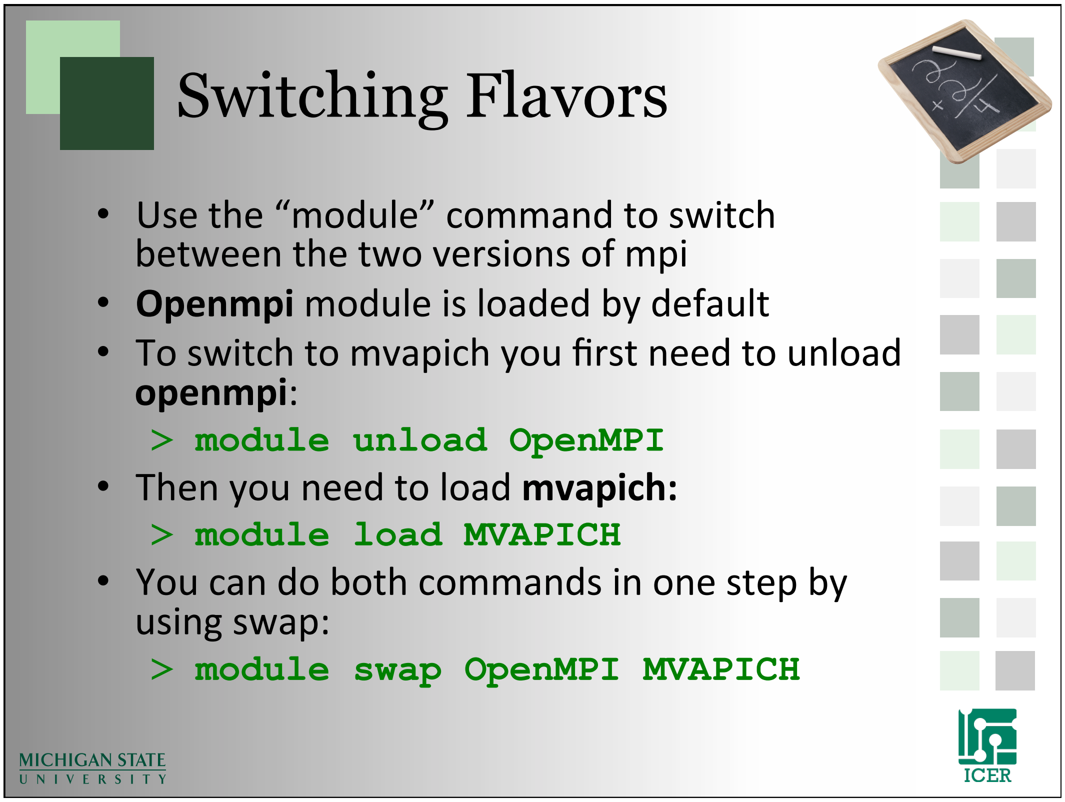
MPI on HPCC

- Two Flavors of MPI
- Switching flavors and compiling
- Running in a script
- Running on the developer nodes



Two Flavors of MPI

- **mvapich** vs **openmpi** (default)
- Historically **mvapich** was much faster that **openmpi**
- The newest version of **openmpi** is just as fast as **mvapich**
- I feel that **openmpi** is much easier to use, but either will work on HPCC



Switching Flavors

- Use the “module” command to switch between the two versions of mpi
- **Openmpi** module is loaded by default
- To switch to mvapich you first need to unload **openmpi**:

```
> module unload OpenMPI
```

