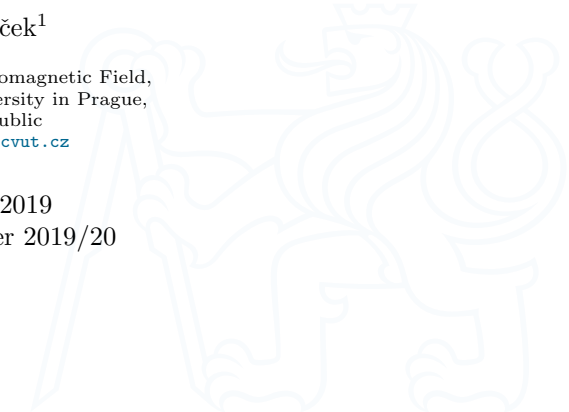
RCI Cluster & MATLAB

Jonáš Tuček[1]

[1]Department of Electromagnetic Field,
Czech Technical University in Prague,
Czech Republic
tucekjon@fel.cvut.cz

October, 2019
Winter semester 2019/20



Outline

RCI Cluster configuration

Compute nodes

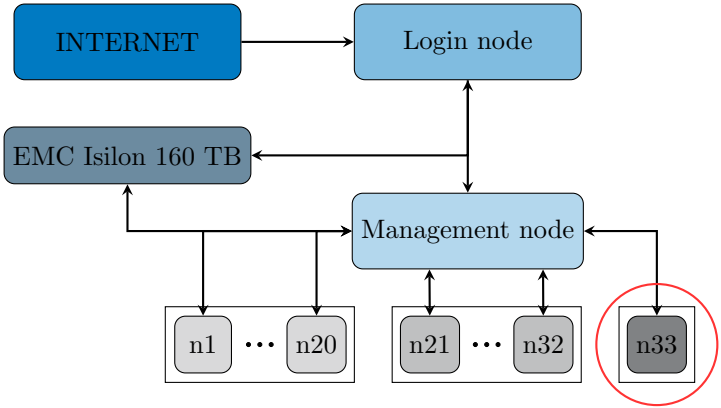
- **n01-20** CPU nodes: 24x2 cores[1] 3.2GHz, 384GB RAM,
- **n21-32** GPU nodes: 36x2 cores 2.7GHz, 384GB RAM, 4 x Tesla V100, 5120 CUDA cores
- n33 Multi CPU node: 192x2 cores 2.1GHz, 1536GB RAM

Network & Storage

- 2TB SSD NVMe as fast local sratch storage in all compute nodes
- 160TB Dell EMC Isilon as Netwok Storage, home directory accessible from all nodes
- 100GbE InfiniBand EDR for MPI communication
- 10GbE Ethernet for NFS

[1] physical cores × threads

RCI Cluster configuration



Interconnection Diagram of RCI Cluster

How to connect to RCI cluster

Registration

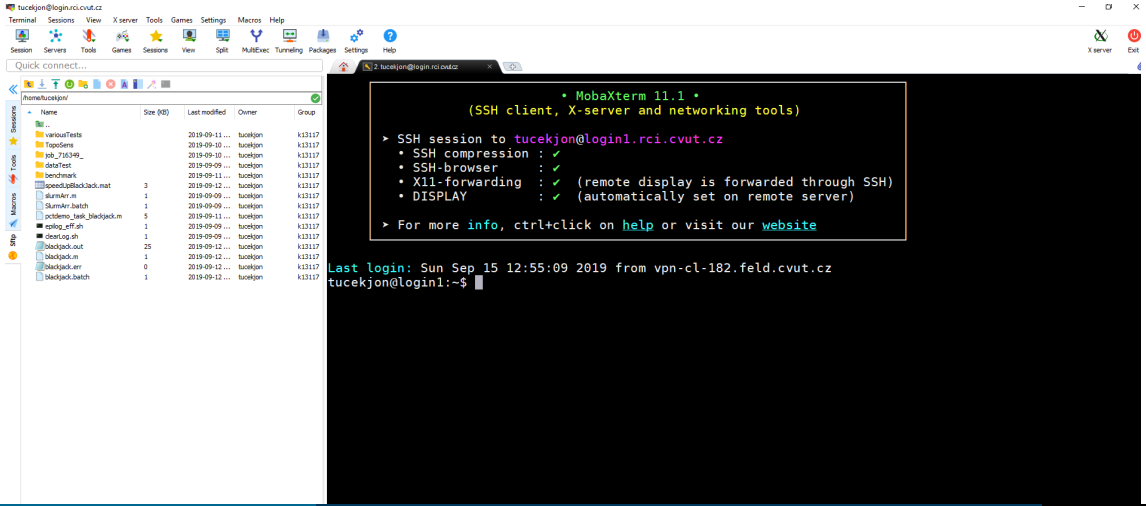
1. Fill the registration form to obtain one year access.
2. Wait for an email from cluster admin.
3. Set password to the RCI account via password change form
   - **RCI username = CTU username**

