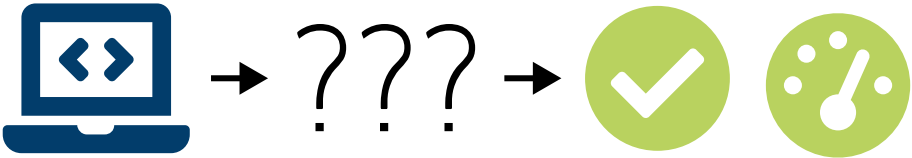


CONTINUOUS INTEGRATION FOR HPC CODES

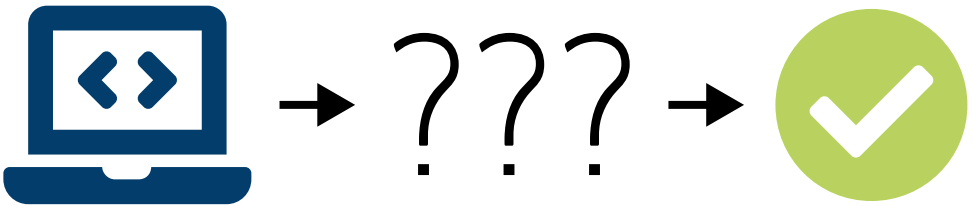
JLESC, April 17, 2024 | Jakob Fritz, Robert Speck | Jülich Supercomputing Centre, Forschungszentrum Jülich

Table of Contents

Continuous Integration for HPC
- Syncing with Gitlab tools
- Syncing Github to Gitlab

Continuous Benchmarking
- How to store results
- How to visualize results

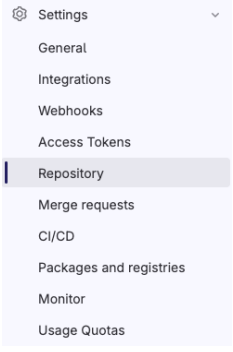
Continuous Integration for HPC

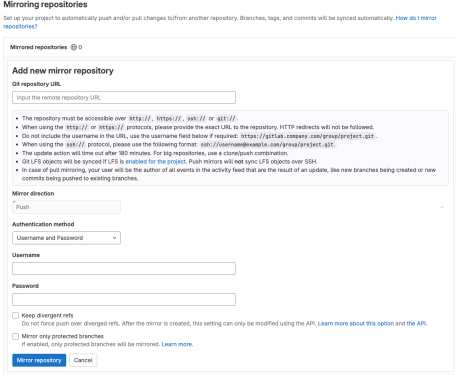


Continuous Integration for HPC

Syncing with Gitlab tools

Use Push-Mirror (can be found in the settings). Pull-Mirror only available in paid versions



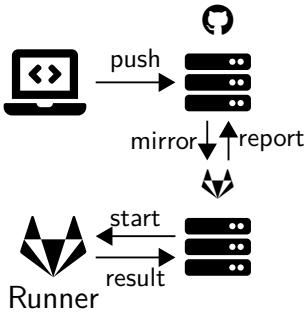


JÜLICH Forschungszentrum

Continuous Integration for HPC

Syncing Github to Gitlab

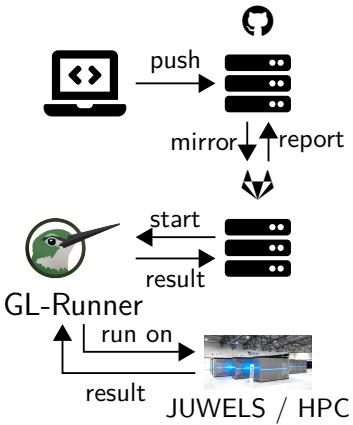
- Already shown last year



Continuous Integration for HPC

Syncing Github to Gitlab

- Already shown last year
- Now used to trigger Jacamar and run on cluster with GPU usage (for testing)

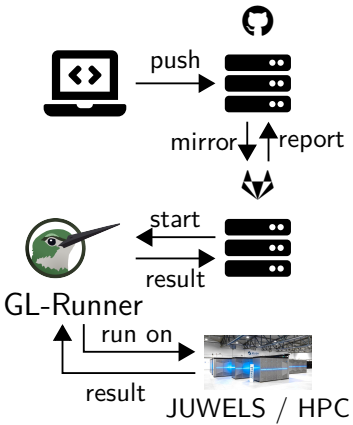


JÜLICH Forschungszentrum

Continuous Integration for HPC

Syncing Github to Gitlab

- Already shown last year
- Now used to trigger Jacamar and run on cluster with GPU usage (for testing)
- Returning results back to repository (Gitlab & Github) is of different difficulty. Success/Failure is easy; numbers are more difficult.
- Numbers are not only for benchmarking, but already for testing, (e.g. test coverage)