- Then you need to load **mvapich:**

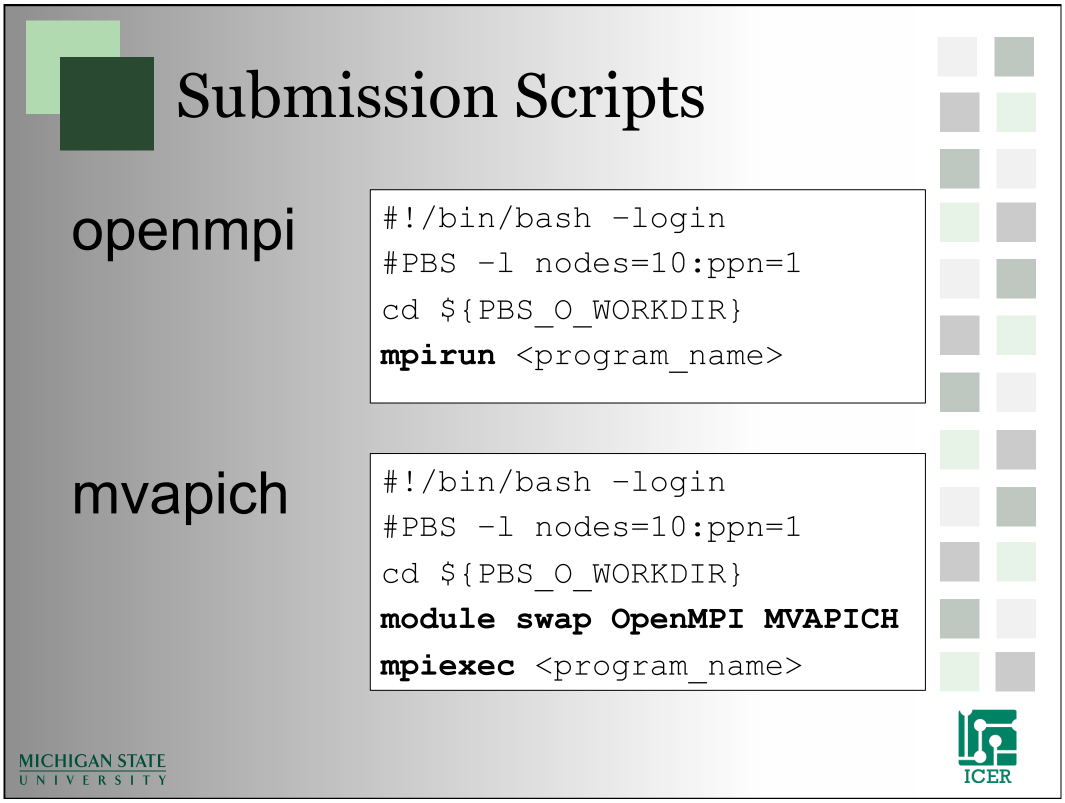
```
> module load MVAPICH
```

- You can do both commands in one step by using swap:

```
> module swap OpenMPI MVAPICH
```

MICHIGAN STATE UNIVERSITY

ICER



Submission Scripts

openmpi

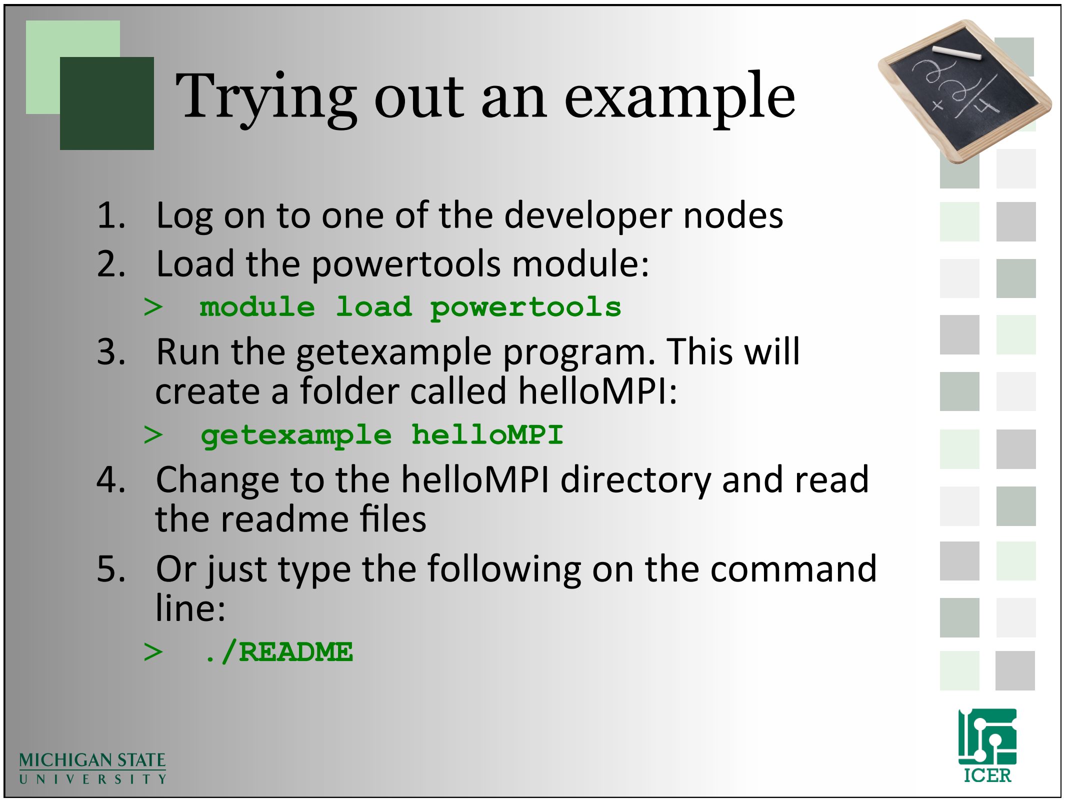
```
#!/bin/bash -login
#PBS -l nodes=10:ppn=1
cd ${PBS_O_WORKDIR}
mpirun <program_name>
```

mvapich

```
#!/bin/bash -login
#PBS -l nodes=10:ppn=1
cd ${PBS_O_WORKDIR}
module swap OpenMPI MVAPICH
mpiexec <program_name>
```