Access

Use MobaXterm as SSH client in Windows.

1. Launch MobaXterm
2. Sessions tab → New Session → SSH
   - **Remote host**: username@login.rci.cvut.cz
3. Use your RCI password.

MobaXterm GUI



Priority

Priority

- Complicated formula, depends on many factors, *e.g.*, partition, time in queue, ... more info
- Generally, non-RCI researchers has lower priority.
- Partitions are groups of nodes with the same settings, see table 1.
- Job with partition of higher priority can overtake other jobs with partition of lower priority.

Partition	Nodes	Time limit	Default	Priority
cpu	n01-20, n33	24 hrs	yes	10
gpu	n21-32	24 hrs	no	15
smp	n33	24 hrs	no	20
longjobs	n01-33	1024 hrs	no	1
deadline	n01-33	24 hrs	no	30

RCI cluster partitions.

Control commands

- RCI cluster uses SLURM as job scheduler.
- Official guide to SLURM.

Fundamental control commands

- **rci_load**: show RCI cluster nodes and partition statistics.
- **showpartitions**: reports partition statistics.
- Following commands have a wide variety of filtering, sorting and formatting options. See PDF.
- **squeue** - reports the state of jobs or job steps.
- **sinfo** - reports the state of partitions and nodes managed by Slurm.
- **sacct** - reports job or job step accounting information about active or completed jobs.
- **seff** - reports the CPU and memory efficiency (real usage compared to the requested resources).

Starting interactive job

- Basic command for job submission into queue is **srun**.

Example

```
tucekjon@login1:~$ srun --partition=gpu --ntasks-per-node=48 --pty bash -i
tucekjon@n02:~$ ml MATLAB/9.4
tucekjon@n02:~$ matlab
⋮
>> A = magic(3);
>> exit
```

- The new bash is started on the first node with partition **gpu** and demands 48 workers.
- Afterwards, MATLAB is loaded and started, *i.e.*, command window is available.
- Command **salloc** is another option to start interactive shell.

Batch job

- Most suitable way to submit long running jobs.
- Simply prepare batch script with multiple commands.
- Submit job using **sbatch**:
- tucekjon@login1:~$ sbatch test.batch

Simple batch script

```
#!/bin/sh
#SBATCH --ntasks=1
#SBATCH --partition=cpu # choose partition
#SBATCH --cpus-per-task=384 # requsted number of CPUs for this task
#SBATCH --error=test.err # standard error file
#SBATCH --output=test.out # standard output file

ulimit -Su 40000 # increase the limit of internal processes
ml MATLAB/9.4 # load Matlab

srun matlab -r "setPool;test;delPool;exit" # -r = run, test = script to run
```

