Chalmers Centre *for* Computational Science and Engineering

Chalmers University of Technology — Göteborg University

How to use Matlab in a HPC environment?

Hugo U. R. Strand
`hugo.strand@chalmers.se`

May 4, 2011

Outline

Matlab

What is Matlab?

- Matrix Laboratory
- Developed by MathWorks
- Scripting of Numerical Algorithms
- Solving real world problems quick and easy
- Included in all Batchelors programmes at Chalmers

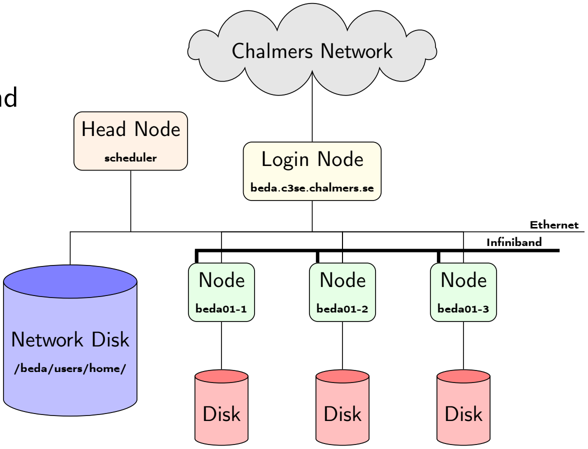
This presentation assumes that you have previous Matlab experience.

Q: Are you an experienced Matlab user?

What is a High Performance Computing Cluster?

A collection of closely connected hardware and software components

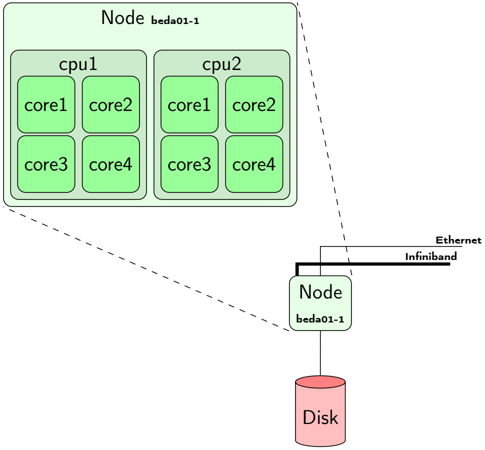
- Login Node
- Compute Nodes
- Network
- Server Node
- Storage



What is a High Performance Computing Cluster?

A collection of closely connected hardware and software components

- Login Node
- Compute Nodes
    - Multi CPU & multi CPU cores
    - Local disk
    - Infiniband node-node communication
- Network
- Server Node
- Storage



Cluster vs. Workstation

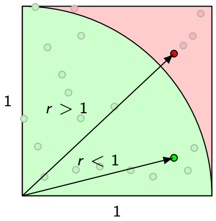
Workstation	Compute Cluster
It's all yours. Powering on/off, upgrading software etc, affects only you	Hundreds of users with varying requirements, needs, dead lines, working hours
It's all yours. You can use to computer as you see fit	CPU, memory, disk and network needs to be shared
You can start or stop any calculations at any time	A request for running a calculation **implies a request for allocating a piece of the resource**
You have the possibility to start or control calculations interactively at any time	The exact details for how to run the calculation **must be specified ahead of execution time using a "batch script"**

Practical Example

- Learning by doing
- Start with simple calculation problem
- Move calculation to the cluster
- Discuss pitfalls and solutions
- Some parallelization

Sochastic π calculation

- Approximate pi by random "throws" in the unit square.
- Ratio of hits $N_{\bullet}$ inside the quarter unit circle and total throws $N_{\bullet+\bullet}$ gives π approximation



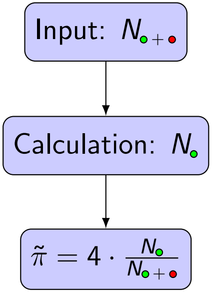
$$\frac{A_{\circ}}{A_{\square}} = \frac{\frac{1}{4}\pi r^2}{r^2} \approx \frac{N_{\bullet}}{N_{\bullet+\bullet}}$$

$$\pi \approx 4 \cdot \frac{N_{\bullet}}{N_{\bullet+\bullet}}$$

- Do $N_{\bullet+\bullet}$ times

```
Draw two random numbers $x, y \in [0, 1]$
If $x^2 + y^2 < 1$
  $N_{\bullet} = N_{\bullet} + 1$
```

