Monitoring software performance with LHCbPR

Maciej Szymański
University of Chinese Academy of Sciences

Kraków, 25 Jun 2018

LHCb Performance and Regression framework

- **LHCbPR** is a framework for systematic monitoring of the LHCb software
- Provides performance baseline in **controlled conditions**
- Enables to **inspect any changes** due to e.g. MC generators, physics of Geant4, new external libraries, new MRs, DDDB tags, …
- **Compare results** across different compilers and architectures
- Not only to monitor resource consumption, but also to **measure the physics performance**
- Cf. nightly tests: larger statistics and more than boolean value

Infrastructure

▸ TWiki

- Periodic tests started by the **Jenkins** job
  - ▸ Configuration of Jenkins job
  - tests triggered when corresponding nightly builds ready (using RabbitMQ)

Infrastructure

▸ TWiki

- Periodic tests started by the **Jenkins** job
  - ▸ Configuration of Jenkins job
  - tests triggered when corresponding nightly builds ready (using RabbitMQ)
- Configuration in **XML** files
  - ▸ LHCbNightlyConf

Infrastructure

▸ TWiki

- Periodic tests started by the **Jenkins** job
  - ▸ Configuration of Jenkins job
  - tests triggered when corresponding nightly builds ready (using RabbitMQ)
- Configuration in **XML** files
  - ▸ LHCbNightlyConf
- Machines that tests are currently running on
  - `lblhcbpr1` with CC7 dedicated for timing tests (single executor in Jenkins), label: `perf-centos7-timing`
  - `lblhcbpr4` with CC7 (8 executors), labels: `perf-centos7, perf`
  - `volhcb05` with SLC6 (8 executors), labels: `perf-slc6, perf`
  - `hltperf-quanta01-e52630v4` for HLT throughput test

Infrastructure

- Results of the tests **parsed by the specific handlers**
  - LHCbPR2HD
  - to save relevant metrics (int, float, string, files, json)

Infrastructure

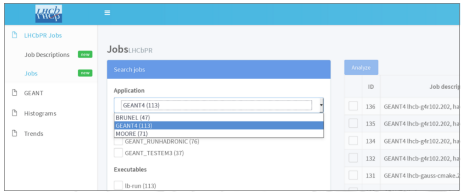
- Results of the tests **parsed by the specific handlers**
  - ▸ LHCbPR2HD
  - to save relevant metrics (int, float, string, files, json)
- Zip file sent to the database through **Dirac Storage Element** `/lhcb/prdata/zips`
  - ▸ LHCbPR2BE

Infrastructure

- Results of the tests **parsed by the specific handlers**
  - ▸ LHCbPR2HD
  - to save relevant metrics (int, float, string, files, json)
- Zip file sent to the database through **Dirac Storage Element** `/lhcb/prdata/zips`
  - ▸ LHCbPR2BE

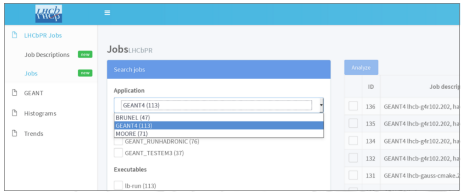
Infrastructure

- Results of the tests **parsed by the specific handlers**
  - ▸ LHCbPR2HD
  - to save relevant metrics (int, float, string, files, json)
- Zip file sent to the database through **Dirac Storage Element**
  `/lhcb/prdata/zips`
  - ▸ LHCbPR2BE
- **Web front–end** ▸ lblhcbpr.cern.ch
  - ▸ LHCbPR2FE
  - generic ROOT files viewer
  - trend analysis
  - custom modules



Infrastructure

- Results of the tests **parsed by the specific handlers**
  - ▸ LHCbPR2HD
  - to save relevant metrics (int, float, string, files, json)
- Zip file sent to the database through **Dirac Storage Element**
  `/lhcb/prdata/zips`
  - ▸ LHCbPR2BE
- **Web front–end** ▸ lblhcbpr.cern.ch
  - ▸ LHCbPR2FE
  - generic ROOT files viewer
  - trend analysis
  - custom modules
- Flexibility to push the results as HTML to EOS (at the level of LHCbPR2HD)
  - HLT case (rate and throughput tests)



Some statistics as of today

- 5 applications
  - Brunel
  - Gauss
  - Geant4
  - Moore
  - MooreOnline
- 44 option files
- 115 tests (running on several slots and platforms)
- 20–51 tests daily
- 14 tests dedicated for timing (thus running only on `lblhcbpr1` machine)
- Single tests run from several minutes up to 10 hours

How to take part

- Prepare the **options file** for the test and push to e.g. PRConfig

How to take part

- Prepare the **options file** for the test and push to e.g. PRConfig
- Specify the **command** to run
  - see whether it's already defined: ▸ lblhcbpr.cern.ch/api/executables, if not, we'll add it

```
"name": "lb-run-gaudirun",
"content": "lb-run -c {platform} --user-area={build} {app_name}/{app_version}
           gaudirun.py {options}"


"name": "lb-run-callgrind",
"content": "( lb-run -c {platform} --user-area={build} {app_name}/{app_version} gaudirun.py
              --printsequence {options} ; lb-run -c {platform} --user-area={build}
              {app_name}/{app_version} valgrind --tool=callgrind --dump-instr=yes
              --instr-atstart=no --cache-sim=yes --branch-sim=yes python
              $(lb-run -c {platform} --user-area={build} {app_name}/{app_version}
                       which gaudirun.py) {options} )"


"name": "perf-lb-run-gaudirun",
"content": "( perf record --call-graph=lbr -o perf.log lb-run -c {platform}
           --user-area={build} {app_name}/{app_version} gaudirun.py {options} ;
           perf report -i perf.log > perf.lbr.txt )"
```

