Graphical interfaces in MetaCentrum

Aleš Křenek

e-Infra CZ conference, Apr 30, 2024

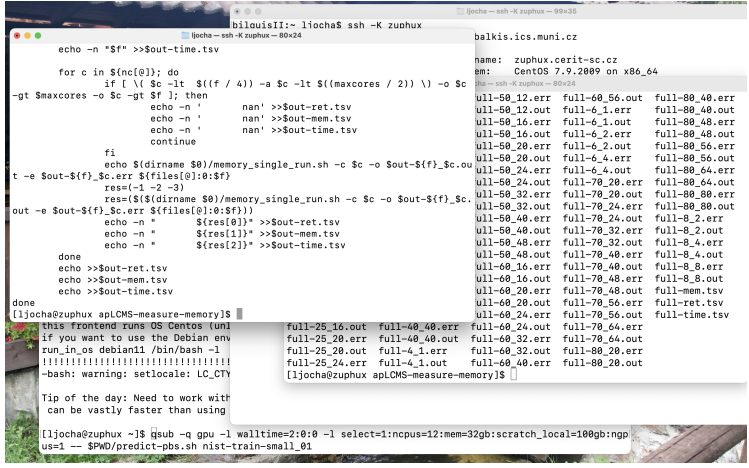
Motivation

Tony Mitchell's view

(Supervolcano, 2005)

Motivation

Reality for many of us



Motivation

Beyond qsub

- Batch jobs still achieve the best resource utilization
  - and the resources are expensive
- The user wants better experience nowadays
- **Graphical interfaces** are a trade-off
- Become more and more FAIR

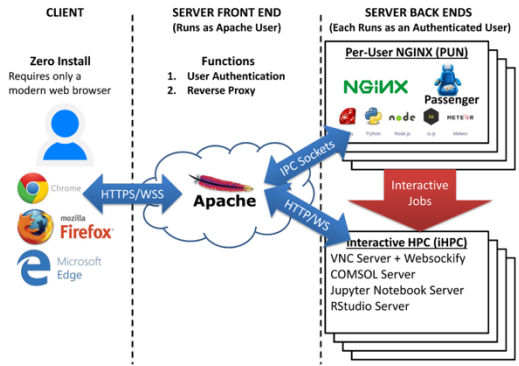
Open OnDemand

User experience

- I need 64 cores, 1 TB RAM, 4 hi-end GPU for my Matlab calculation
- I can’t spend so much money to purchase this computer
- Well, I need it 2 hours per month either
- Go to `https://ondemand.metacentrum.cz/`
  - say what you need for how long

Open OnDemand

Architecture



https://osc.github.io/ood-documentation/latest/architecture.html

Open OnDemand

3D accelerated remote desktop

- VirtualGL, `https://virtualgl.org`
  - intercept OpenGL calls, render on headless GPU, copy image
- TurboVNC, `https://turbovnc.org`
  - VNC server optimized for transmitting rendered 3D images
- noVNC, `https://novnc.com`
  - zero install VNC client (HTML5)

Open OnDemand

Optimized scheduling

- dedicated PBS queue
- handled by dedicated scheduler – fast turnaround
- dedicated compute nodes
- “run-or-move” strategy
  - in average, 50 % jobs are run immediately ($<$ 1 minute)
  - otherwise moved to “normal” queue

Open OnDemand

Available applications

- Ansys/Fluent, Ansys/Workbench
- CLCgenomcsWB
- VMD
- Matlab, RStudio, Jupyter notebook
- generic desktop (“module add anything”)
- plain shell

Rancher

- `https://rancher.cloud.e-infra.cz`
- Similar purpose, different technology (Kubernetes)
- Far more flexible – less intuitive sometimes
- Slightly different set of prepared applications

Galaxy

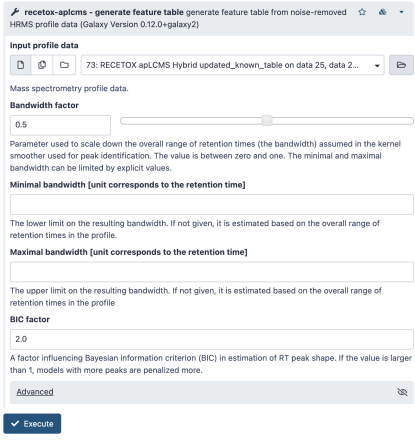
Overview

- `https://usegalaxy.org`, `https://usegalaxy.eu`, `https://usegalaxy.cz`
- “Galaxy is a free, open-source system for analyzing data, authoring workflows, training and education, publishing tools, managing infrastructure, and more.”

