Practical Introduction to UL HPC

Xavier Besseron

UL HPC School
June 25, 2015
Outline

1. What is HPC?
2. Best Practices for UL HPC
3. HPC: Getting Fast and Efficient
Computer Simulation is everywhere

- Computational Fluid Dynamics (OpenFOAM)
- Finite Element Analysis (Abaqus)
- Climate / Weather / Ocean Simulation (WRF)
- Molecular Dynamics (Gromacs, Amber)
- Quantum Chemistry (Quantum Espresso)
- Visualization (Paraview)
- Data processing (R, Matlab)
- ...

What is HPC? Best Practices Getting Fast & Efficient
What is High Performance Computing?

High Performance Computing (HPC)
- Use of parallel and distributed computers with fast interconnects
- To execute an application quickly and efficiently

Why parallel computers?
- Performance of single CPU core is getting limited (power, physics)
- Multiple cores are used to increase the computing capacity

HPC is challenging
- Active research domain
- Provides tools for many other researchers
Can you benefit from HPC?

Your application is limited by the performance of your computer

<table>
<thead>
<tr>
<th></th>
<th>Your workstation</th>
<th>UL HPC platform$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>160 GFlops</td>
<td>49.9 TFlops</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>16 GB</td>
<td>20.6 TB</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>2 PB</td>
<td>3.4 PB</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>Ethernet 1 Gb/s</td>
<td>Infiniband 40 Gb/s</td>
</tr>
<tr>
<td><strong>Accelerators</strong></td>
<td>1 GPU</td>
<td>29 GPUs</td>
</tr>
</tbody>
</table>

⇒ HPC provides the **tools** for your application to run faster

$^1$shared among many users
Other benefits of using the UL HPC platform over using your personal computer

Long uptime and stable
- Cluster nodes are always running

Remote access
- Start a job from work, check results from home

No administration
- The HPC team maintain the hardware/software

Large software collection
- Scientific and general-purpose applications pre-installed

Backup
- Automatic backup of your Home directory\(^2\)

\(^2\)Always keep a backup of your critical data!
Outline

1. What is HPC?
2. Best Practices for UL HPC
3. HPC: Getting Fast and Efficient
Know the basics!

Get an account

- [https://hpc.uni.lu/users/get_an_account.html](https://hpc.uni.lu/users/get_an_account.html)
- Please read carefully the Acceptable Use Policy

Access the clusters, access and reserve nodes

- Use SSH and public key authentication
  [https://hpc.uni.lu/users/docs/access.html](https://hpc.uni.lu/users/docs/access.html)
- Learn how to use the OAR resource manager
  [https://hpc.uni.lu/users/docs/oar.html](https://hpc.uni.lu/users/docs/oar.html)

Transfer files between your computer and the clusters

- Learn how to use tools like `scp`, `rsync`, etc.
  [https://hpc.uni.lu/users/docs/filetransfer.html](https://hpc.uni.lu/users/docs/filetransfer.html)

Use pre-installed software

- Search and use software with the `module` command
  [https://hpc.uni.lu/users/docs/modules.html](https://hpc.uni.lu/users/docs/modules.html)

Join session PS 1B to learn the basics!
Check Live Status of the platform

- **Current node status with Monika**
  [https://hpc.uni.lu/status/monika.html](https://hpc.uni.lu/status/monika.html)

- **Platform occupation – Gantt chart with DrawGantt**
  [https://hpc.uni.lu/status/drawgantt.html](https://hpc.uni.lu/status/drawgantt.html)

- **Resource usage with Ganglia**
  [https://hpc.uni.lu/status/ganglia.html](https://hpc.uni.lu/status/ganglia.html)
Getting help

- Check the UL HPC quick reference
- RTFM! Online Documentation available at
  https://hpc.uni.lu/users/docs/
- Google is your friend!
- Ask other users on the mailing list hpc-users@uni.lu
- Ask the HPC sys-admins hpc-sysadmins@uni.lu
Workflow for Experiment Campaigns