MICHIGAN STATE UNIVERSITY

ICER



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:

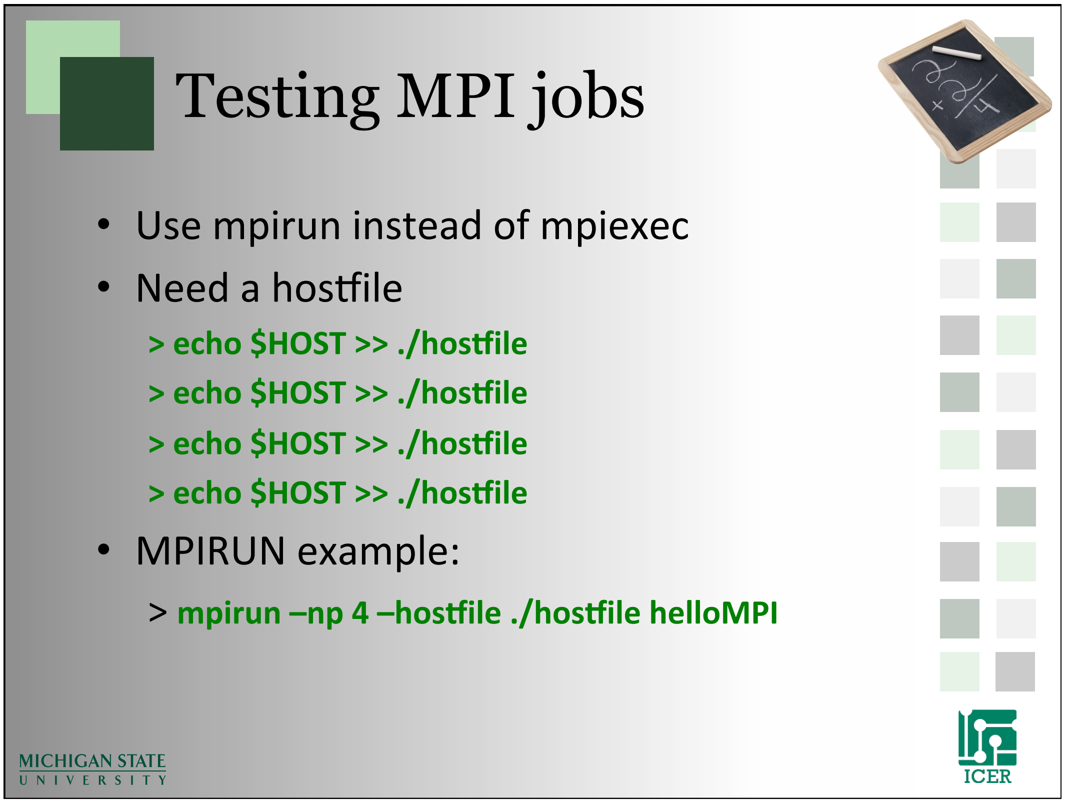
```
> getexample helloMPI
```

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

```
> ./README
```

MICHIGAN STATE UNIVERSITY

ICER



Testing MPI jobs

- Use mpirun instead of mpiexec
- Need a hostfile

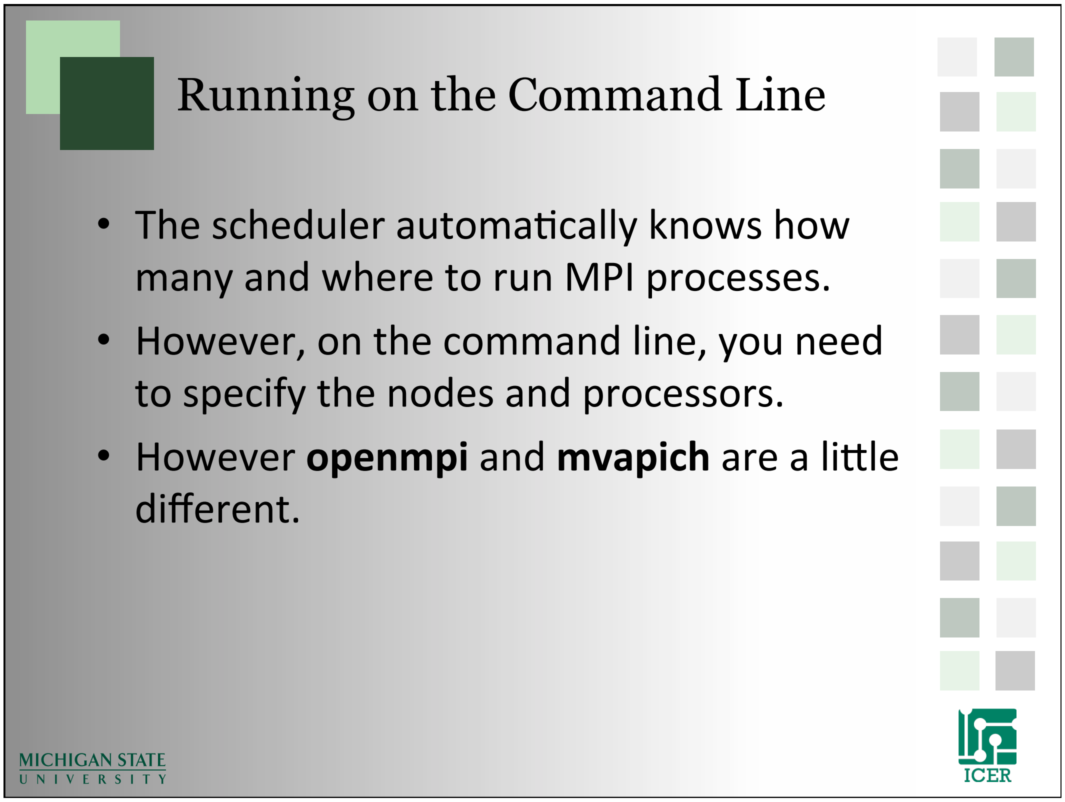
```
> echo $HOST >> ./hostfile
> echo $HOST >> ./hostfile