Implementation Example



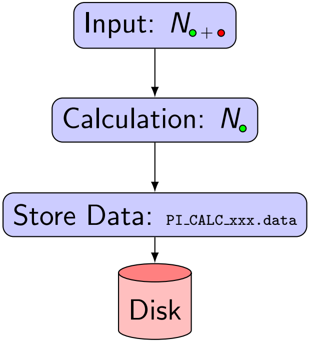
pi_calc.m

```matlab
function [N_hits, seed] = pi_calc(N)

seed = sum(100*clock);
RandStream.setDefaultStream(...
    RandStream('mt19937ar','seed',seed));

N_hits = 0;
for iter = 1:N
    coord = rand(1, 2);
    r2 = sum(coord .* coord, 2);
    N_hits = N_hits + sum(r2 < 1.0);
end

pi_approx = 4.0 * N_hits/N
```

Matlab Output

```
>> [N_hit, seed] = pi_calc(10000);
pi_approx =
   3.140400000000000
>>
```

Store to file


Input: $N_{\bullet + \bullet}$
Calculation: $N_{\bullet}$
Store Data: PI_CALC_xxx.data
Disk

pi_calc_tofile.m

```
function pi_calc_tofile(N)

[N_hits, seed] = pi_calc(N);

data = struct('N', N, 'N_hits', N_hits);
store_data(data, 'PI_CALC');
```

Matlab Output

```
>> pi_calc_tofile(10000);
pi_approx =
    3.1728
>> ls PI*
PI_CALC_pid2566_beda.nfs.private_...
2011-05-02_22:09:26.data
```

Store to file

- Store a matlab struct to file
- Need unique file name to avoid overwriting
  - Host Name, unique for each node
  - Process ID (PID), unique for each process
  - Time, why not..
- Store binary format `'-mat'`

store_data.m

```
function store_data(data, fileheader)

[~, hostname] = system('hostname');
hostname = hostname(1:end-1);

[~, pid] = system('ps -p $$ -o ppid=');
pid = pid(2:end-1);

[~, date] = system('date +%F_%T');
date = date(1:end-1);

filename = sprintf( ...
    '%s_pid%s_%s_%s.data', ...
    fileheader, pid, hostname, date);

fprintf('Saving data to:\n%s\n', filename);
save(filename, 'data', '-mat');
```

Submit and run!?

Disclaimer:
DO NOT USE THIS EXAMPLE!

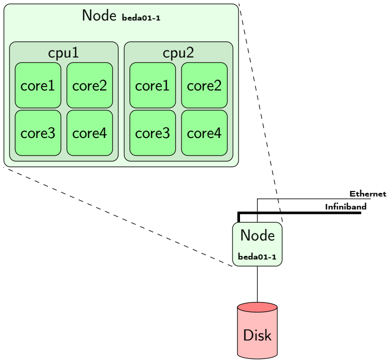
submit_job_simple.sh

```
#!/usr/bin/env bash
#PBS -A C3SE-STAFF
#PBS -q beda
#PBS -l walltime=01:00:00
#PBS -l nodes=1:ppn=8
#PBS -o stdout
#PBS -e stderr

module load matlab/7.10

cd $PBS_O_WORKDIR

flags='-nodisplay -singleCompThread'
cmd='pi_calc_tofile(1e6); quit'

matlab ${flags} -r "${cmd}"
```

Why is this bad use of resources?

Submit and run!?

☠ Only using one CPU core



Disclaimer:
DO NOT USE THIS EXAMPLE!