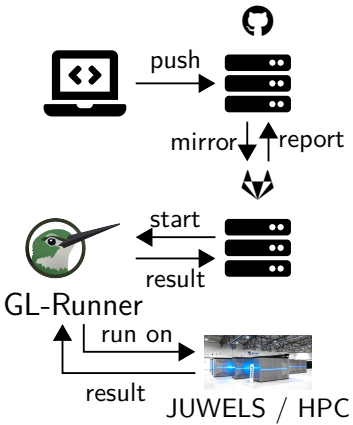


JÜLICH Forschungszentrum

Continuous Integration for HPC

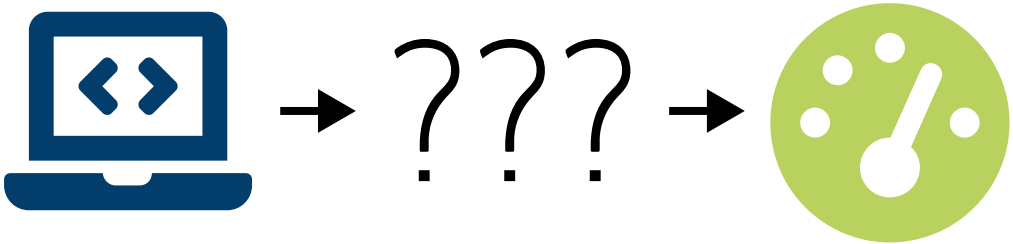
Syncing Github to Gitlab

- Already shown last year
- Now used to trigger Jacamar and run on cluster with GPU usage (for testing)
- Returning results back to repository (Gitlab & Github) is of different difficulty. Success/Failure is easy; numbers are more difficult.
- Numbers are not only for benchmarking, but already for testing, (e.g. test coverage)
- Artifacts created in Jacamar returned to Gitlab. Transfer to Github already present, but transfer between Github workflows of the same commit still needs development



JÜLICH Forschungszentrum

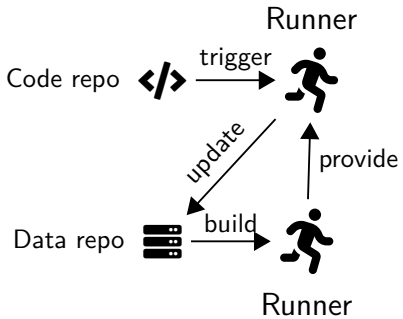
Continuous Benchmarking



Continuous Benchmarking

How to store results

- Storage of results from benchmarks can be in separate repo
- For few metrics and/or meta-data a database can be used (e.g. sqlite) in a repository (therefore, versioned)

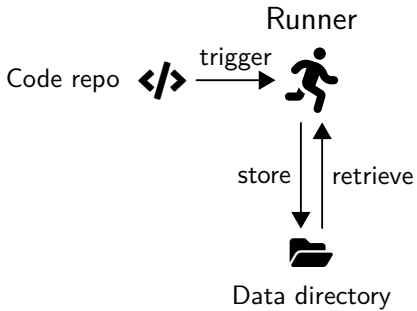


JÜLICH Forschungszentrum

Continuous Benchmarking

How to store results

- Storage of results from benchmarks can be in separate repo
- For few metrics and/or meta-data a database can be used (e.g. sqlite) in a repository (therefore, versioned)
- For larger amounts of data, a file-dump or git-annex or datalad could be used (not done myself)



Continuous Benchmarking

How to visualize results

For large visualizations, a server with e.g. Graphana could be used

$\Rightarrow$ Backend server needed

$\Rightarrow$ Faster to provide updated data

Continuous Benchmarking

How to visualize results

For large visualizations, a server with e.g. Graphana could be used

⇒ Backend server needed

⇒ Faster to provide updated data

For smaller versions, static html with embedded JS is an alternative (e.g. via plotly)

⇒ Simple webserver sufficient

⇒ Easily scalable, as only static html-files are served

Continuous Benchmarking

How to visualize results

For large visualizations, a server with e.g. Graphana could be used

⇒ Backend server needed

⇒ Faster to provide updated data

For smaller versions, static html with embedded JS is an alternative (e.g. via plotly)

⇒ Simple webserver sufficient

⇒ Easily scalable, as only static html-files are served

This could be combined with Jupyter NB for easy development and with mkdocs if multiple pages shall be deployed.