(`https://galaxyproject.org`)

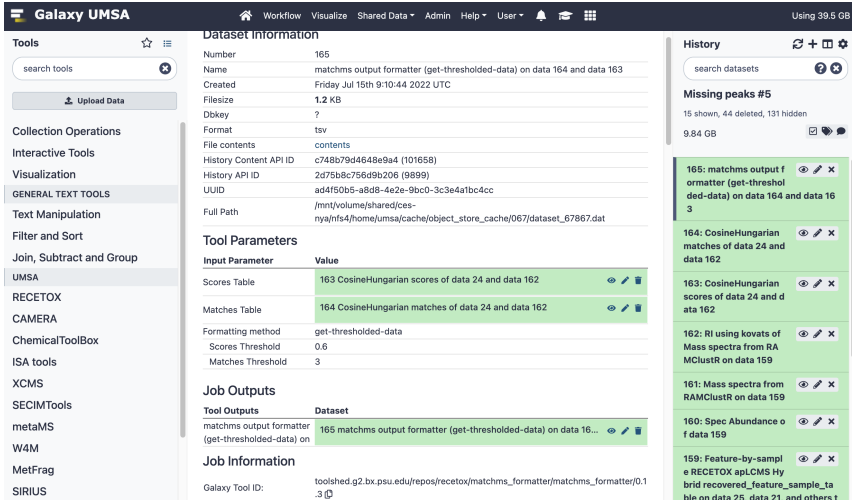
Galaxy

Tool invocation



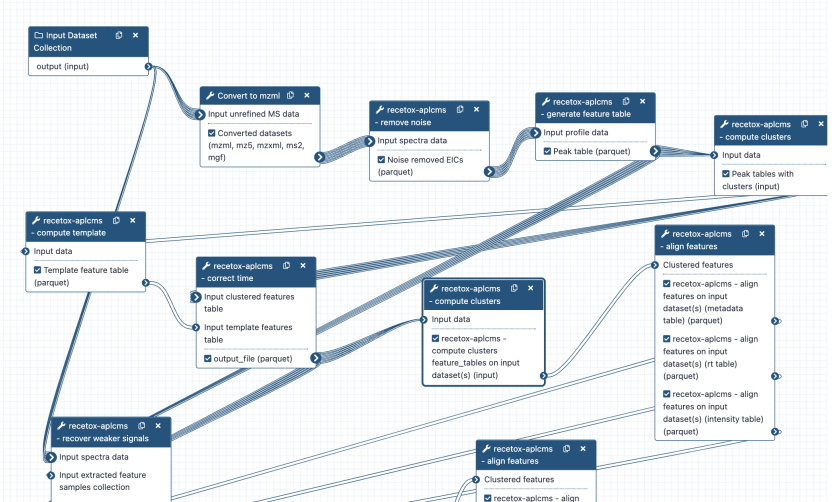
Galaxy

Work with history



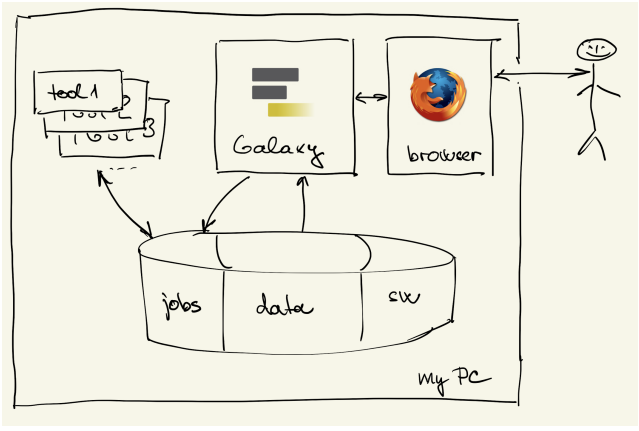
Galaxy

Workflows



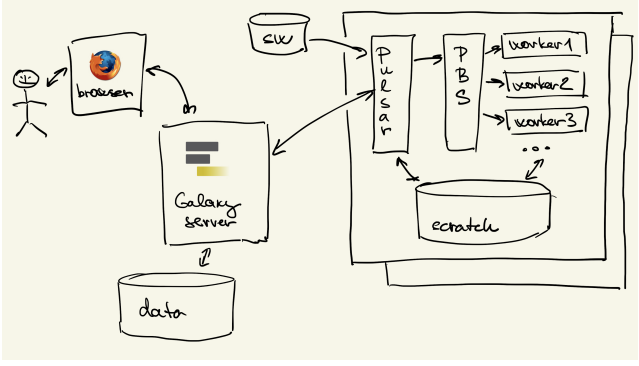
Galaxy

Install your own



Galaxy

Pulsar network



Galaxy

usegalaxy.cz

- Pulsar architecture, submit jobs to Metacentrum PBS
- Mirrored set of tools from `usegalaxy.eu`
- More resources for e-Infra CZ users, including GPU

Jupyter Notebooks

Successful example: Metadynminer (Jan Beránek, UCT)

python_metadynminer.ipynb

Code

Open in... Python 3 (ipykernel)

5. Identify the local minima

You have two options: If you set the keyword `precise=True`, the local minima will be identified by an algorithm which:

1. finds all local minima, even very shallow and probably unimportant minima,
2. each point on the FES will be assigned to the minimum the system would most likely go to, if it only follows the gradient of free energy, and
3. free energy value of minima will be calculated from each point on FES assigned to the respective minima. This results in more precise free energy values, as it accounts for the width of the minimum as well. For this calculation the unit of free energy (`energy_unit="kJ/mol"` or `energy_unit="kcal/mol"`) and the thermodynamical temperature (`temp`) of the simulation must be supplied. This algorithm doesn't use the nbins keyword.

Example:

```
minima = mm.Minima(fes, precise=True, temp=300.0, energy_unit="kJ/mol")
```

If you set `precise = False`, the method will use the original algorithm from the metadynminer package for R. In this algorithm the FES is first divided to number of bins (can be set with option `nbins`, default is 8) and the absolute minima is found for each bin. Then the algorithm checks if this point is really a local minimum by comparing to the surrounding points of FES. This algorithm only accounts for the depth of each minima, which is less precise, but usually sufficient.