How to take part

- Prepare the **options file** for the test and push to e.g. PRConfig
- Specify the **command** to run
  - see whether it's already defined: ▸ lblhcbpr.cern.ch/api/executables, if not, we'll add it

```
"name": "lb-run-gaudirun",
"content": "lb-run -c {platform} --user-area={build} {app_name}/{app_version}
           gaudirun.py {options}"


"name": "lb-run-callgrind",
"content": "( lb-run -c {platform} --user-area={build} {app_name}/{app_version} gaudirun.py
              --printsequence {options} ; lb-run -c {platform} --user-area={build}
              {app_name}/{app_version} valgrind --tool=callgrind --dump-instr=yes
              --instr-atstart=no --cache-sim=yes --branch-sim=yes python
              $(lb-run -c {platform} --user-area={build} {app_name}/{app_version}
                       which gaudirun.py) {options} )"


"name": "perf-lb-run-gaudirun",
"content": "( perf record --call-graph=lbr -o perf.log lb-run -c {platform}
           --user-area={build} {app_name}/{app_version} gaudirun.py {options} ;
           perf report -i perf.log > perf.lbr.txt )"
```

- Create the **handler** to parse the output
  - many handlers already there, e.g. to parse `TimingAuditor`, output of `perf`, etc.
  - see README on: ▸ LHCbPR2HD

How to take part

- **Schedule** your test in ▸ LHCbNightlyConf

How to take part

- **Schedule** your test in LHCbNightlyConf

```
<periodictest>
  <schedule type="week" time="10:00">Mon,Tue,Wed,Thu,Fri</schedule>
  <slot>lhcb-future</slot>
  <project>Brunel</project>
  <platform>x86_64-slc6-gcc62-opt</platform>
  <test runner="lhcbpr" group="MiniBrunel" env="lb-run-gaudirun|TimeLineHandler"/>
  <os_label>perf</os_label>
  <count>5</count>
</periodictest>
```

How to take part

- **Schedule** your test in LHCbNightlyConf


```
<periodictest>
  <schedule type="week" time="10:00">Mon,Tue,Wed,Thu,Fri</schedule>
  <slot>lhcb-future</slot>
  <project>Brunel</project>
  <platform>x86_64-slc6-gcc62-opt</platform>
  <test runner="lhcbpr" group="MiniBrunel" env="lb-run-gaudirun|TimeLineHandler"/>
  <os_label>perf</os_label>
  <count>5</count>
</periodictest>
```

How to take part

- **Schedule** your test in ▸ LHCbNightlyConf

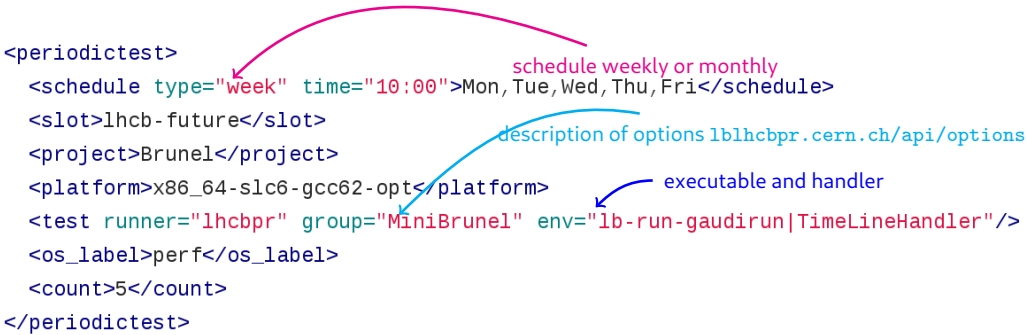
```
<periodictest>
  <schedule type="week" time="10:00">Mon,Tue,Wed,Thu,Fri</schedule>
  <slot>lhcb-future</slot>
  <project>Brunel</project>
  <platform>x86_64-slc6-gcc62-opt</platform>
  <test runner="lhcbpr" group="MiniBrunel" env="lb-run-gaudirun|TimeLineHandler"/>
  <os_label>perf</os_label>
  <count>5</count>
</periodictest>
```