> echo $HOST >> ./hostfile
> echo $HOST >> ./hostfile
```

- MPIRUN example:

```
> mpirun –np 4 –hostfile ./hostfile helloMPI
```

MICHIGAN STATE UNIVERSITY

ICER

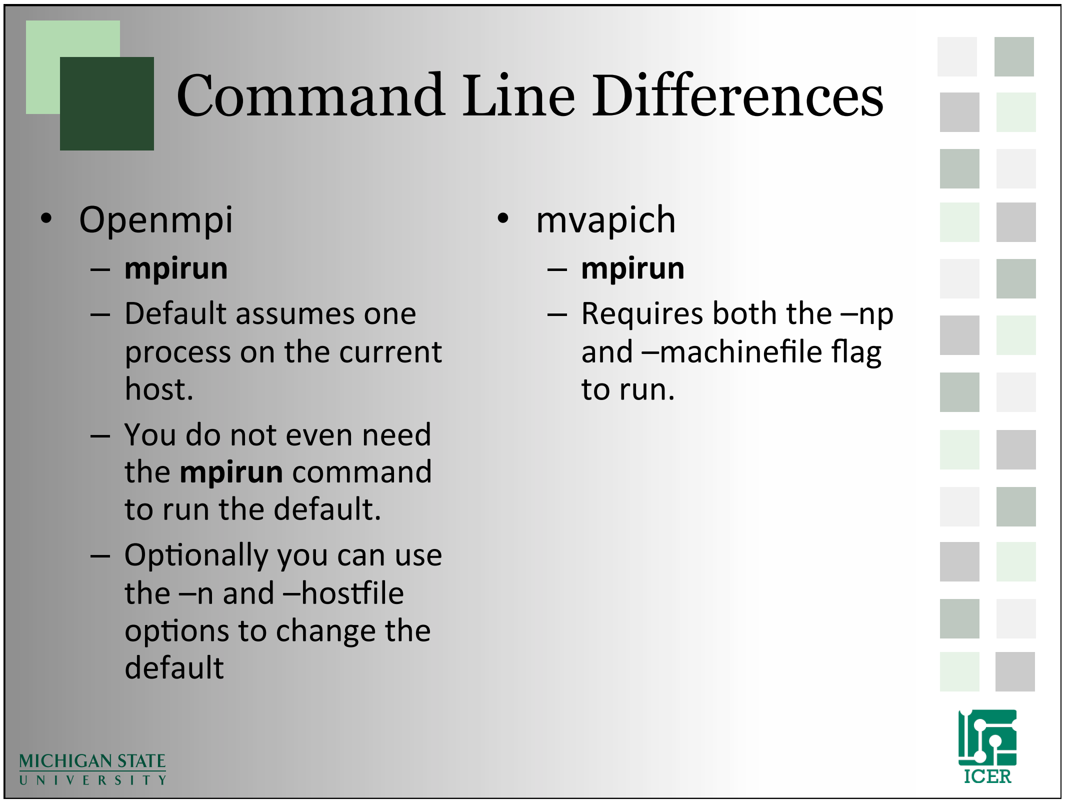


Running on the Command Line

- The scheduler automatically knows how many and where to run MPI processes.
- However, on the command line, you need to specify the nodes and processors.
- However **openmpi** and **mvapich** are a little different.