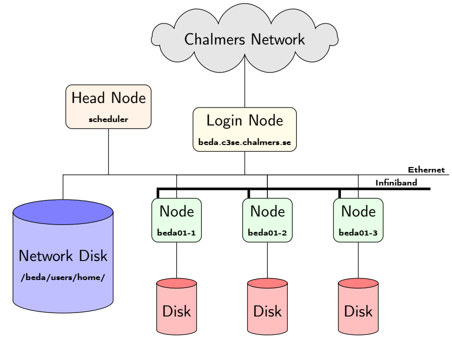
submit_job_simple.sh

```
#!/usr/bin/env bash
#PBS -A C3SE-STAFF
#PBS -q beda
#PBS -l walltime=01:00:00
#PBS -l nodes=1:ppn=8
#PBS -o stdout
#PBS -e stderr

module load matlab/7.10

cd $PBS_O_WORKDIR

flags='-nodisplay -singleCompThread'
cmd='pi_calc_tofile(1e6); quit'

matlab ${flags} -r "${cmd}"
```

Why is this bad use of resources?

Submit and run!?

- ☠ Only using one CPU core
- ☠ File IO to Network Disk
  `$PBS_O_WORKDIR`



Disclaimer:
DO NOT USE THIS EXAMPLE!

submit_job_simple.sh

```
#!/usr/bin/env bash
#PBS -A C3SE-STAFF
#PBS -q beda
#PBS -l walltime=01:00:00
#PBS -l nodes=1:ppn=8
#PBS -o stdout
#PBS -e stderr

module load matlab/7.10

cd $PBS_O_WORKDIR

flags='-nodisplay -singleCompThread'
cmd='pi_calc_tofile(1e6); quit'

matlab ${flags} -r "${cmd}"
```

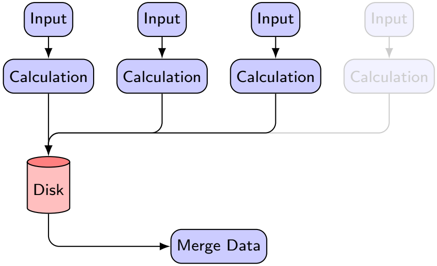
Why is this bad use of resources?

Can we do better?

- Our calculation is trivially parallelizable

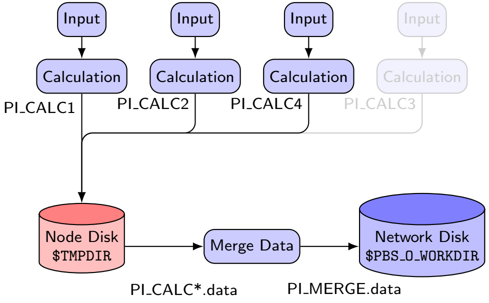
$$\pi \approx 4 \cdot \frac{N_{\bullet}}{N_{\bullet+\bullet}} \quad \Leftrightarrow \quad \pi \approx 4 \cdot \frac{\sum_i N_{\bullet,i}}{\sum_i N_{\bullet+\bullet,i}},$$

  where $N_{\bullet,i}$ and $N_{\bullet+\bullet,i}$ are from the i:th independent calculation.
- Perform independent calculations of, $N_{\bullet,i}$, and store to disk.



- Similar approach in the case of parameter sweeps

Post Processing



merge_files.m

```
[N, N_hits] = post_process('PI_CALC');
data = struct('N', N, 'N_hits', N_hits);
store_data(data, 'PI_MERGE');
```

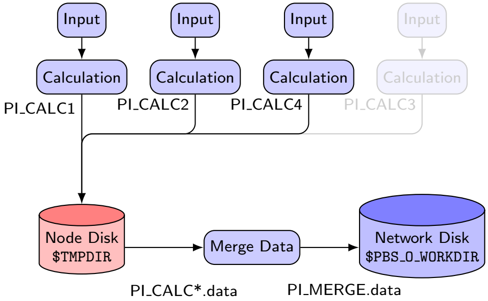
post_process.m

```
function [N, N_hits] = post_process(fileheader)

[~, out] = system(sprintf('ls %s*', fileheader));
c = textscan(out, '%s');
file_list = c{1};

N = 0; N_hits = 0;
disp('Loading files:');
for i = 1:length(file_list)
    file_name = file_list{i};
    fprintf('%s\n', file_name);
    file = load(file_name, '-mat', 'data');

    N = N + file.data.N;
    N_hits = N_hits + file.data.N_hits;
end

pi_approx = 4.0 * N_hits / N

end
```

