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
  - `lbhltperf01` for upgrade tests

Infrastructure

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

python™

Infrastructure

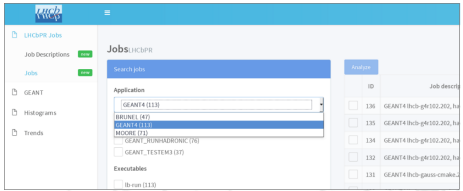
- Results of the tests **parsed by the specific handlers**
  - ▸ LHCbPR2HD
  - to save relevant metrics (int, float, string, ROOT, JSON)
- Zip file sent to the database through **Dirac Storage Element** `/lhcb/prdata/zips`
  - ▸ LHCbPR2BE

Infrastructure

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

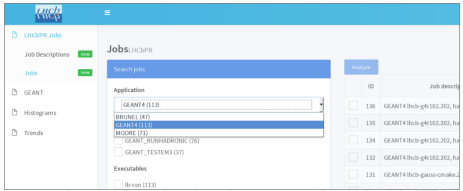
Infrastructure

- Results of the tests **parsed by the specific handlers**
  - ▸ LHCbPR2HD
  - to save relevant metrics (int, float, string, ROOT, 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, ROOT, 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 rate and throughput tests
  - upgrade benchmarking



Some statistics as of today

- 5 applications
  - Brunel
  - Gauss
  - Geant4
  - Moore
  - MooreOnline
- 67 option files
- 128 tests (running on several slots and platforms)
- ∼50 tests daily
- ∼20 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 commit to e.g. PRConfig

How to take part

- Prepare the **options file** for the test and commit 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 commit 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