In some cases, this algorithm is the prefered one, because on some free energy landscapes the total number of local free energy minima can reach tens of thousands, which makes the calculation using precise algorithm slow and impractical.

Example:

```
minima = mm.Minima(fes, precise=False, nbins=8)
```

[9]:
```
minima = mm.Minima(fes, precise=True, temp=300.0, energy_unit="kJ/mol")
```

```
Calculating gradients for FES with 65536 bins...
Searching for the nearest local minima...
Done.
```

Print the list of the local minima:

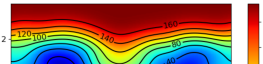
[10]:
```
minima.minima
```

[10]:

	Minimum	free energy	CV1bin	CV2bin	CV1 - torsion_SS	CV2 - torsion_cycle
0	A	0.0	193.0	161.0	1.5953400194010667	0.8517301829032395
1	B	6.963876206871731	61.0	177.0	-1.6444274046134073	1.1935812021613703

You can visualise the FES with the local minima as letters. All keywords are analogous as for the `fes.plot()` function. The color of the letters changes automatically to ensure their good visibility, but if you want to override this behaviour, provide a matplotlib color like this: `color="black"`.

[11]:
```
# plot the free energy surface with minima
minima.plot(contours_spacing=20)
```



Jupyter Notebooks

One size doesn't fit all

1. via `https://ondemand.metacentrum.cz`
   - standard PBS job
   - singularity container (predefined or custom)
   - almost full Metacentrum environment
2. `https://hub.cloud.e-infra.cz`
   - in Kubernetes, with all its flexibility and complexity
   - easy to make custom clones (Metadynminer)
3. `https://notebooks.egi.eu`
   - more standardized for international collaborations
   - support for community installations
   - integration with OneData, B2DROP
   - EGI Check-in authentication

Summary

- Still lagging 20 years behind the scientists in movies
- Metacentrum offers something more than qsub:
  - generic graphical environments: OnDemand and Rancher
  - Galaxy – de facto standard in large international community
  - Jupyter notebooks – mix text and code, record experiments
- Access to the whole infrastructure
- Better user experience, support for open science