schedule weekly or monthly

description of options `lblhcbpr.cern.ch/api/options`

How to take part

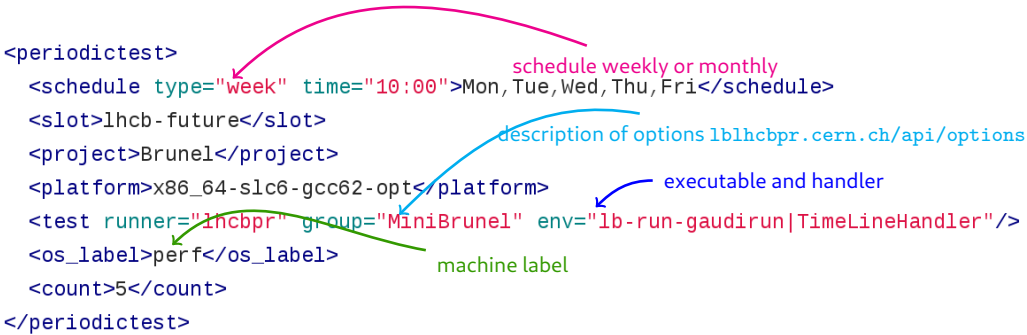
- **Schedule** your test in ▸ LHCbNightlyConf




```
<periodictest>
  <schedule type="week" time="10:00">Mon,Tue,Wed,Thu,Fri</schedule>
  <slot>lhcb-future</slot>
  <project>Brunel</project>
  <platform>x86_64-slc6-gcc62-opt</platform>
  <test runner="lhcbpr" group="MiniBrunel" env="lb-run-gaudirun|TimeLineHandler"/>
  <os_label>perf</os_label>
  <count>5</count>
</periodictest>
```

How to take part

- **Schedule** your test in ▸ LHCbNightlyConf



How to use it

- Tests will automatically start on a day given by schedule (if the nightly build is ok)

How to use it

- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- Watch the tests being executed in ▸ dashboard
  - colour code: running tests, successful tests, failed tests, tests which have been executed with success, but the handler failed
  - URL to log files of the test, handler and output of the jenkins job

How to use it

- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- Watch the tests being executed in ▸ dashboard
  - colour code: running tests, successful tests, failed tests, tests which have been executed with success, but the handler failed
  - URL to log files of the test, handler and output of the jenkins job
- You can launch the test yourself from the dashboard
  - e.g. to the test the handler
  - click on **Start new periodic test** button (available after login)

How to use it

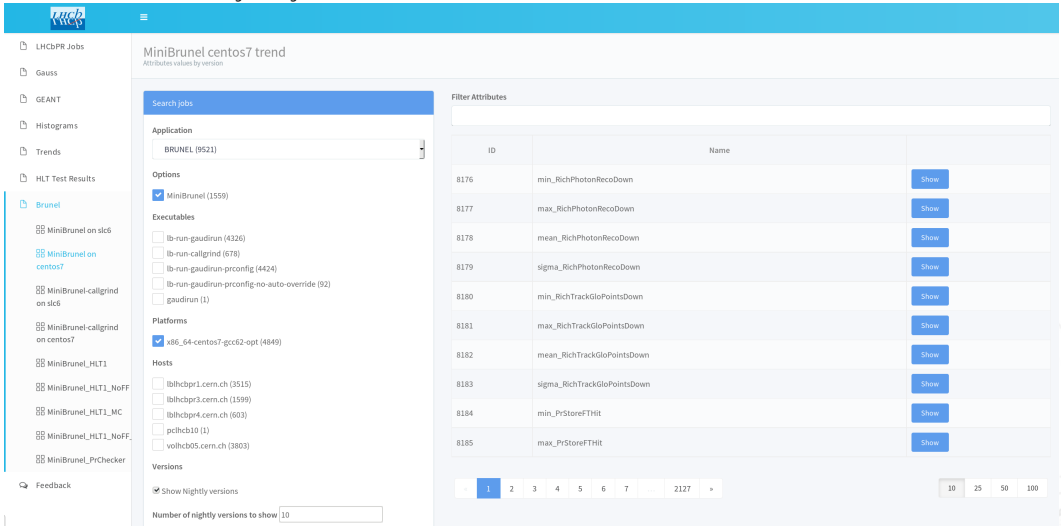
- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- Watch the tests being executed in ▸ dashboard
  - colour code: running tests, successful tests, failed tests, tests which have been executed with success, but the handler failed
  - URL to log files of the test, handler and output of the jenkins job
- You can launch the test yourself from the dashboard
  - e.g. to the test the handler
  - click on **Start new periodic test** button (available after login)
- To see the results of the test, by default you can use generic **trend analysis** and **ROOT file viewer** on `lblhcbpr.cern.ch`

How to use it

- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- Watch the tests being executed in ▸ dashboard
  - colour code: running tests, successful tests, failed tests, tests which have been executed with success, but the handler failed
  - URL to log files of the test, handler and output of the jenkins job
- You can launch the test yourself from the dashboard
  - e.g. to the test the handler
  - click on **Start new periodic test** button (available after login)
- To see the results of the test, by default you can use generic **trend analysis** and **ROOT file viewer** on `lblhcbpr.cern.ch`
- (Optionally) create custom **analysis module** ▸ LHCbPR2FE
  - or re-use existing one ...