MICHIGAN STATE UNIVERSITY

ICER

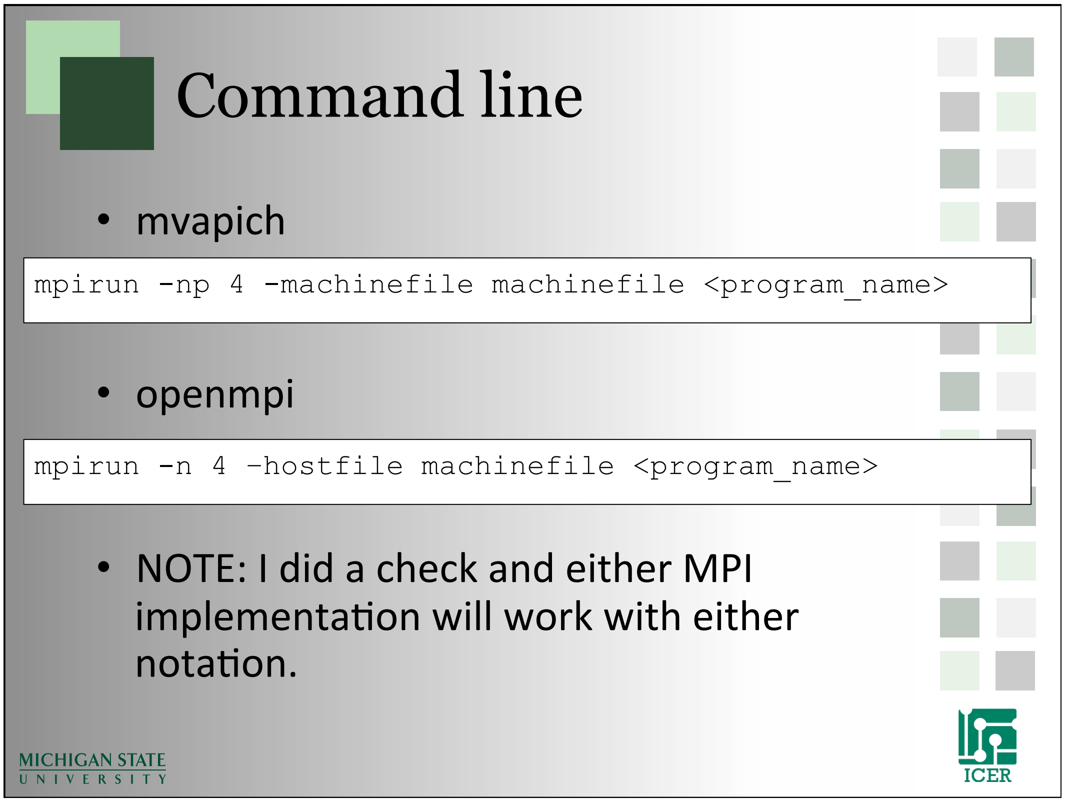


Command Line Differences

- Openmpi
  - **mpirun**
  - Default assumes one process on the current host.
  - You do not even need the **mpirun** command to run the default.
  - Optionally you can use the –n and –hostfile options to change the default
- mvapich
  - **mpirun**
  - Requires both the –np and –machinefile flag to run.