1. Before the campaign
   - Send data to the clusters
   - Check/Install required software

2. Preparation
   - Test and debug
   - Prepare launcher script

3. Execution
   - Run the campaign
   - Monitor the execution

4. After a campaign
   - Retrieve output data
   - Archive and cleanup your data
Workflow for Experiment Campaigns

1. Before the campaign → *cf the Basics*
   - Send data to the clusters
   - Check/Install required software

2. Preparation
   - Test and debug
   - Prepare launcher script

3. Execution
   - Run the campaign
   - Monitor the execution

4. After a campaign → *cf the Basics*
   - Retrieve output data
   - Archive and cleanup your data
Experiment campaign: Preparation

Goals

- Make sure everything will run OK
- Prepare submission script / launcher

Interactive approach

- Use option \(-I\) (Interactive) of \texttt{oarsub} command
- Allows to try commands one by one
- Work on a small case with a small number of cores
- Debug and check the results

Why prepare a submission script?

- Contains all commands and parameters
  - Easy re-execution
- No need to stay in front your computer
Experiment campaign: Execution

Submit the jobs

- Use the submission script / launcher
- Submit to OAR with option \(-S\) (Script) of oarsub
- Actual experiment execution with possibly many nodes
- Non interactive execution, it might not start immediately

Monitor the execution

- Status of your job: \texttt{oarstat \(-j \ <OAR\_JOBID> \ -f\)}
- Output/Logfile of your application
- Resource usage (CPU, memory, etc.) on the node: \texttt{htop}
  with Ganglia: \url{https://hpc.uni.lu/status/ganglia.html}
Outline

1. What is HPC?
2. Best Practices for UL HPC
3. HPC: Getting Fast and Efficient
Getting faster: Identify performance bottlenecks

Note for code developers: The first bottleneck is your algorithm!

Know the hardware

- Computer nodes are connected using a fast interconnect
- Different types of resources:
  Processors, GPU, Memory, Storage, Network
Getting faster: Identify performance bottlenecks

Note for code developers: The first bottleneck is your algorithm!

Know the hardware

- Computer nodes are connected using a fast interconnect
- Different types of resources: Processors, GPU, Memory, Storage, Network
Getting faster: Identify performance bottlenecks

Note for code developers: The first bottleneck is your algorithm!

Know the hardware

- Computer nodes are connected using a fast interconnect
- Different types of resources: Processors, GPU, Memory, Storage, Network
Getting faster: Identify performance bottlenecks

Note for code developers: The first bottleneck is your algorithm!

Know the hardware

- Computer nodes are connected using a fast interconnect
- Different types of resources: Processors, GPU, Memory, Storage, Network
Getting faster: Identify performance bottlenecks

Note for code developers: The first bottleneck is your algorithm!

Know the hardware

- Computer nodes are connected using a fast interconnect
- Different types of resources: Processors, GPU, Memory, Storage, Network
Getting faster: Identify performance bottlenecks

Note for code developers: The first bottleneck is your algorithm!

Know the hardware

- Computer nodes are connected using a fast interconnect
- Different types of resources: Processors, GPU, Memory, Storage, Network
What is HPC? Best Practices Getting Fast & Efficient

Processor bottleneck

Application is limited by the speed of the processor
Processor bottleneck

Application is limited by the speed of the processor

→ Optimize your code
Processor bottleneck

Application is limited by the speed of the processor

→ Optimize your code
→ Parallel execution on a single node (pthread, OpenMP, Intel TBB)
Processor bottleneck

Application is limited by the speed of the processor
→ Optimize your code
→ Parallel execution on a single node (pthread, OpenMP, Intel TBB)
→ Use GPU accelerator (CUDA)
Processor bottleneck