SBATCH parameters

Useful SBATCH parameters

- Specified in the batch script (or on command line).
- Some parameters are listed below (See sbatch.)

```
#SBATCH --job-name="test" # job name
#SBATCH --nodes=1 # 1 node
#SBATCH --time=1:00:00 # time limit: 1 hour
#SBATCH --mem=230 # request 230 GB
#SBATCH --mail-user=email_address # send info about job
#SBATCH --mail-type=ALL # what to send
```

Environment variables

Environment variable

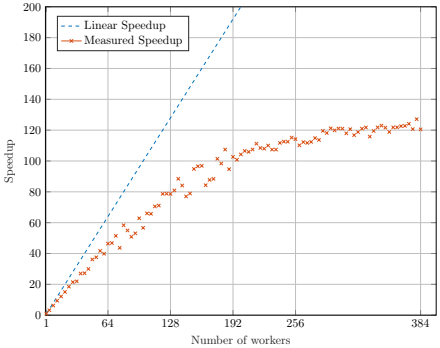
- There are number of useful environment variables provided by the SLURM.
- Few env. variables are listed below with some typical values (See sbatch.)

SLURM_JOB_NAME = test
SLURM_JOB_ID = 773919
SLURM_CPUS_ON_NODE = 384
SLURM_SUBMIT_DIR = /home/username

Usage: Setting parallel pool in MATLAB

```
pc = parcluster('local'); % Information about local cluster profile
CPUs = str2num(getenv('SLURM_CPUS_ON_NODE'))/2; % Get number of workers requested
pc.NumWorkers = CPUs; % Set NumWorkers (problems with logical cores)
ppool = parpool(pc, CPUs); % Inicialize pool
```

Speedup



Measured speedup on node 33 of RCI cluster.

- Simple benchmark of PARFOR is performed on node 33 of RCI cluster, result is plotted in figure 2.
- Speedup is saturated due to the presence of logical cores.
- Allocate whole n33 to ensure that you get all physical cores or ..
- Provide corresponding #SBATCH parameters:
  #SBATCH - -extra-node-info=8:24:2
- Sockets:Cores:Threads
- Discover node decomposition:
  tucekjon@login1:~$ sinfo -e -o "%N %z"

Working with memory

- Very fast local scratch folder `/data/temporary/` is available on all nodes.
- Scheduler requires high data transfer and saving (Home is default folder).
- More efficient is creating local folder to manage job.
- Note: Linux uses "/" in path definition (Win "\").
- Use fullfile in MATLAB.

1. Use `mkdir /data/temporary/job_${SLURM_JOB_ID}` in batch script to create folder.
2. Call
   ```
   pc.JobStorageLocation =
   strcat('/data/temporary/job_', getenv('SLURM_JOB_ID'));
   ```
   within Matlab to set up folder to manage job.
3. Or pass reference on path to matlab, *e.g.*, `setPool('${PATH}')`
4. Folders can be copied, *e.g.*, `cp -R home/tucekjon/AToM/ /data/temporary/`
   - Do not forget to add path in Matlab.
5. Copy results back to home folder and clear scratch folder,
   `rm -rf /data/temporary/job_${SLURM_JOB_ID}`

Shape optimization

- Shape optimization based on Topology sensitivity (arXiv) was performed on CPU Threadripper 1950X (3.4 GHz, 16×2 cores), 128 GB RAM.
- Performance is shown in table below

plate	4×8	6×12	8×16
d-o-f, N	180	414	744
runs, I	$5 \cdot 10^4$	$5 \cdot 10^4$	$1 \cdot 10^3$
Comp. time, T [s]	$2.4 \cdot 10^3$	$5.8 \cdot 10^4$	$1.2 \cdot 10^4$
$Q_{\min}/Q_{\text{lb}}^{\text{TM}}$	1.18	1.12	1.11

Shape optimization performance on Threadripper.