MICHIGAN STATE UNIVERSITY

ICER



Command line

- mvapich

```
mpirun -np 4 -machinefile machinefile <program_name>
```

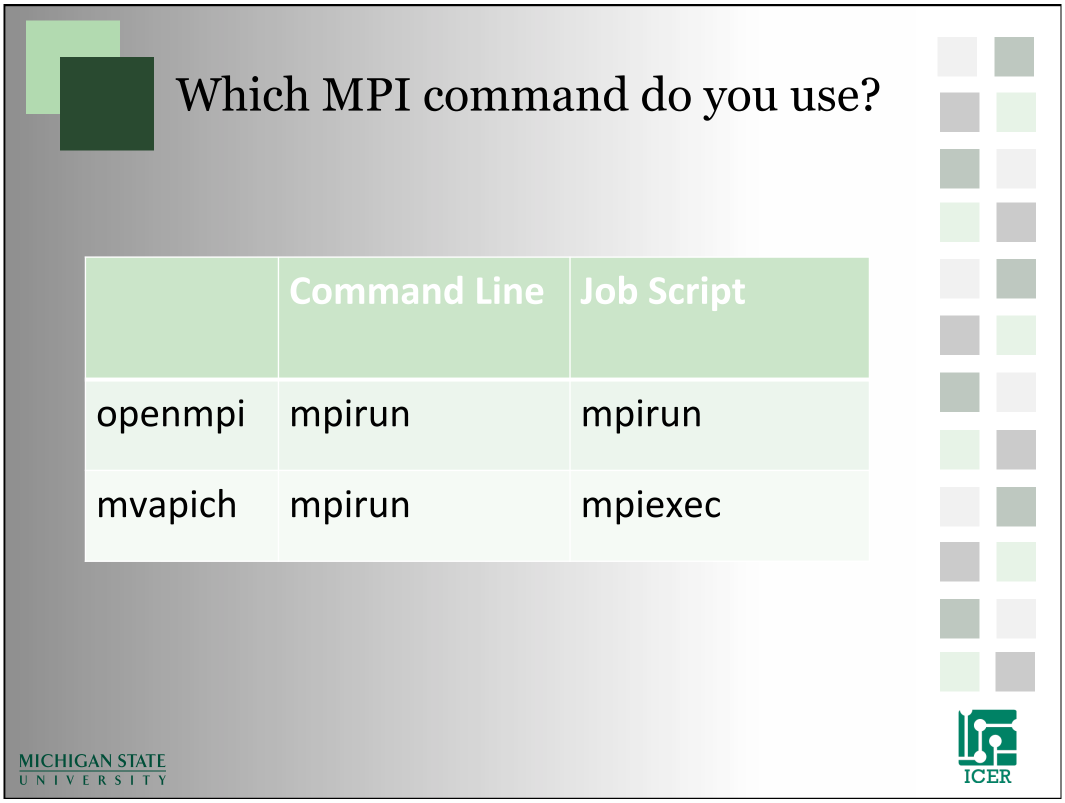
- openmpi

```
mpirun -n 4 –hostfile machinefile <program_name>
```

- NOTE: I did a check and either MPI implementation will work with either notation.

MICHIGAN STATE UNIVERSITY

ICER

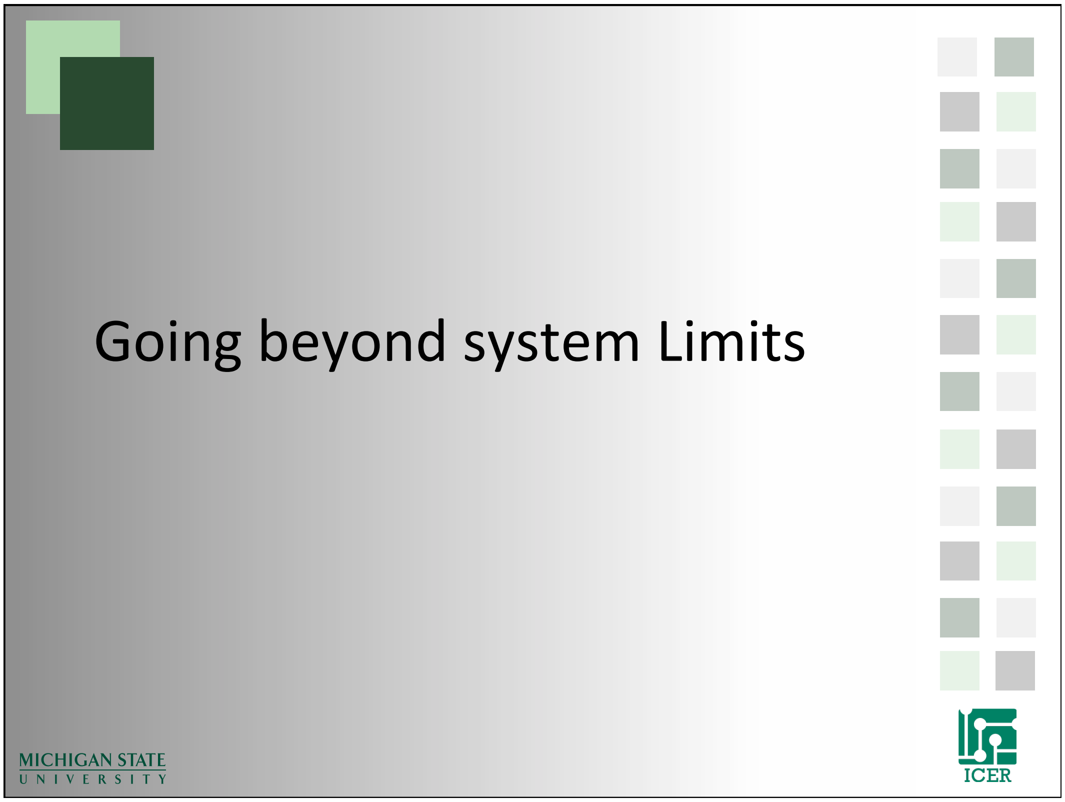


Which MPI command do you use?