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>
```

schedule weekly or monthly

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

executable and handler

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`

executable and handler

machine label

How to use LHCbPR

- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- You will see **PR** link in the nightlies dashboard, close to the project name in the given slot

How to use LHCbPR

- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- You will see **PR** link in the nightlies dashboard, close to the project name in the given slot
- 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 LHCbPR

- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- You will see **PR** link in the nightlies dashboard, close to the project name in the given slot
- 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 LHCbPR

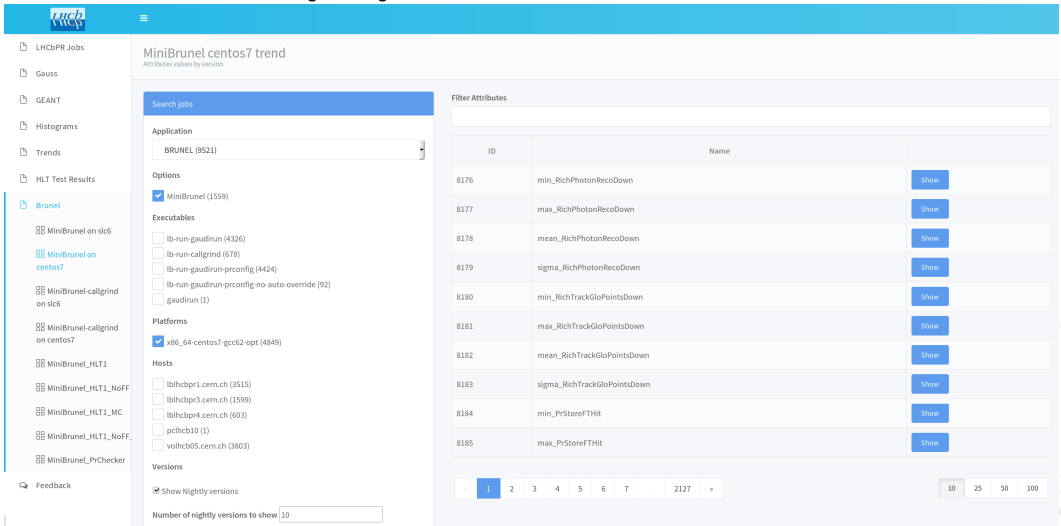
- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- You will see **PR** link in the nightlies dashboard, close to the project name in the given slot
- Watch the tests being executed in ▸ dashboard
  - colour code: running tests, successful tests, failed tests, tests 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 LHCbPR

- Tests will automatically start on a day given by schedule (if the nightly build is ok)
- You will see **PR** link in the nightlies dashboard, close to the project name in the given slot