Matlab Output

```
>> ls PI_CALC*
PI_CALC_pid0767_beda..._2011-05-03_16:10:23.data
PI_CALC_pid0767_beda..._2011-05-03_17:25:30.data
>> merge_files
Loading files:
PI_CALC_pid0767_beda..._2011-05-03_16:10:23.data
PI_CALC_pid0767_beda..._2011-05-03_17:25:30.data
pi_approx = 3.1412
Saving data to:
PI_MERGE_pid0767_beda..._2011-05-03_17:25:58.data
```

Submit some jobs to the cluster



- Copy files from
  - Network ($PBS_O_WORKDIR) to
  - Node ($TMPDIR)
- Run one matlab session per core
- Merge data on $TMPDIR
- Copy merged data back to $PBS_O_WORKDIR

submit_job.sh

```
#!/usr/bin/env bash
#PBS -A C3SE-STAFF
#PBS -q beda
#PBS -l walltime=01:00:00
#PBS -l nodes=1:ppn=8
#PBS -o stdout
#PBS -e stderr
module load matlab/7.10

cp $PBS_O_WORKDIR/*.m $TMPDIR
cd $TMPDIR

flags='-nodisplay -singleCompThread'
cmd='pi_calc_tofile(1e6); quit'

for i in {1..8}
do
    matlab ${flags} -r "${cmd}" &
    sleep 0.1
done

wait # Wait for all processes

matlab ${flags} -r 'merge_files'
rm $TMPDIR/PI_CALC*.data

cp $TMPDIR/PI_MERGE*.data $PBS_O_WORKDIR
rm $TMPDIR/*
```

Submit some jobs to the cluster

Pitfalls

- Use node local disk (`$TMPDIR`)
- Avoid gazillions of files
- Condense results to few files
- Remove temporary files

- Use binary `.mat` files
- A lot of small files?
  - Pack in archives `tar`
  - Ramdisk, `/dev/shm/`?
- Want to run several different things?
  - Check out `run-p-shells.sh` in previous Bash seminar.

submit_job.sh

```
#!/usr/bin/env bash
#PBS -A C3SE-STAFF
#PBS -q beda
#PBS -l walltime=01:00:00
#PBS -l nodes=1:ppn=8
#PBS -o stdout
#PBS -e stderr
module load matlab/7.10

cp $PBS_O_WORKDIR/*.m $TMPDIR
cd $TMPDIR

flags='-nodisplay -singleCompThread'
cmd='pi_calc_tofile(1e6); quit'

for i in {1..8}
do
    matlab ${flags} -r "${cmd}" &
    sleep 0.1
done

wait # Wait for all processes

matlab ${flags} -r 'merge_files'
rm $TMPDIR/PI_CALC*.data

cp $TMPDIR/PI_MERGE*.data $PBS_O_WORKDIR
rm $TMPDIR/*
```

Intermezzo

Questions?

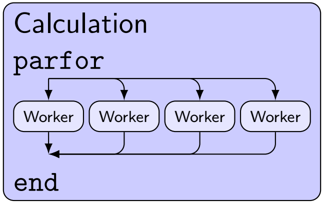
Code Parallelization?

How to use more than one core in a single Matlab script

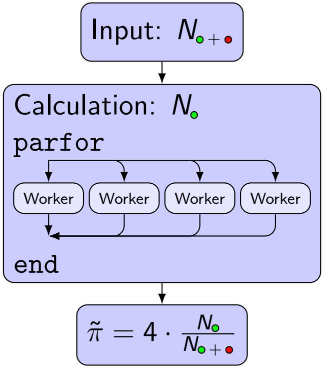
Matlab Parallel Toolbox

- Parallelization over CPU cores
- Parallel for-loops: `parfor`
- Distributed arrays & matrices
- Parallelized toolboxes (Optimization Toolbox, ...)
- etc.