Shape optimization performance on RCI

- Shape optimization algorithm is peformed on node 33 of the RCI cluster, *i.e.*, 192 CPUs/ 384 Threads (2.1 GHz), 1536 GB RAM

plate	4×8	6×12	8×16	10×20	12×24
d-o-f, N	180	414	744	1170	1692
runs, I	$5 \cdot 10^4$	$5 \cdot 10^4$	$1 \cdot 10^3$	$5 \cdot 10^3$	$5 \cdot 10^3$
Comp. time, T [s]	236	$2.8 \cdot 10^3$	418	$1 \cdot 10^4$	$3.9 \cdot 10^4$
$Q_{\min}$	51.40	47.81	47.52	45.67	45.14
$Q_{\mathrm{lb}}^{\mathrm{TM,S}}$	45.73	45.22	44.98	44.81	44.70
$Q_{\min}/Q_{\mathrm{lb}}^{\mathrm{TM,S}}$	1.12	1.06	1.06	1.02	1.01

Shape optimization performance on node 33 of RCI cluster

Shape optimization performance

- Monte Carlo analysis with `ff_minQTM.m` as fitness function.
- Table below shows realized speed up on n33 compared to CPU Threadripper.

plate	4×8	6×12	8×16
speedup [-]	≈ 10	≈ 21	≈ 29

Computation speed up, n33 compared to Threadripper.

- Speed up scales with the size of the problem.
- Perform computationally expensive jobs on n33.

SLURM job Arrays

- Offers a mechanism to submitting and managing collection of similar job.
- Job arrays are supported only for batch jobs.
- Array index values are specified using sbatch option parameter, *i.e.*, `--array=` or `-a`.
- Array index is range between **0** and **1000**.
- Single job from job arrays will have the environment variable SLURM_ARRAY_TASK_ID set to its array index value.
- This variable might be used to branch matlab script with `switch-case` statement or as initial condition.
- Check example batch script.

Example

```
#SBATCH --array=1-5 # Job array with ID = 1,2,3,4,5
#SBATCH --array=1-5:2 # Job array with ID = 1,3,5 (step 2)
```

Parallel MATLAB computing

- Full documentation to Parallel Computing Toolbox is available.
- **Parfor**(parallel for loop) is fundamental command PCT statement.
  - Parfor can provide significantly better performance than for-loop.
  - Decide, when to use parfor.

Parfor demands:

- Since parfor is more sensitive than for-loop, it requires precise tuning.
- Parfor needs integer increasing loop variable.
- Nested parallel loops are not allowed. Help.
- Each iteration of parfor must be independent. Help.
- Pay attention to variable classification. Help.

Improve performance of parfor

1. Where to Create Arrays
   - Creating a large array before parfor might lead to slow execution.
   - To improve performance, tell each worker to create its own array or portion of it in parallel.
   - As alternative, consider parallel.pool.Constant to establish variables. These variables are copied to each worker and remain on the workers after the parfor loop ends.
2. Profiling parfor-loops
   - Profile a parfor structure using tic-toc measurement.
   - Measure how much data is transferred to and from each worker with ticBytes-tocBytes.
3. Slicing Arrays
   - Each variable, which is initialized before parfor loop, has to be transferred to each worker evaluating the loop iterations.
   - Only those variables used inside the loop are passed from the client!
   - Try avoid using a reference to the variable defined before parfor.
4. Script, which is running on RCI cluster, might behave differently, as workers can create their arrays in parallel, *i.e.*, saving transfer time. Code that is optimized for local workers might not be optimized for cluster workers.
5. Example m file with comments is available in folder `Matlab`.

Summary

Recommendation

- Computation on node 33 of the RCI cluster is suitable for high number of unknowns.
- SLURM is complex job scheduler and it is convenient to tune the job submission to the particular computation.
- Big documentations to SLURM and parallel MATLAB are available.
- Optimize your parallel code on your local computer and then measure its performance on RCI cluster and perform finer tuning.
- Example batch scripts are available for use. See folder `template`.

Further work

- Find more effective job submission properties to save computation time (if there is).
- Try other computationally expensive algorithms.

Comments?

Jonáš Tuček
tucekjon@fel.cvut.cz

October, 2019
Winter semester 2019/20