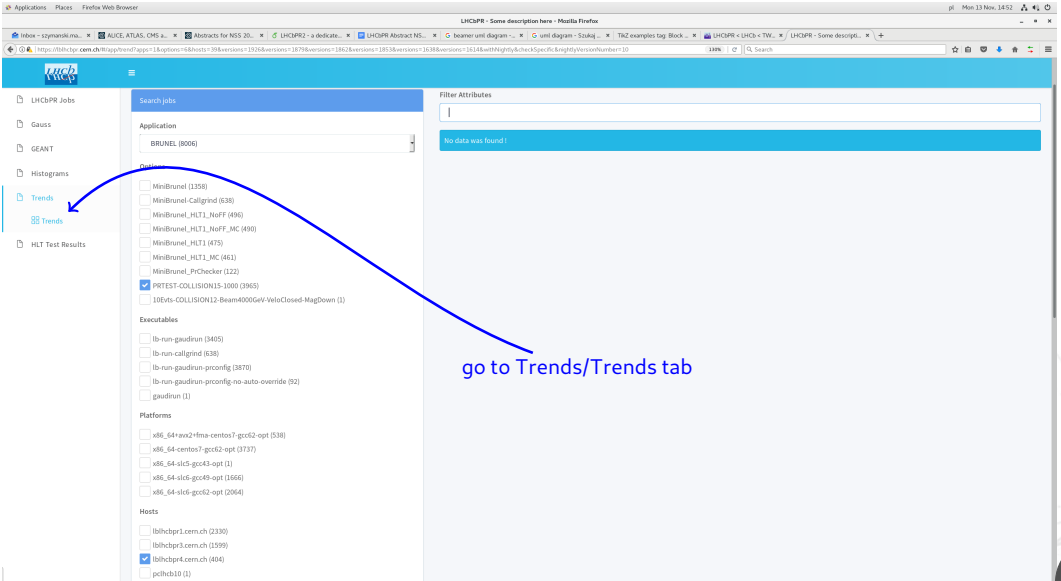
Trend module with predefined parameters

Plot last 10 measurements for a given algorithm



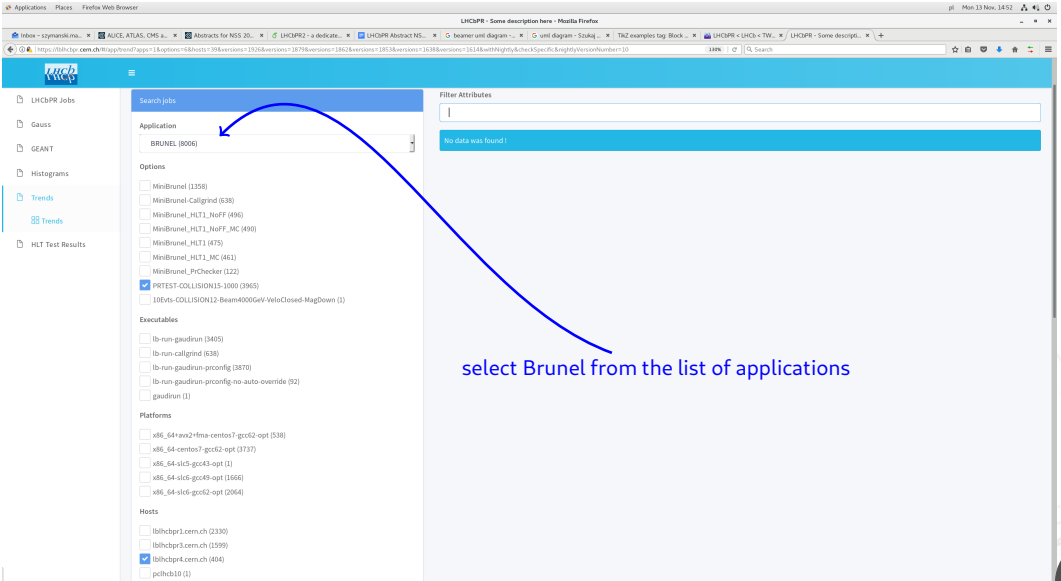
Using trend analysis

Plot the time spent by EVENT LOOP in Brunel test as a function of the software version



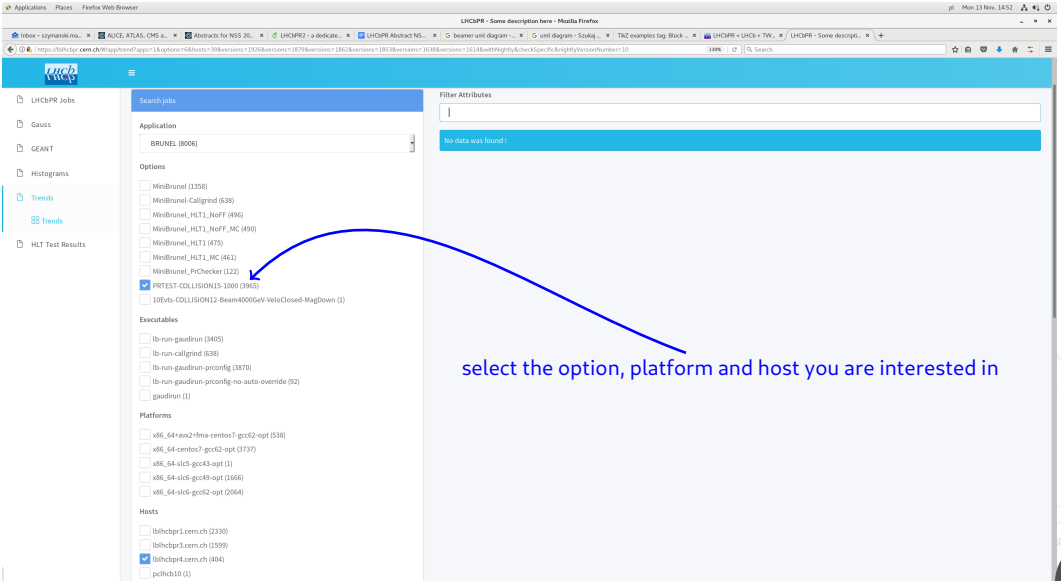
Using trend analysis

Plot the time spent by EVENT LOOP in Brunel test as a function of the software version