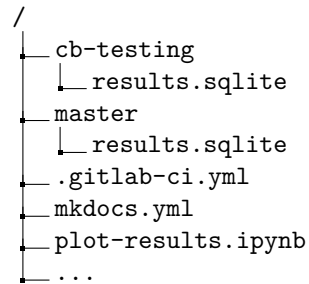
⇒ Simple creation of sites and automatic creation of menu and footers

Continuous Benchmarking

How to visualize results

Example for website with Jupyter Notebooks and mkdocs

- Separate sqlite-files for each branch

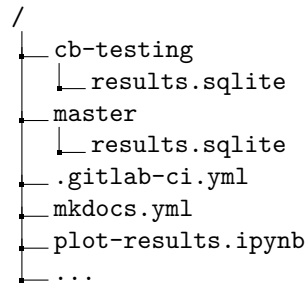


Continuous Benchmarking

How to visualize results

Example for website with Jupyter Notebooks and mkdocs

- Separate sqlite-files for each branch
- Usage of template file of Jupyter NB
- This get copied for each branch automatically in CI

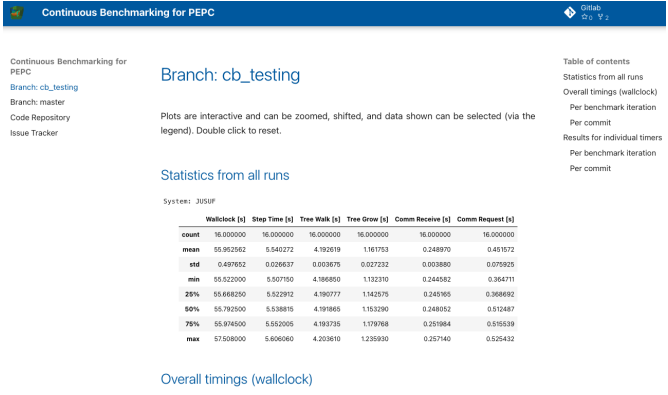


Continuous Benchmarking

How to visualize results

Example for website with Jupyter Notebooks and mkdocs

- Separate sqlite-files for each branch
- Usage of template file of Jupyter NB
- This get copied for each branch automatically in CI
- Created website



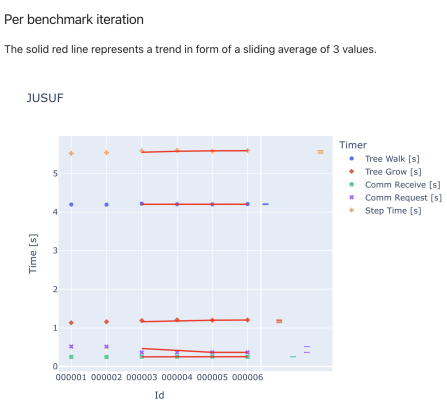
	Wallclock [s]	Step Time [s]	Tree Walk [s]	Tree Grow [s]	Comm Receive [s]	Comm Request [s]
count	16.000000	16.000000	16.000000	16.000000	16.000000	16.000000
mean	55.952562	5.540272	4.192619	1.161753	0.248970	0.451572
std	0.497652	0.026637	0.003675	0.027232	0.003880	0.075925
min	55.522000	5.507150	4.186850	1.132310	0.244582	0.364711
25%	55.668250	5.522912	4.190777	1.142575	0.245165	0.368692
50%	55.792500	5.538815	4.191865	1.153290	0.248052	0.512487
75%	55.974500	5.552005	4.193735	1.179768	0.251984	0.515539
max	57.508000	5.606060	4.203610	1.235930	0.257140	0.525432

Continuous Benchmarking

How to visualize results

Example for website with Jupyter Notebooks and mkdocs

- Separate sqlite-files for each branch
- Usage of template file of Jupyter NB
- This get copied for each branch automatically in CI
- Created website
- With interactive graphs



JÜLICH Forschungszentrum

Thank you for your attention!

Link to GitHub2Lab:
`https://github.com/jakob-fritz/github2lab_action`

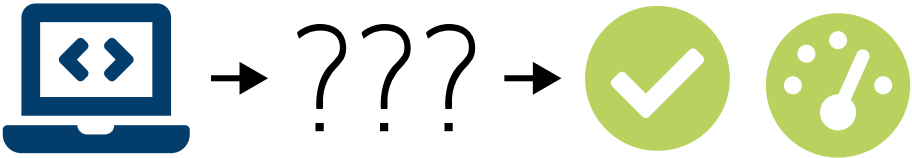
Link to CB-Website of PEPC:
`https://slpp.pages.jsc.fz-juelich.de/pepc/pepc/`

PEPC

JÜLICH Forschungszentrum

Thank you for your attention!

If you are interested in the details of the codes or have questions,
feel free to contact me!
j.fritz@fz-juelich.de



CONTINUOUS INTEGRATION FOR HPC CODES

JLESC, April 17, 2024 | Jakob Fritz, Robert Speck | Jülich Supercomputing Centre, Forschungszentrum Jülich