- 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 ...

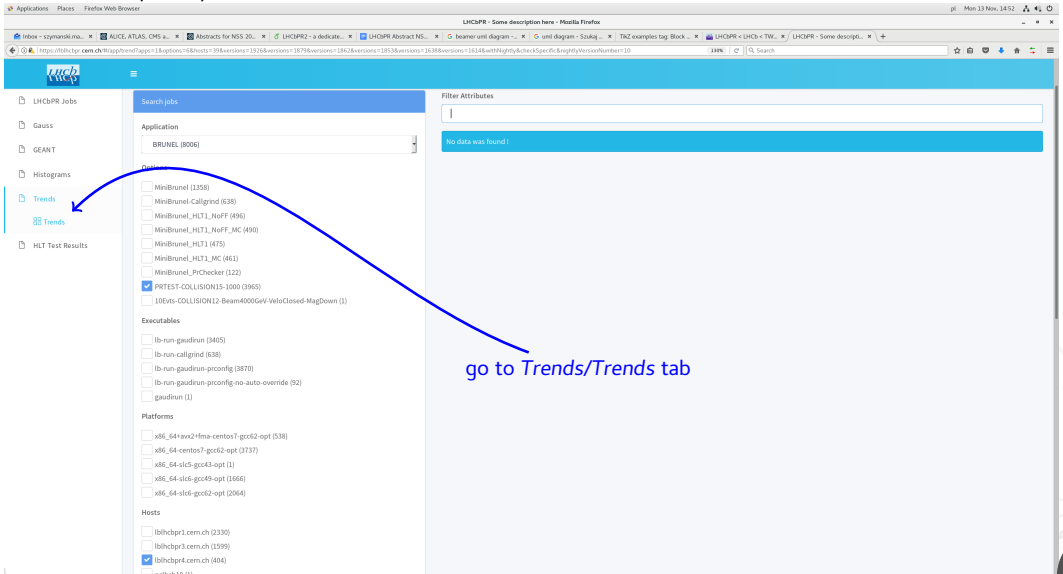
Trend module with predefined parameters

Plot last 10 measurements for a given algorithm



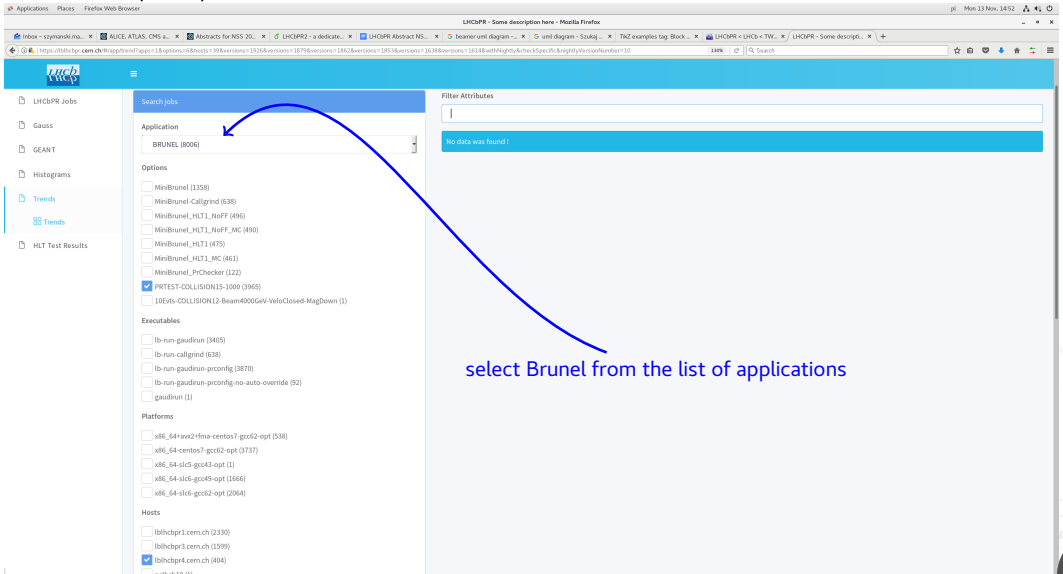
Using trend analysis

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



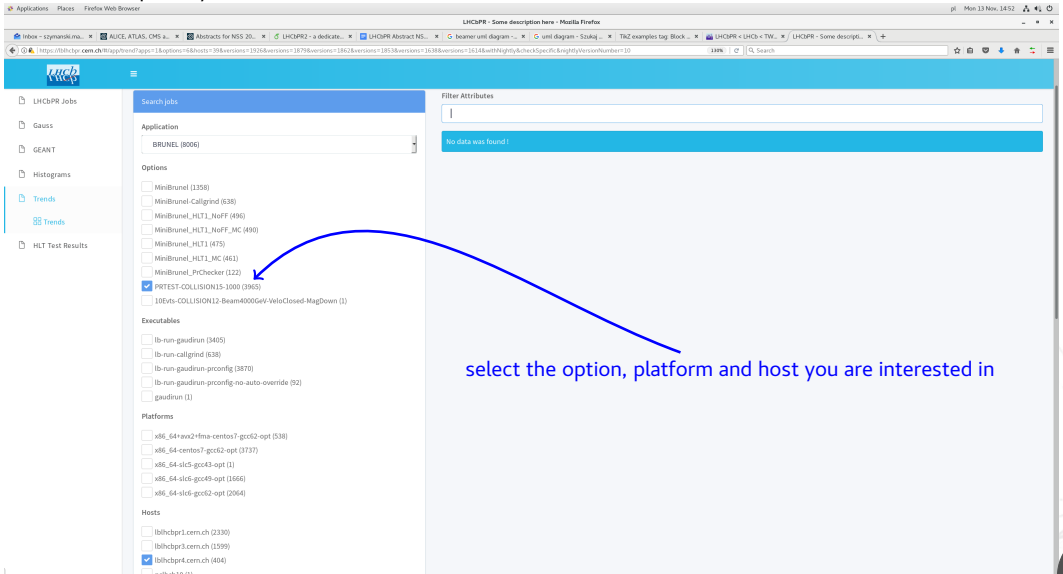
Using trend analysis

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



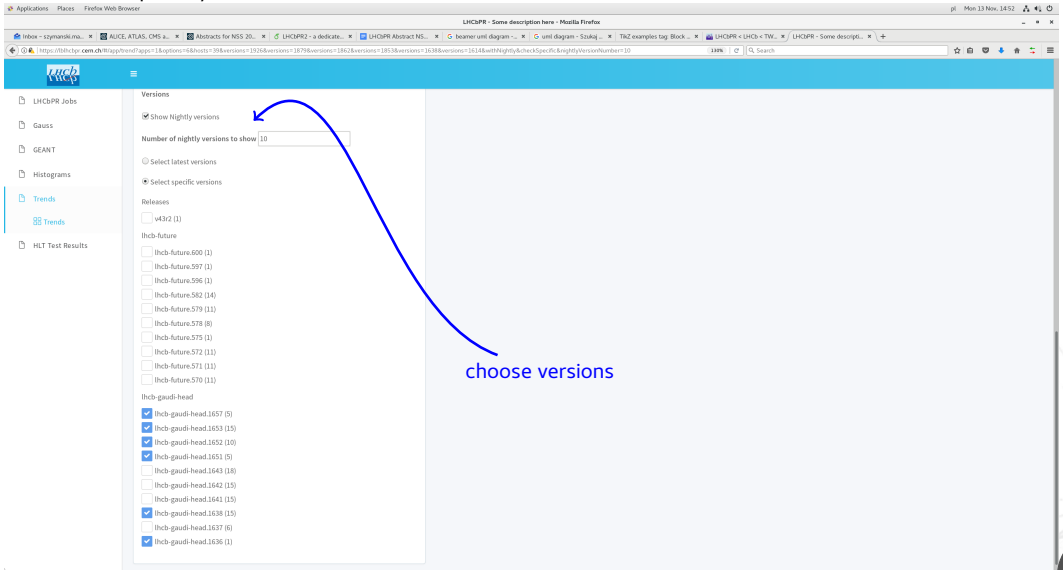
Using trend analysis

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



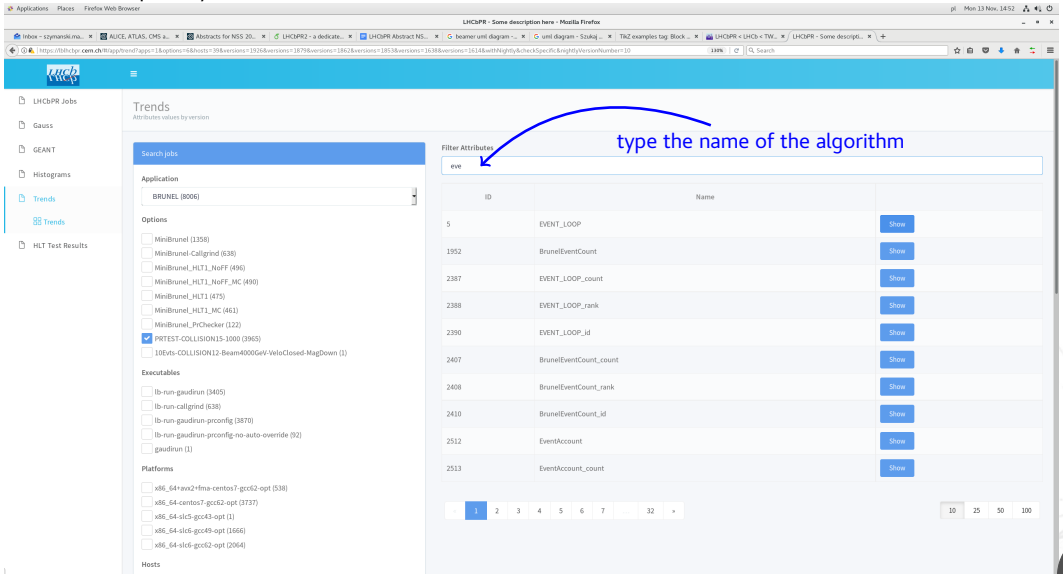
Using trend analysis

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



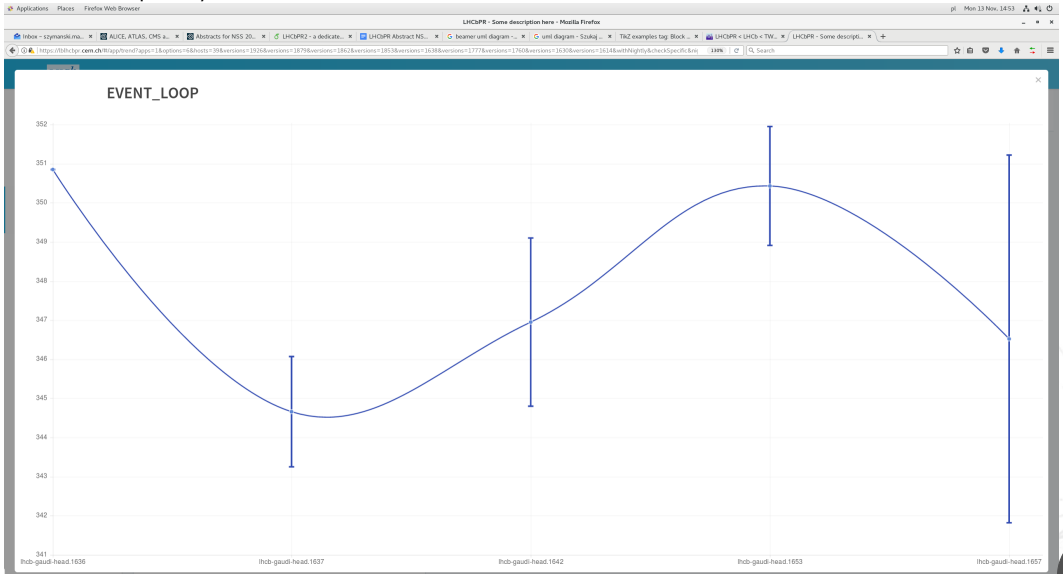