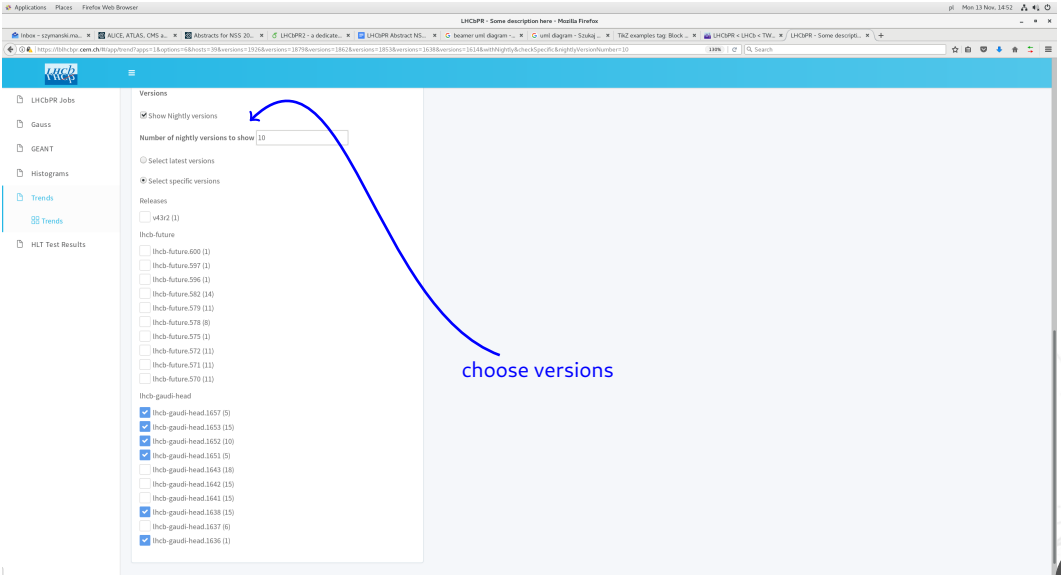
Using trend analysis

Plot the time spent by EVENT LOOP in Brunel test as a function of the software version



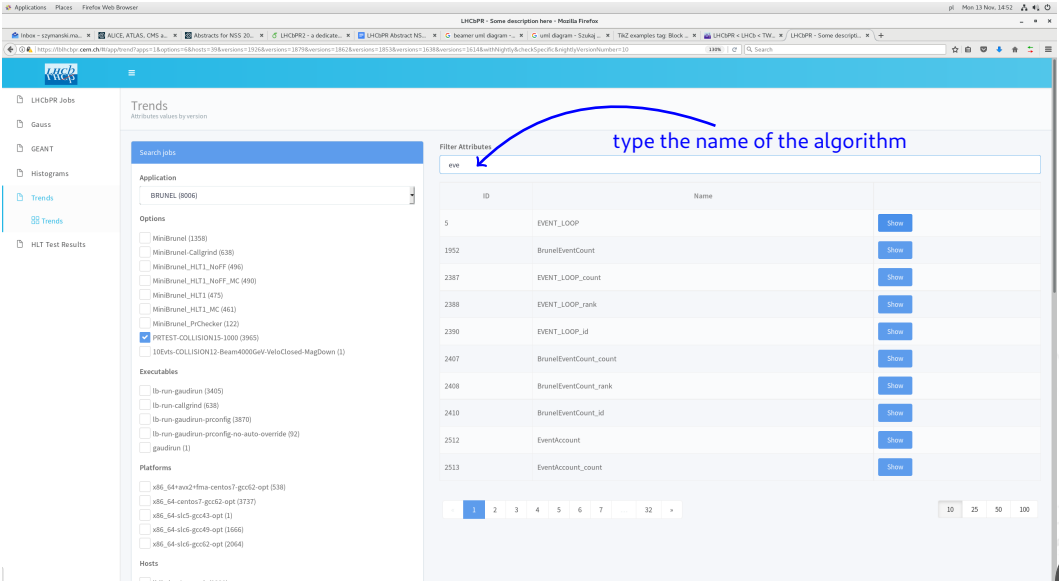
Using trend analysis

Plot the time spent by EVENT LOOP in Brunel test as a function of the software version