Example: `parfor`



Parallelized π calculation



Input: $N_{\bullet+\bullet}$
Calculation: $N_{\bullet}$
parfor
Worker
Worker
Worker
Worker
end
$\tilde{\pi} = 4 \cdot \frac{N_{\bullet}}{N_{\bullet+\bullet}}$

- Parfor replaces for-loop
- Blocking, classical Matlab "optimization"

  (Bad habit, consider compiled code... without it, 4860 seconds runtime)

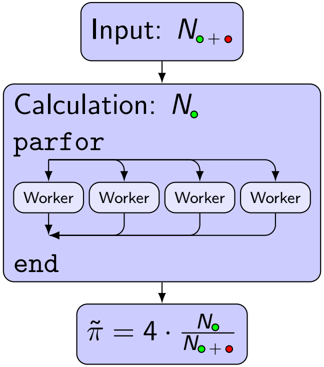
pi_calc_parfor.m

```
function [N_hits, seed] = pi_calc_parfor(N, N_block)

seed = sum(100*clock);
RandStream.setDefaultStream(...
    RandStream('mt19937ar','seed',seed));

N_hits = 0;
parfor iter = 1:N
    coord = rand(N_block, 2);
    r2 = sum(coord .* coord, 2);
    N_hits = N_hits + sum(r2 < 1.0);
end

pi_approx = 4.0*N_hits/(N * N_block)

end
```

Matlab Output

```
>> tic; pi_calc_parfor(1e3, 1e6); toc
pi_approx = 3.1416
Elapsed time is 38.244556 seconds.
>> matlabpool open 'local' 8
Starting matlabpool using the 'local' configuration
... connected to 8 labs.
>> tic; pi_calc_parfor(1e3, 1e6); toc
pi_approx = 3.1416
Elapsed time is 8.615250 seconds.
>> matlabpool close
Sending a stop signal to all the labs ... stopped.
```

Parallelized π calculation



- Parfor replaces for-loop
- Blocking, classical Matlab "optimization"

  (Bad habit, consider compiled code... without it, 4860 seconds runtime)

submit_job_parfor.sh

```
#!/usr/bin/env bash
#PBS -A C3SE-STAFF
#PBS -q dyn
#PBS -l walltime=01:00:00
#PBS -l nodes=1:ppn=8
#PBS -o stdout
#PBS -e stderr

module load matlab/7.10
export DISABLE_MDCS=''

cp $PBS_O_WORKDIR/*.m $TMPDIR
cd $TMPDIR

flags='-nodisplay -singleCompThread'

cat > script.m <<EOF
matlabpool open 'local' 8;
N = 1e3; N_blocks = 1e6;
[N_hits, ~] = pi_calc_parfor(N, N_blocks);
matlabpool close;
data = struct('N', N*N_blocks, 'N_hits', N_hits);
store_data(data, 'PI_PARFOR');
quit
EOF

matlab ${flags} -r script

cp $TMPDIR/PI_PARFOR*.data $PBS_O_WORKDIR
rm $TMPDIR/*
```

Matlab Distributed Computing Server (MDCS)

- Cluster extension of the Parallel Toolbox
- Parallelization and distributed arrays
  over more than one compute-node
- Using the Message Passing Interface (MPI) in the background

- Under consideration by Chalmers IT-control department
- Please let us know if your research group is interested,
  `support@c3se.chalmers.se`

- Requires submission of jobs from inside of Matlab
- Possible seminars on MDCS from Mathworks in the future

Summary & Outlook

Running Matlab on HPC clusters

- How to do it: local disk, multi cores
- Scaling up number of calculations
- Possibly speeding up heavy calculations
- Do not put 3 months on optimization... consider alternatives (compiled code)

Alternatives: Python combined with Fortran/C/C++

- Open Source
- Linear algebra (Numpy)
- Numerical routines (Scipy)
- Multi Threading (multiprocess)
- Message Passing Interface (MPI) (mpi4py)
- Interfacing with compiled code (ctypes, f2py)
- Symbolic math (sympy)
- Sparse parallel linear algebra (petsc4py, slepc4py)
- Visualisation & plotting (matplotlib, mayavi)
- etc.

Finale

Thank you!

谢谢