Using trend analysis

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

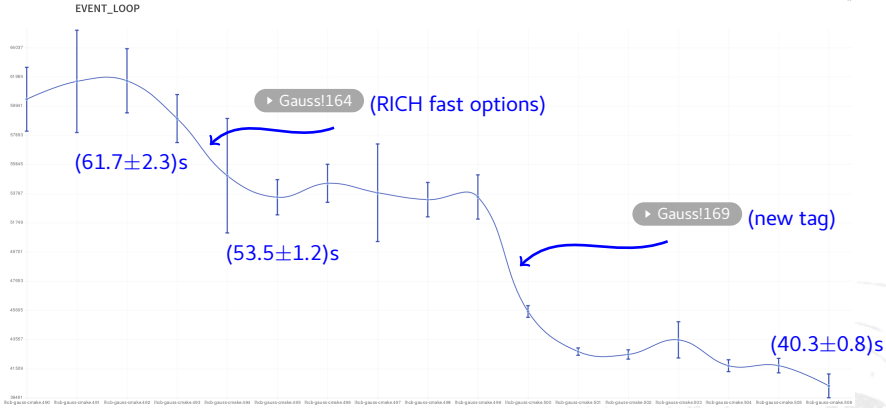


Using trend analysis

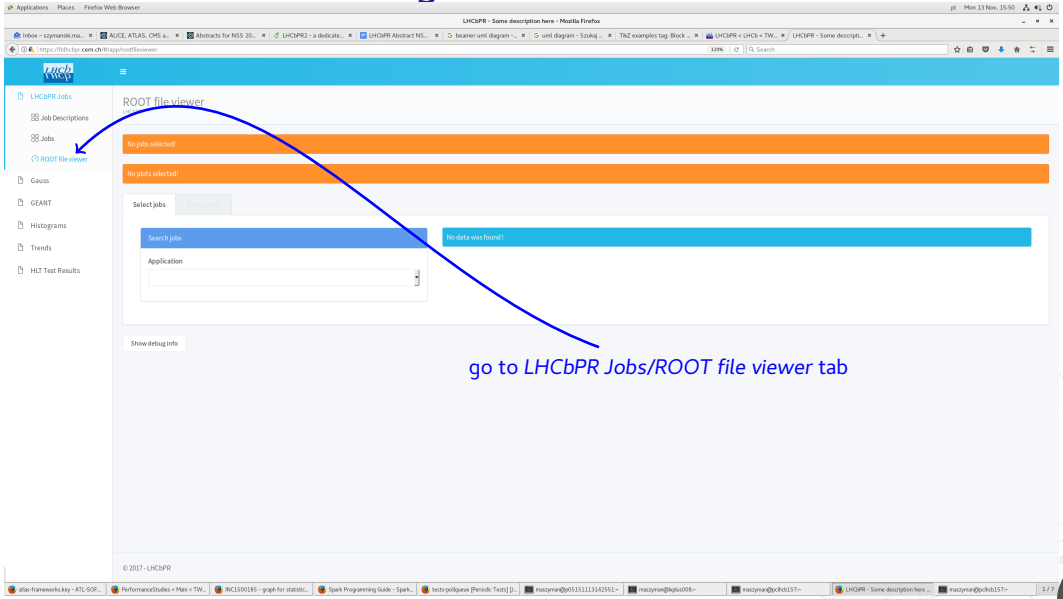
Plot the time spent by EVENT LOOP in Brunel test as a function of the software version



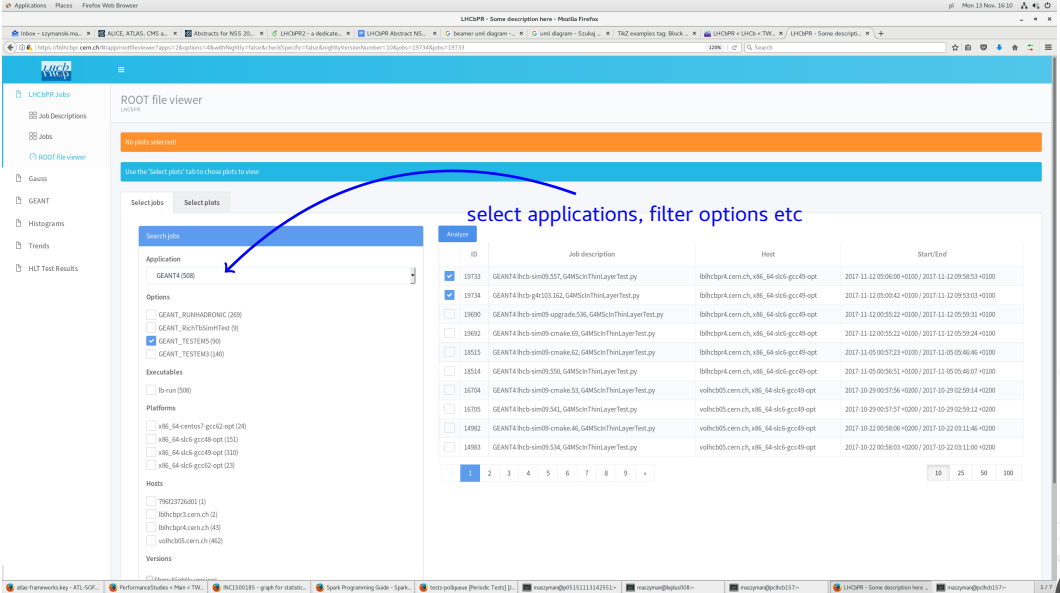
Event loop timing in Gauss vs. SW version



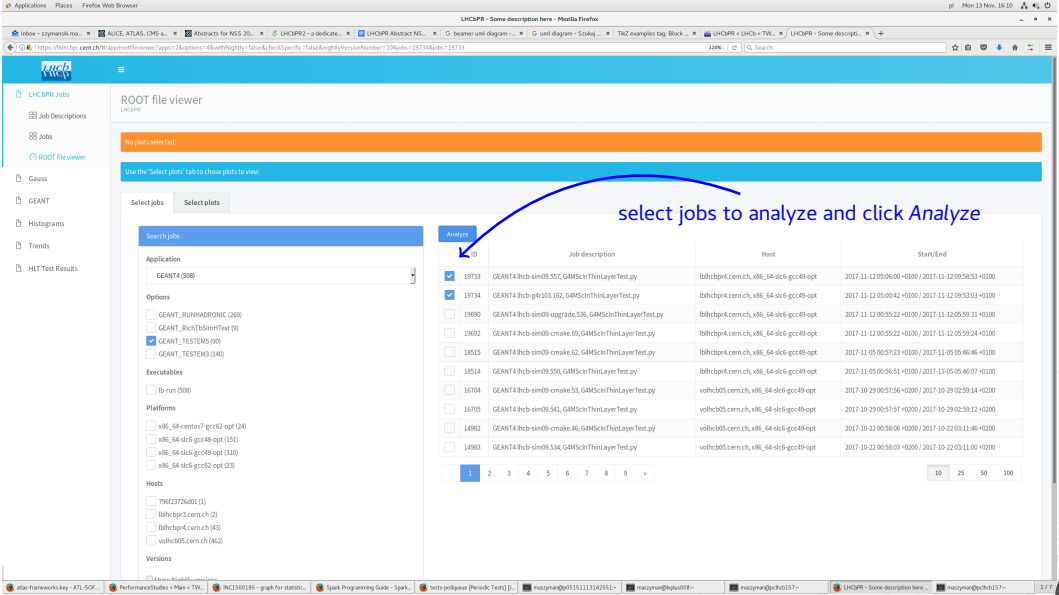
Using ROOT file viewer



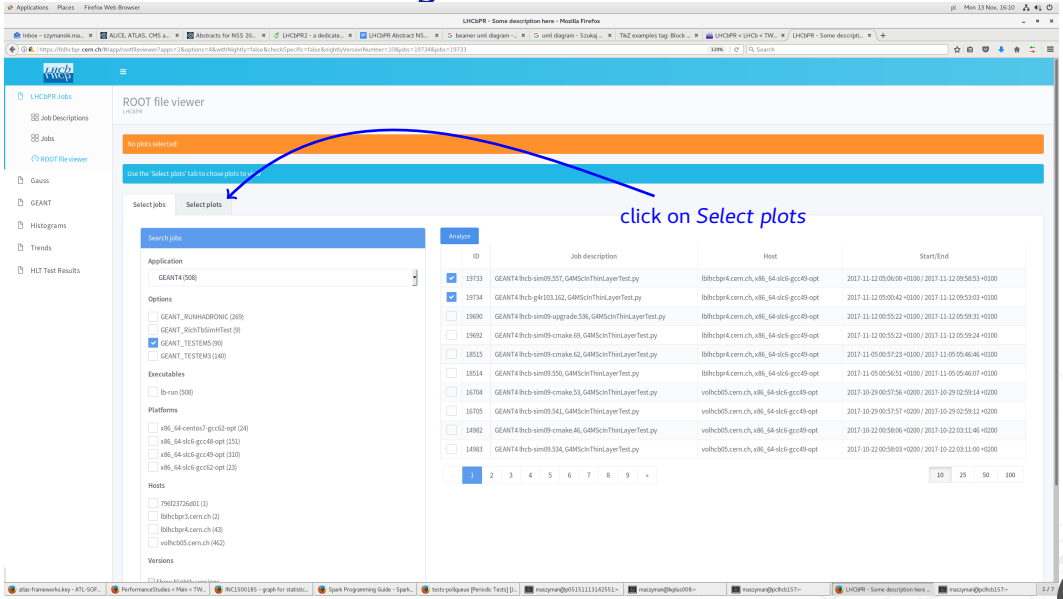
Using ROOT file viewer



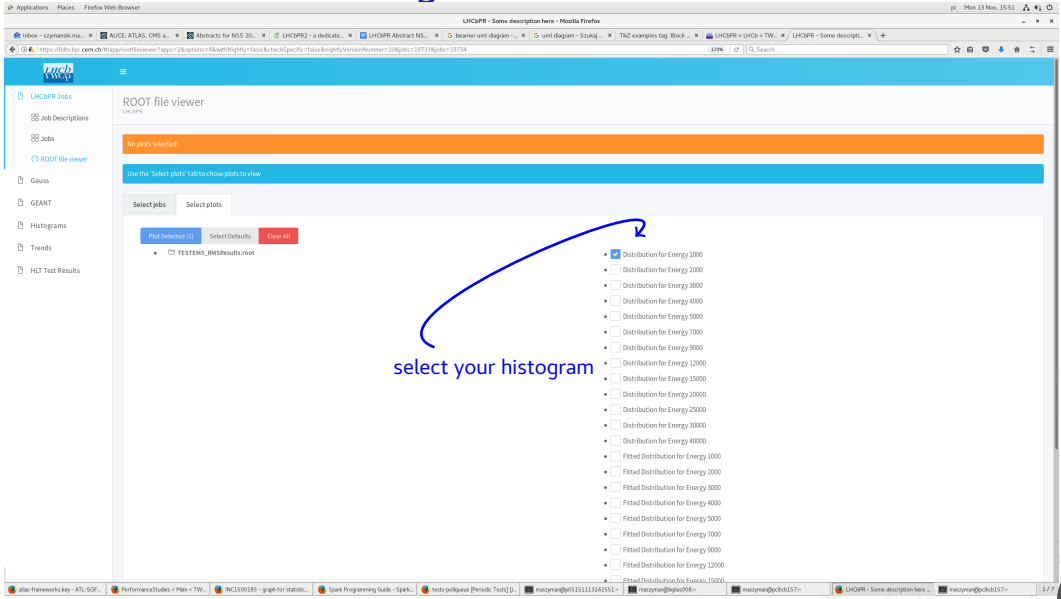
Using ROOT file viewer



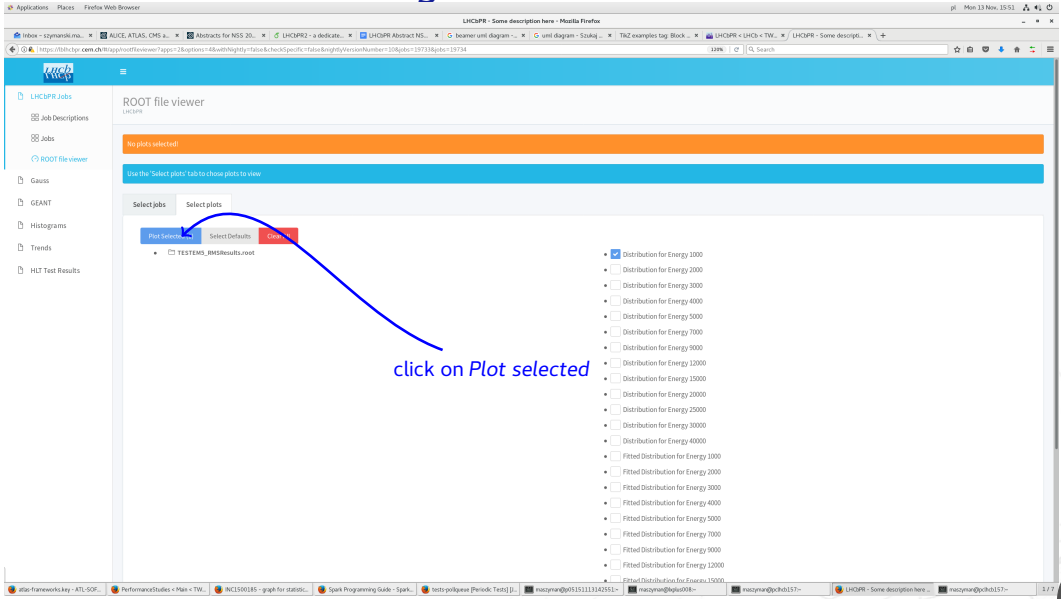
Using ROOT file viewer



Using ROOT file viewer



Using ROOT file viewer



Using ROOT file viewer