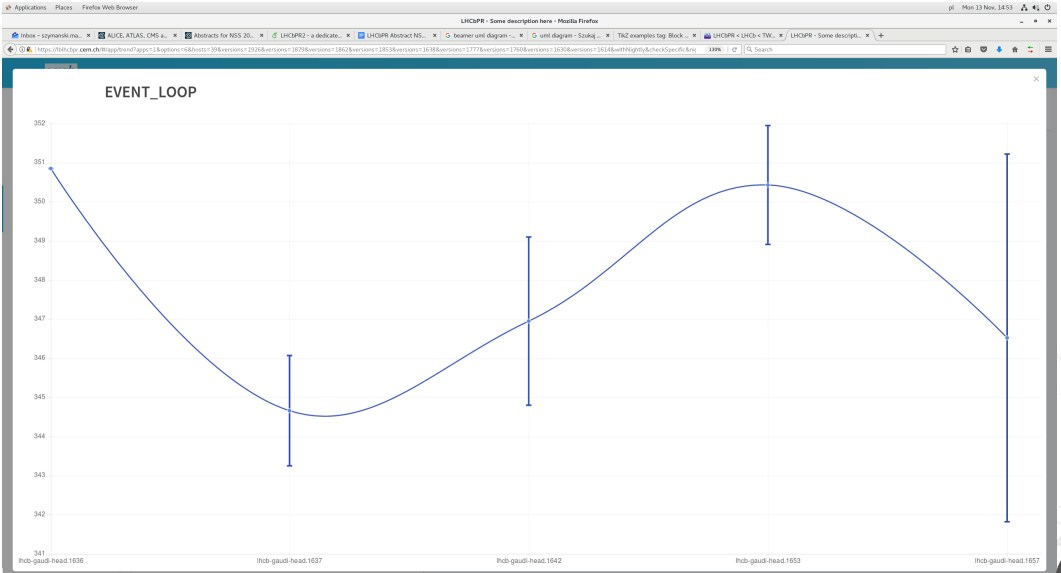
Using trend analysis

Plot the time spent by EVENT LOOP in Brunel test as a function of the software version

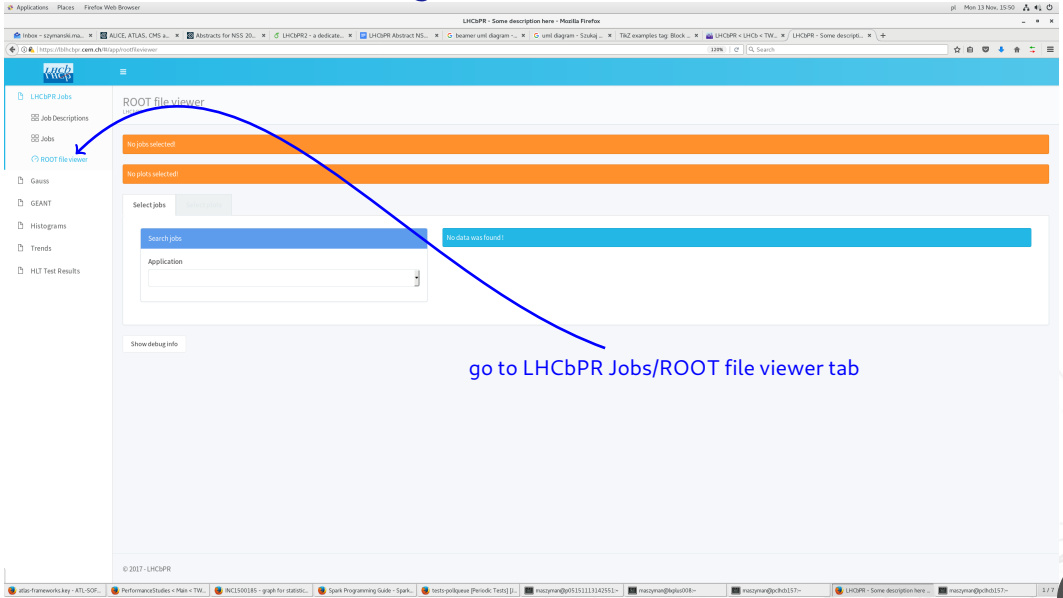


Using trend analysis

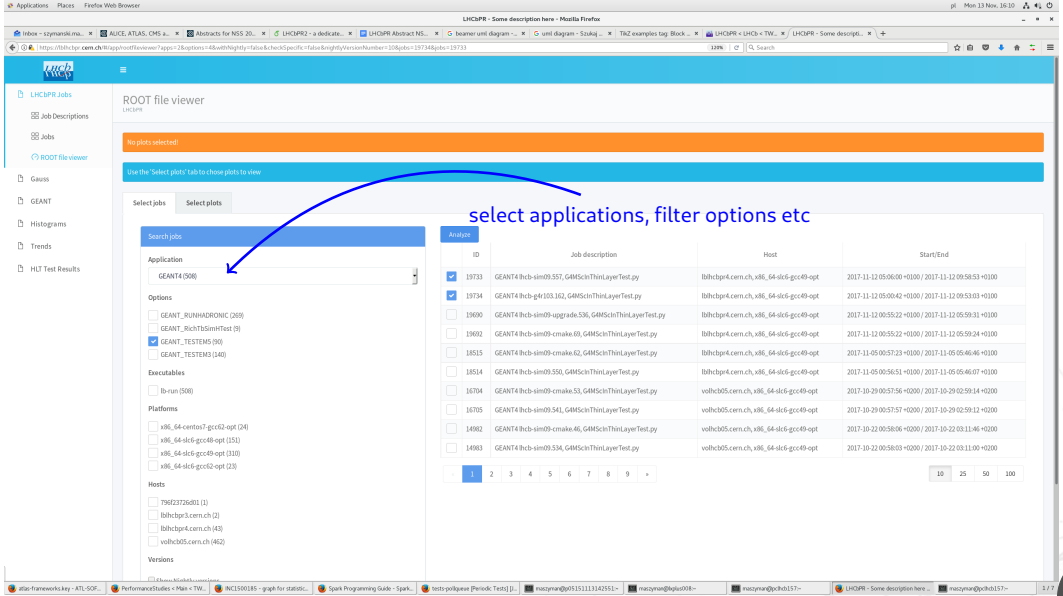
Plot the time spent by EVENT LOOP in Brunel test as a function of the software version