	Command Line	Job Script
openmpi	mpirun	mpirun
mvapich	mpirun	mpiexec

MICHIGAN STATE UNIVERSITY

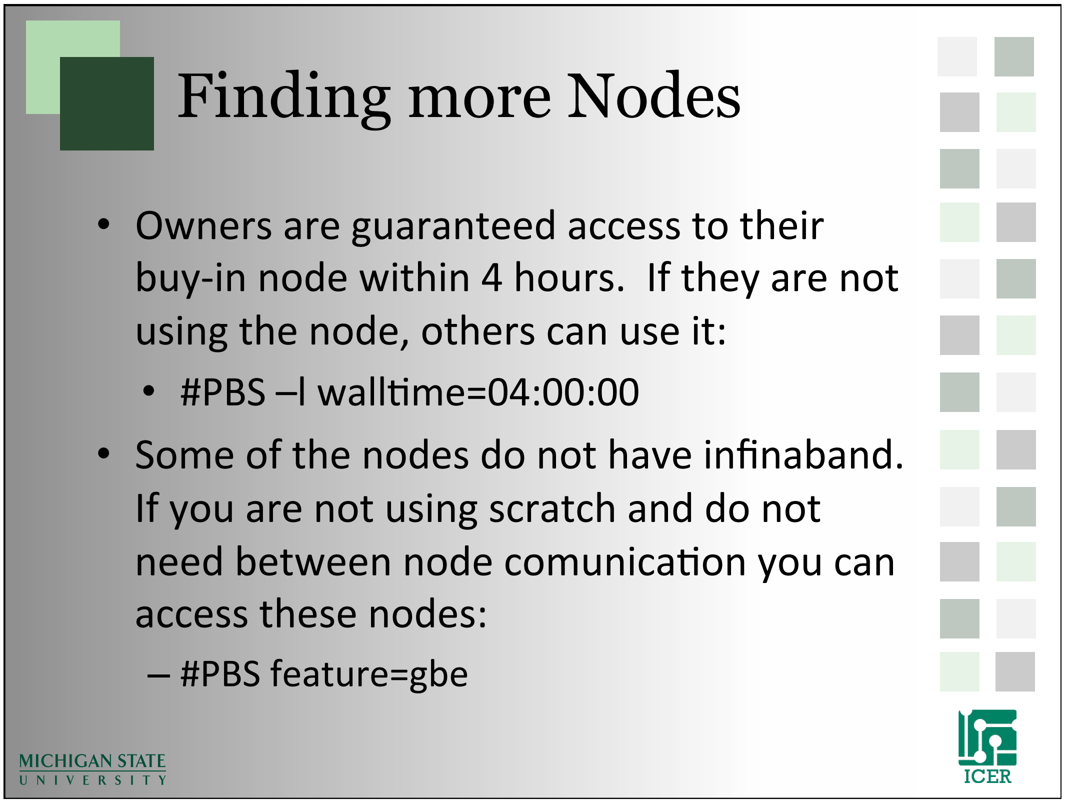
ICER



Going beyond system Limits

MICHIGAN STATE UNIVERSITY

ICER

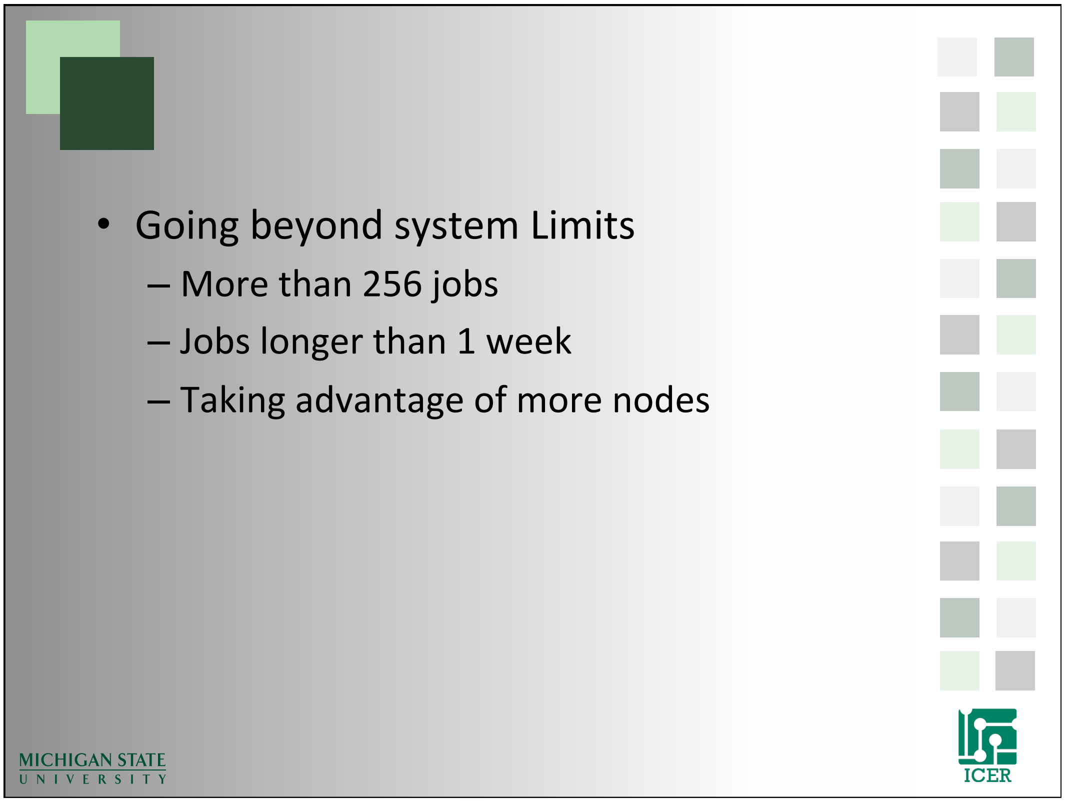


Finding more Nodes

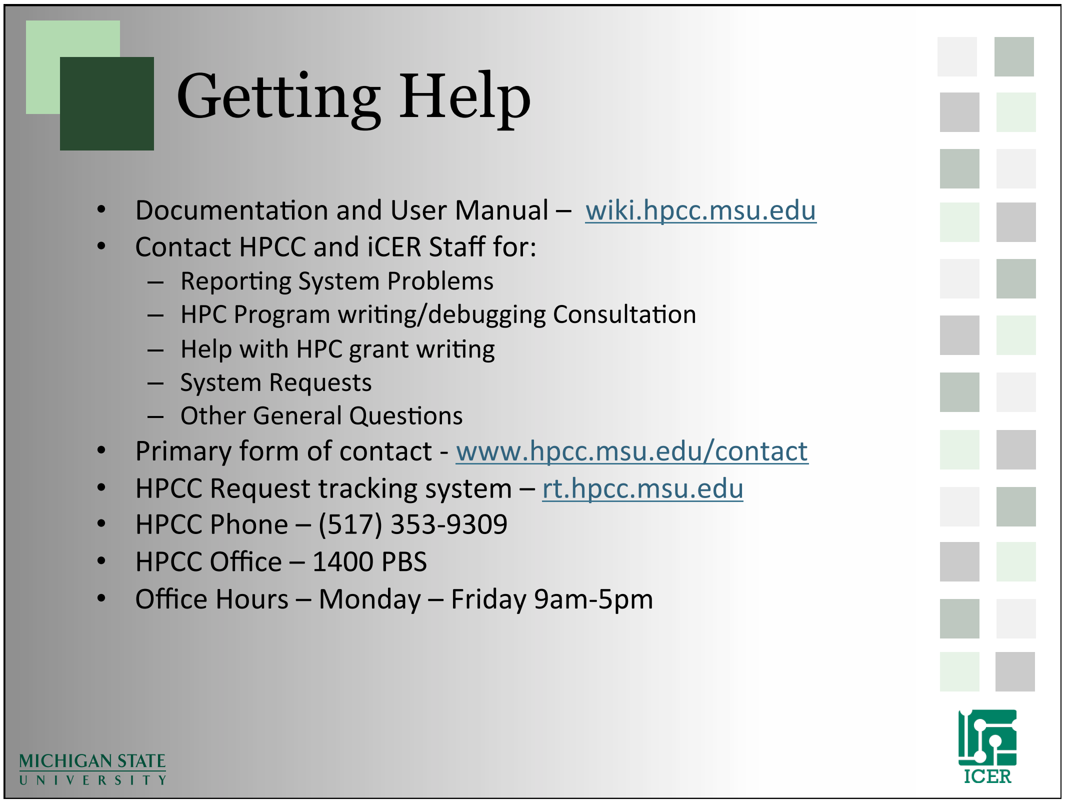
- Owners are guaranteed access to their buy-in node within 4 hours. If they are not using the node, others can use it:
  - #PBS –l walltime=04:00:00
- Some of the nodes do not have infinaband. If you are not using scratch and do not need between node comunication you can access these nodes:
  - #PBS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

MICHIGAN STATE UNIVERSITY

ICER



- Going beyond system Limits
  - More than 256 jobs
  - Jobs longer than 1 week
  - Taking advantage of more nodes



Getting Help

- Documentation and User Manual – wiki.hpcc.msu.edu
- Contact HPCC and iCER Staff for:
  - Reporting System Problems
  - HPC Program writing/debugging Consultation
  - Help with HPC grant writing
  - System Requests
  - Other General Questions
- Primary form of contact - www.hpcc.msu.edu/contact
- HPCC Request tracking system – rt.hpcc.msu.edu
- HPCC Phone – (517) 353-9309
- HPCC Office – 1400 PBS
- Office Hours – Monday – Friday 9am-5pm