Application is limited by the speed of the processor
→ Optimize your code
→ Parallel execution on a single node (pthread, OpenMP, Intel TBB)
→ Use GPU accelerator (CUDA)
→ Parallel execution on multiple nodes (MPI)
What is HPC? Best Practices Getting Fast & Efficient

Processor bottleneck

Application is limited by the speed of the processor

→ Optimize your code
→ Parallel execution on a single node (pthread, OpenMP, Intel TBB)
→ Use GPU accelerator (CUDA)
→ Parallel execution on multiple nodes (MPI)
→ Parallel execution on multiple nodes with GPUs (MPI+CUDA)
Memory bottleneck

Application is limited by the size of the memory

- There is one memory bank per CPU
Memory bottleneck

Application is limited by the size of the memory

- There is one memory bank per CPU
  → Allocate all CPUs on a single node
Memory bottleneck

Application is limited by the size of the memory

- There is one memory bank per CPU
  → Allocate all CPUs on a single node
  → Use a node with a bigger memory (4TB-memory node at UL)
Application is limited by the size of the memory

- There is one memory bank per CPU
- Allocate all CPUs on a single node
- Use a node with a bigger memory (4TB-memory node at UL)
- Distributed execution on multiple nodes (MPI)
Application is limited by the available storage space
What is HPC?
Best Practices
Getting Fast & Efficient

Storage space bottleneck

Application is limited by the available storage space
→ Use $\text{\texttt{WORK}}$ (3 TB) or $\text{\texttt{SCRATCH}}$ (10 TB)
(no backup!)
Storage speed bottleneck

Application is limited by the speed of the storage
Storage speed bottleneck

Application is limited by the speed of the storage

→ Use local storage instead of network storage
  (copy data back to network storage after execution)
Application is limited by the speed of the storage

→ Use local storage instead of network storage
   (copy data back to network storage after execution)

→ Use local memory, *eg* `/dev/shm` *(space is limited!)*
Application is limited by the speed of the network (too many communications)
Application is limited by the speed of the network (too many communications)

→ Use Infiniband network instead of Ethernet
Application is limited by the speed of the network (too many communications)

→ Use Infiniband network instead of Ethernet
→ Reduce the number of nodes
### Quick Tips for classic use-cases

<table>
<thead>
<tr>
<th>Sequential job</th>
<th>PS 2A, PS 5C</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ Parallelization: OpenMP, MPI</td>
<td></td>
</tr>
<tr>
<td>→ Use accelerators: CUDA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Long job</th>
<th>PS 6C</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ Checkpoint/Restart: BLCR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Large number of jobs</th>
<th>PS 6C</th>
</tr>
</thead>
<tbody>
<tr>
<td>eg parametric studies</td>
<td></td>
</tr>
<tr>
<td>→ Parallel launcher</td>
<td></td>
</tr>
<tr>
<td>→ Best effort queue</td>
<td></td>
</tr>
<tr>
<td>→ OAR Job containers</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visualization / Rendering</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>→ GPU nodes with graphic session using XCS portal</td>
<td></td>
</tr>
<tr>
<td>Keynote 4, PS 6B</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MATLAB</th>
<th>PS 2B, PS 3B</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ Parallelization: Parallel Computing Toolbox</td>
<td></td>
</tr>
<tr>
<td>→ Checkpoint/Restart: <code>save()</code>/<code>load()</code> functions</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>PS 3C</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ Use optimized data structure / package: <code>data.table</code>/<code>plyr</code></td>
<td></td>
</tr>
<tr>
<td>→ Parallelization: packages <code>parallel</code>, <code>Rmpi</code>, <code>snow</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Python</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>→ Library for scientific computing: <code>NumPy</code>, <code>Scipy</code>, <code>matplotlib</code></td>
<td></td>
</tr>
<tr>
<td>→ Parallelization: <code>multiprocessing</code> library</td>
<td></td>
</tr>
<tr>
<td>→ Use latest version of Python</td>
<td></td>
</tr>
</tbody>
</table>

...