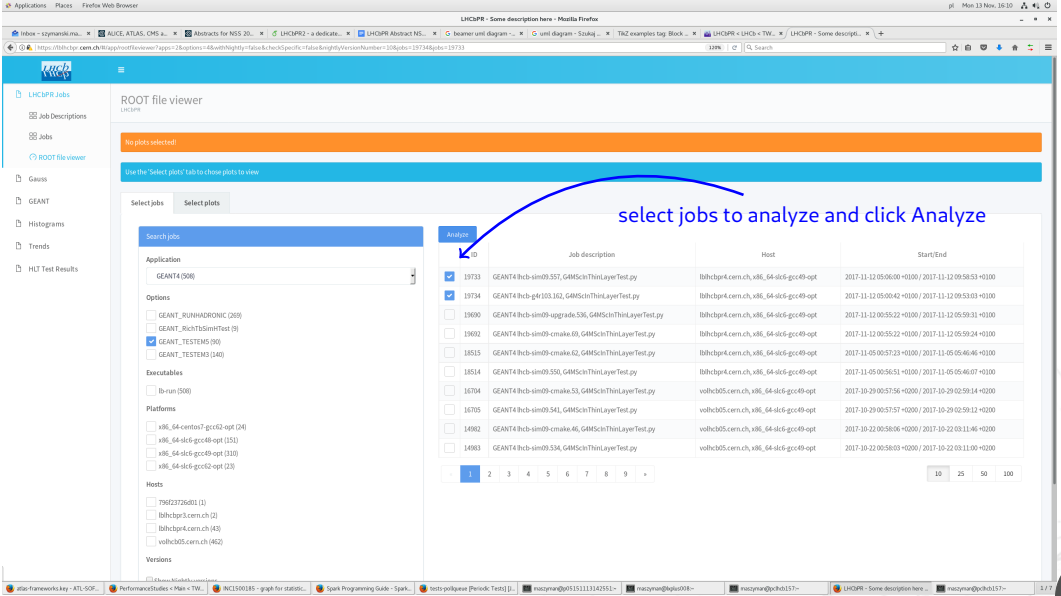
Using ROOT file viewer



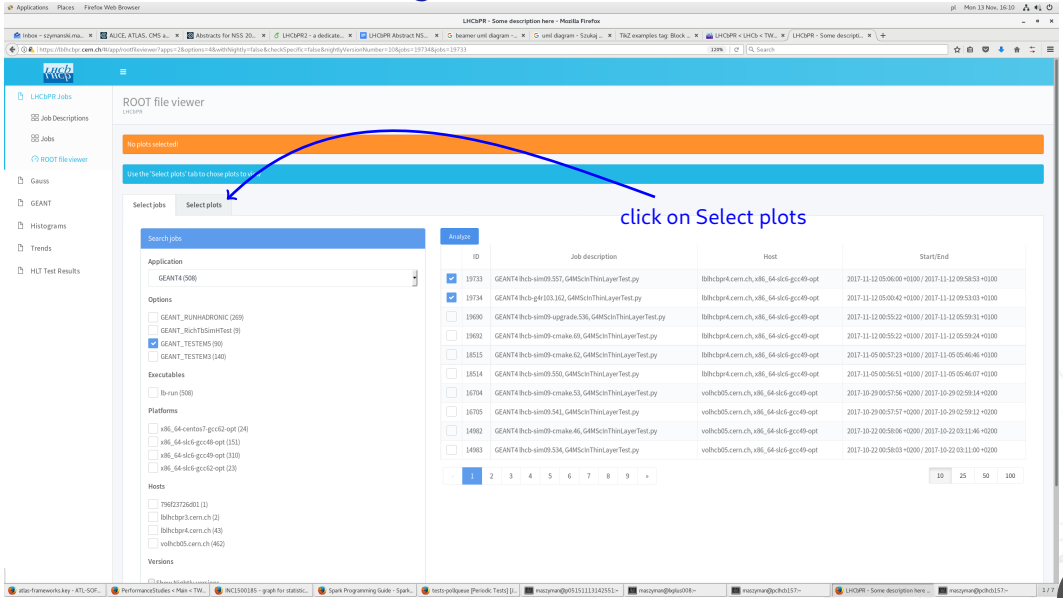
Using ROOT file viewer



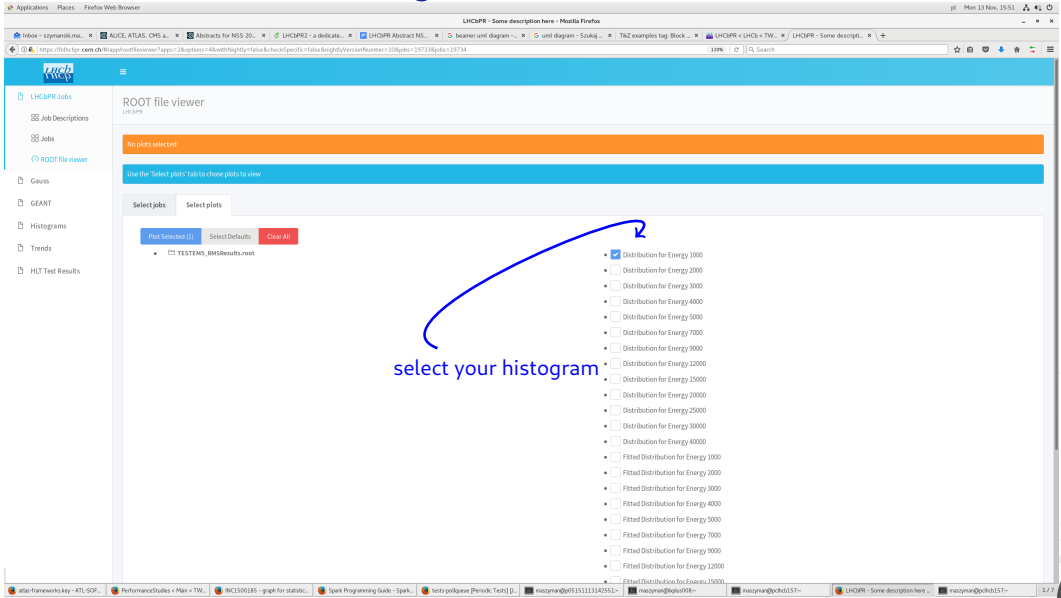
Using ROOT file viewer



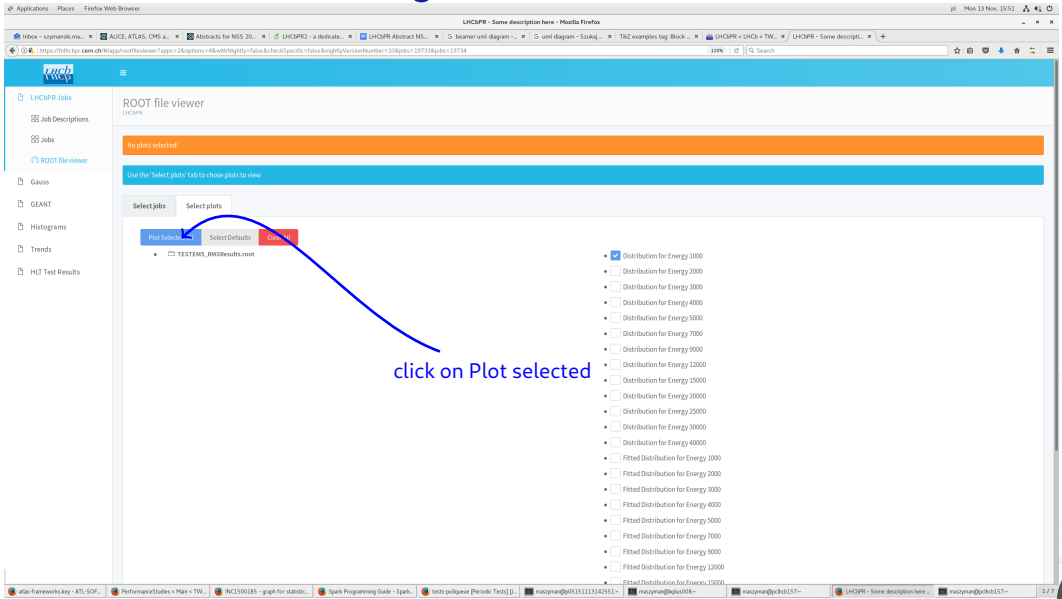
Using ROOT file viewer



Using ROOT file viewer



Using ROOT file viewer



click on Plot selected

Using ROOT file viewer

Thank you!
Please give feedback on our ‣ mattermost channel

LHCbPR - Resources

- Web application:
  `https://lblhcbpr.cern.ch`
  `https://lblhcbpr.cern.ch/api/`
  `https://gitlab.cern.ch/lhcb-core/LHCbPR2FE`
- API service:
  `https://gitlab.cern.ch/lhcb-core/LHCbPR2BE`
- ROOT HTTP service:
  `https://gitlab.cern.ch/lhcb-core/LHCbPR2ROOT`
- Tests' output handlers:
  `https://gitlab.cern.ch/lhcb-core/LHCbPR2HD`
- Project builder:
  `https://gitlab.cern.ch/lhcb-core/LHCbPR2`
- Jenkins configuration
  `https://gitlab.cern.ch/lhcb-core/LbNightlyTools`
- Configuration of the periodic tests
  `https://gitlab.cern.ch/lhcb-core/LHCbNightlyConf/`