UL HPC School 2017
PS2: HPC workflow with sequential jobs
(test cases on GROMACS, Java and Python)

UL High Performance Computing (HPC) Team
H. Cartiaux

University of Luxembourg (UL), Luxembourg
http://hpc.uni.lu
Latest versions available on Github:

UL HPC tutorials:  
https://github.com/ULHPC/tutorials

UL HPC School:  
http://hpc.uni.lu/hpc-school/

PS2 tutorial sources:  
http://ulhpc-tutorials.readthedocs.io/en/latest/basic/sequential_jobs/
Introduction

Summary

1. Introduction

2. Pre-requisites

3. Exercise 1: Parametric execution of Gromacs

4. Exercise 2: Watermarking images in Python

5. Exercise 3: Advanced use case, using a Java program: "JCell"

6. Conclusion
Main Objectives of this Session

- Run sequential, parametric programs on the clusters
- Learn how-to use our set of launcher scripts
- Submit jobs
- use the cluster monitoring tools
  - Ganglia
  - Monika & Drawgantt

Read the full subject of this PS here

http://git.io/5cYmPw
HPC
uni.lu
UNIVERSITÉ DU LUXEMBOURG

Pre-requisites

Summary

1. Introduction

2. Pre-requisites

3. Exercise 1: Parametric execution of Gromacs

4. Exercise 2: Watermarking images in Python

5. Exercise 3: Advanced use case, using a Java program: "JCell"

6. Conclusion
Pre-requisites

Getting started

1. Connect to the cluster(s)

(laptop)$> ssh {iris,gaia,chaos}-cluster

2. Send files

(laptop)$> rsync -avz local_directory {iris,gaia,chaos}-cluster:

3. Retrieve files

(laptop)$> rsync -avz {iris,gaia,chaos}-cluster:path/to/files local_directory

4. Submit jobs

<table>
<thead>
<tr>
<th>OAR on Chaos/Gaia</th>
<th>Slurm on Iris</th>
</tr>
</thead>
<tbody>
<tr>
<td>oarsub -l</td>
<td>srun -p interactive –qos qos-interactive –pty bash</td>
</tr>
<tr>
<td>oarsub ./program</td>
<td>sbatch program</td>
</tr>
</tbody>
</table>
This tutorial is available on github!

https://git.io/vHyh3
Summary

1. Introduction
2. Pre-requisites
3. Exercise 1: Parametric execution of Gromacs
4. Exercise 2: Watermarking images in Python
5. Exercise 3: Advanced use case, using a Java program: "JCell"
6. Conclusion
Exercise 1: Parametric execution of Gromacs

Gromacs

GROMACS: GROningen MAchine for Chemical Simulations
versatile package for molecular dynamics, primarily designed for biochemical molecules

- very large codebase: 1.836.917 SLOC
- many applications in the package, several parallelization modes
- **mdrun**: computational chemistry engine, performing:
  - molecular dynamics simulations
  - Brownian Dynamics, Langevin Dynamics
  - Conjugate Gradient
  - L-BFGS
  - Steepest Descents energy minimization
  - Normal Mode Analysis
- **mdrun** - parallelized using MPI, OpenMP, pthreads and with support for GPU acceleration
Exercise 1: Parametric execution of Gromacs

Comparison

2 approaches

- Sequential (loop)
- Parallelized (with GNU parallel)
Exercise 1: Parametric execution of Gromacs

Comparison - Ganglia

[Graph showing CPU usage over time]
Exercise 2: Watermarking images in Python

Summary

1. Introduction

2. Pre-requisites

3. Exercise 1: Parametric execution of Gromacs

4. **Exercise 2: Watermarking images in Python**

5. Exercise 3: Advanced use case, using a Java program: "JCell"

6. Conclusion
Exercise 2: Watermarking images in Python

Watermark Application

- **Objective:** Apply a watermark to a given set of pictures
  - Simple Python script
  - Generic parallel launcher
  - Distribute the work on several nodes
Exercise 2: Watermarking images in Python

Source image
Exercise 2: Watermarking images in Python

Watermarked image
Summary

1. Introduction
2. Pre-requisites
3. Exercise 1: Parametric execution of Gromacs
4. Exercise 2: Watermarking images in Python
5. Exercise 3: Advanced use case, using a Java program: "JCell"
6. Conclusion
Exercise 3: Advanced use case, using a Java program: "JCell"

**JCell**: a Java framework for working with genetic algorithms

- Ex: Generational algorithm for the Combinatorial ECC problem
- Test the variations of these parameters:
  - Mutation probability and Crossover probability
Summary

1. Introduction
2. Pre-requisites
3. Exercise 1: Parametric execution of Gromacs
4. Exercise 2: Watermarking images in Python
5. Exercise 3: Advanced use case, using a Java program: "JCell"
6. Conclusion
Conclusion

We have covered one of the most common workflow:

- parametric jobs

Our launchers can be improved!

Perspectives

- Array jobs
- Best effort jobs
- Checkpoint/Restart mechanism
Thank you for your attention...

Questions?

The UL High Performance Computing (HPC) Team
University of Luxembourg, Belval Campus:
Maison du Nombre, 4th floor
2, avenue de l’Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu

1 Introduction
2 Pre-requisites
3 Exercise 1: Parametric execution of Gromacs
4 Exercise 2: Watermarking images in Python
5 Exercise 3: Advanced use case, using a Java program: "JCell"
6 Conclusion