Upgrade benchmarking

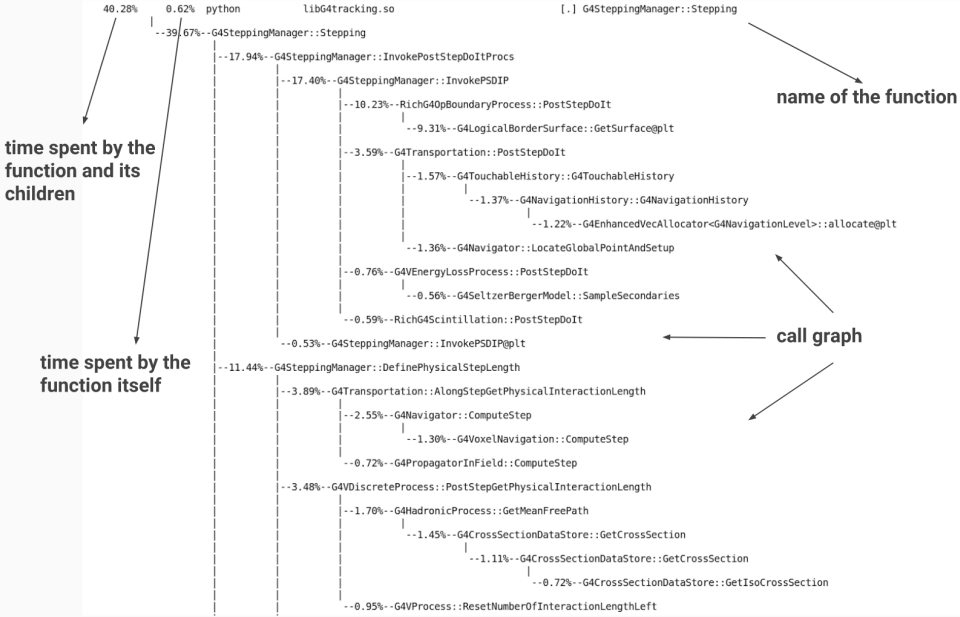
- Special case for the upgrade studies
- Tests results pushed to eos website:
  `https://cern.ch/lhcbpr-hlt/PerfTests/UpgradeVelo`
- Throughput test (Velo only and best performance)
- PrChecker (Velo only)
- Running on `lhcb-tdr-test` (`HEAD` of everything)
- Notifications when results available sent to mattermost private channel (please let us know if you're interested)
- More tests to come

Perf test for Gauss

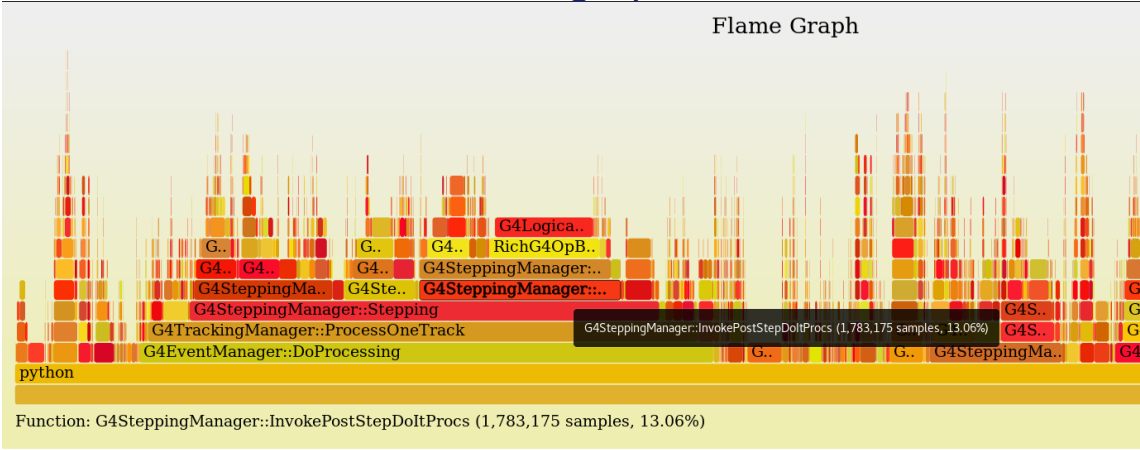
- Linux tool integrated in the kernel to (see ▸ tutorial by Hadrien):
  - measure CPU statistics
  - investigate where the program spends time
  - produce call graphs
- Test runs with command:

```
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
```

- Using lhcb-sim09-upgrade and $PRCONFIGOPTS/Gauss/PRTEST-2016-SIM-P8-10000000-100evts.py
- Scheduled to run weekly in LHCbPR
- Example of results:
  https://lblhcbpr.cern.ch/#/app/jobs/list?apps=4&executables=13&withNightly=false&checkSpecific=false&nightlyVersionNumber=10&jobs=47746&ssf=false


40.28% 0.62% python libG4tracking.so [.] G4SteppingManager::Stepping
--39.67%--G4SteppingManager::Stepping
|--17.94%--G4SteppingManager::InvokePostStepDoItProcs
|--17.40%--G4SteppingManager::InvokePSDIP
|--10.23%--RichG4OpBoundaryProcess::PostStepDoIt
--9.31%--G4LogicalBorderSurface::GetSurface@plt
|--3.59%--G4Transportation::PostStepDoIt
|--1.57%--G4TouchableHistory::G4TouchableHistory
--1.37%--G4NavigationHistory::G4NavigationHistory
--1.22%--G4EnhancedVecAllocator<G4NavigationLevel>::allocate@plt
--1.36%--G4Navigator::LocateGlobalPointAndSetup
|--0.76%--G4VEnergyLossProcess::PostStepDoIt
--0.56%--G4SeltzerBergerModel::SampleSecondaries
--0.59%--RichG4Scintillation::PostStepDoIt
--0.53%--G4SteppingManager::InvokePSDIP@plt
|--11.44%--G4SteppingManager::DefinePhysicalStepLength
|--3.89%--G4Transportation::AlongStepGetPhysicalInteractionLength
|--2.55%--G4Navigator::ComputeStep
--1.30%--G4VoxelNavigation::ComputeStep
--0.72%--G4PropagatorInField::ComputeStep
|--3.48%--G4VDiscreteProcess::PostStepGetPhysicalInteractionLength
|--1.70%--G4HadronicProcess::GetMeanFreePath
--1.45%--G4CrossSectionDataStore::GetCrossSection
--1.11%--G4CrossSectionDataStore::GetCrossSection
--0.72%--G4CrossSectionDataStore::GetIsoCrossSection
--0.95%--G4VProcess::ResetNumberOfInteractionLengthLeft
name of the function
time spent by the function and its children
time spent by the function itself
call graph

Flame graph


Flame Graph
G4Logica..
G..
G..
G4..
RichG4OpB..
G4..
G4..
G4..
G4SteppingManager:..
G4SteppingMa..
G4Ste..
G4SteppingManager::..
G4SteppingManager::Stepping
G4S..
G4TrackingManager::ProcessOneTrack
G4SteppingManager::InvokePostStepDoItProcs (1,783,175 samples, 13.06%)
G4S..
G4EventManager::DoProcessing
G..
G..
G4SteppingMa..
python
Function: G4SteppingManager::InvokePostStepDoItProcs (1,783,175 samples, 13.06%)

Igprof test for Gauss

- Performance and memory profiling tool, here just focus on:
  - memory that hasn't be freed
  - total memory allocated by any function
- Test runs with command:

```
igprof_env="lb-run -c {platform} --ext igprof --user-area={build}
{app_name}/{app_version}"
$igprof_env igprof -d -mp -z -o igout.mp.gz gaudirun.py {options}
$igprof_env igprof-analyse -v -g -r MEM_LIVE igout.mp.gz >
igout.mp.live.txt
$igprof_env igprof-analyse -v -g -r MEM_TOTAL igout.mp.gz >
igout.mp.total.txt
```

- Using lhcb-sim09-upgrade and $PRCONFIGOPTS/Gauss/PRTEST-2016-SIM-P8-10000000-100evts.py
- Scheduled to run weekly in LHCbPR
- Example of the results:
  https://lblhcbpr.cern.ch/#/app/jobs/list?apps=4&executables=14&withNightly=false&checkSpecific=false&nightlyVersionNumber=10&jobs=30204&ssf=false

memory leaked by the function and its children

```
Flat profile (cumulative >= 1%)

% total        Total        Calls  Function
  100.0  213'831'173      457'234  <spontaneous> [1]
   99.5  212'809'583      449'213  _start [2]
   99.5  212'809'583      449'213  __libc_start_main [3]
```

memory leaked by the function itself

```
-----------------------------------------------------------------
Flat profile (self >= 0.01%)

% total         Self        Calls  Function
  19.32   41'314'528          968  _ZN9RichHpdQE12setAnHpdQEenEiRKSt6vectorIdSaIdEES4_ [81]
  18.71   40'004'657          240  _ZnwmRKSt9nothrow_t [82]
   7.84   16'769'024          408  _ZN5clang4DeclnwEmRKNS_10ASTContextEjm [295]
   7.17   15'326'496            1  _ZNSt6vectorI5Vec4fSaIS0_EE14_M_fill_insertEN9__gnu_cxx17__normal_iteratorIPS0_S2_EEmRKS0_ [310]
   3.53    7'548'672        2'457  dictresize [398]
   2.81    6'002'703          232  _ZN4llvm20BumpPtrAllocatorImplINS_15MallocAllocatorELm4096ELm4096EE8AllocateEmm [478]
   2.67    5'712'424            1  _ZNSt6vectorIPN5clang4DeclESaIS2_EE6resizeEm [484]
   2.62    5'605'472       20'178  _ZNSt6vectorIdSaIdEE6insertEN9__gnu_cxx17__normal_iteratorIPKdS1_EERS4_ [491]
   2.26    4'835'656            1  _ZNSt6vectorIN5clang8QualTypeESaIS1_EE6resizeEm [512]
   1.92    4'106'288       71'608  _ZNSs4_Rep9_S_createEmmRKSaIcE [552]
   1.63    3'479'968        6'397  _ZN7EvtPhsp5cloneEv [577]
   1.45    3'104'768           53  _ZN5clang20TemplateArgumentList10CreateCopyERNS_10ASTContextEN4llvm8ArrayRefINS_16TemplateArgumentEEE [601]
   1.20    2'555'912          316  _ZN5clang11FileManager7getFileEN4llvm9StringRefEbb [547]
   1.07    2'278'520        3'963  _ZN4llvm15SmallVectorBase8grow_podEPvmm [685]
   1.01    2'159'016       17'618  _ZN14MCTruthManager11AddParticleERN5HepMC10FourVectorES2_S2_iiibiPN4LHCb10MCParticleEbb [645]
```

in bytes

call graph

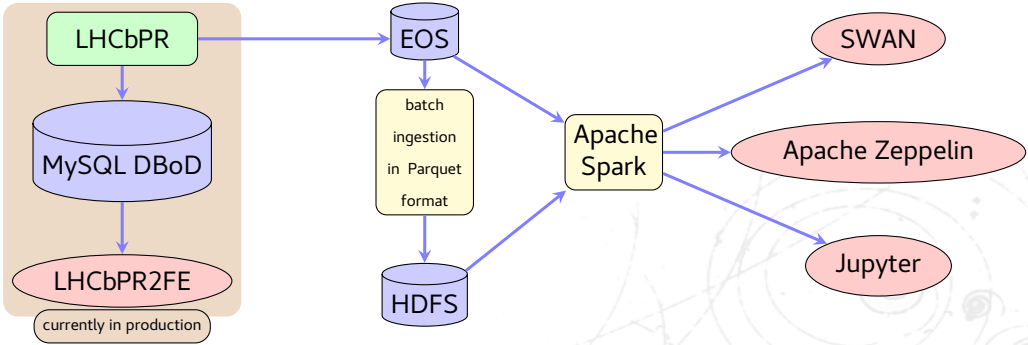
callers

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
          19.3  ...........   41'357'120 / 41'545'661        1'452 / 4'737          _ZN17RichHpdProperties23InitializeHpdPropertiesEv [79]
[80]      19.3   41'357'120       42'592 / 41'314'528        1'452                _ZN17RichHpdProperties21FillHpdQETablesAtInitEP16IDataProviderSvcP11IMessageSvc
          19.3  ...........   41'314'528 / 41'314'528          968 / 968            _ZN9RichHpdQE12setAnHpdQEenEiRKSt6vectorIdSaIdEES4_ [81]

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Rank    % total          Self        Self / Children          Calls / Total        Function
          19.3  ...........   41'314'528 / 41'357'120          968 / 1'452            _ZN17RichHpdProperties21FillHpdQETablesAtInitEP16IDataProviderSvcP11IMessageSvc [80
[81]      19.3   41'314'528   41'314'528 / 0                   968                  _ZN9RichHpdQE12setAnHpdQEenEiRKSt6vectorIdSaIdEES4_

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

callees

Big data tools for LHCbPR

- Data from tests pushed to Hadoop Distributed File System
- Possibility to use Spark connector in SWAN notebooks to read those data
- **Interactive** exploration, **collaboration** on tests reports
- Needs to belong to the `ai-hadoop-users` e-group



SWAN notebooks for the analysis of tests results

Query

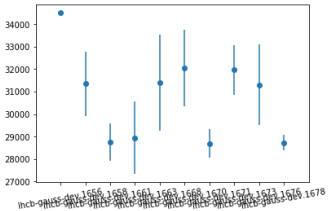
In [54]:

```
app="Gauss"
opt="PRTEST-2016-SIM-P8-10000000-100evts"
plat="x86_64-slc6-gcc49-opt"
slot="lhcb-sim09-upgrade"
host="lblhcbpr1.cern.ch"
attrname="EVENT_LOOP"
lastn=10

from pyspark.sql.functions import rank, col, avg, stddev_pop

results=flattened.select("app_version","JobAttributes.data")\
                 .filter("app_name = '"+app+"'")\
                 .filter("opt_name = '"+opt+"'")\
                 .filter("JobAttributes.name='"+attrname+"'")\
                 .filter("HOST.hostname='"+host+"'")\
                 .filter("app_version like '"+slot+"%'")\
                 .filter("CMTCONFIG.platform='"+plat+"'")\
                 .orderBy(flattened["app_version"].desc())\
                 .groupBy("app_version")\
                 .agg(avg(col("data")))\
                 .limit(int(lastn))
results_unc=flattened.select("app_version","JobAttributes.data")\
                 .filter("app_name = '"+app+"'")\
                 .filter("opt_name = '"+opt+"'")\
                 .filter("JobAttributes.name='"+attrname+"'")\
                 .filter("HOST.hostname='"+host+"'")\
                 .filter("app_version like '"+slot+"%'")\
                 .filter("CMTCONFIG.platform='"+plat+"'")\
                 .orderBy(["app_version"], ascending=[0])\
                 .groupBy("app_version")\
                 .agg(stddev_pop(col("data")))\
                 .limit(int(lastn))
```

```
import matplotlib.pyplot as plt
x_val = [x[0] for x in reversed(results.collect())]
y_val = [float(x[1]) for x in reversed(results.collect())]
yerr = [float(x[1]) for x in reversed(results_unc.collect())]
print(x_val)
print(y_val)
print(yerr)
plt.xticks(list(range(len(x_val))), x_val)
plt.xticks(rotation=10)
plt.xlim([-1,len(x_val)])
plt.errorbar(range(len(y_val)), y_val, yerr=yerr, fmt='o')
plt.show()
```

▸ Apache Spark: 2 EXECUTORS 4 CORES Jobs: 11 COMPLETED

```
[u'lhcb-gauss-dev.1656', u'lhcb-gauss-dev.1658', u'lhcb-gauss-dev.1661', u'l
-gauss-dev.1676', u'lhcb-gauss-dev.1678']
[34509.898, 31345.9206, 28751.5598, 28947.599599999998, 31394.860399999998,
[0.0, 1447.5902695502755, 838.1198149096348, 1604.3968748786074, 2133.645968
```



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/`