

Improvements to the LHCb software performance testing infrastructure using message queues and big data technologies

Ben Couturier¹, Maciej Szymański^{1,2} on behalf of the LHCb collaboration

¹CERN, ²University of Chinese Academy of Sciences

Objectives

- Recent developments to the LHCb Performance and Regression (LHCbPR) framework
- Integration with LHCb Nightly Builds using message queues
- Big data technologies for storing and analysing the output of tests

Software in HEP

- Software is the **essential component** of experiments in HEP
- Upgraded on a **short timescales** compared with e.g. detectors
- Therefore, **flexible**, but susceptible to issues and bugs
- Need of **systematic regression and performance tests**

LHCbPR

- Performance baseline in **controlled conditions**
- Inspect any changes** due to e.g. merge requests, new externals, etc.
- Compare results** across various compilers and architectures
- Not only **resource consumption**, but also **physics performance**
- Benchmarking code especially crucial in the **LHC upgrade era!**
See T5, S.Ponce, Tue 14:00

Design

- Microservice architecture**
- Software **containers**
- Flexibility** in running the software
- Friendly reporting**: comparing histograms, trend analysis

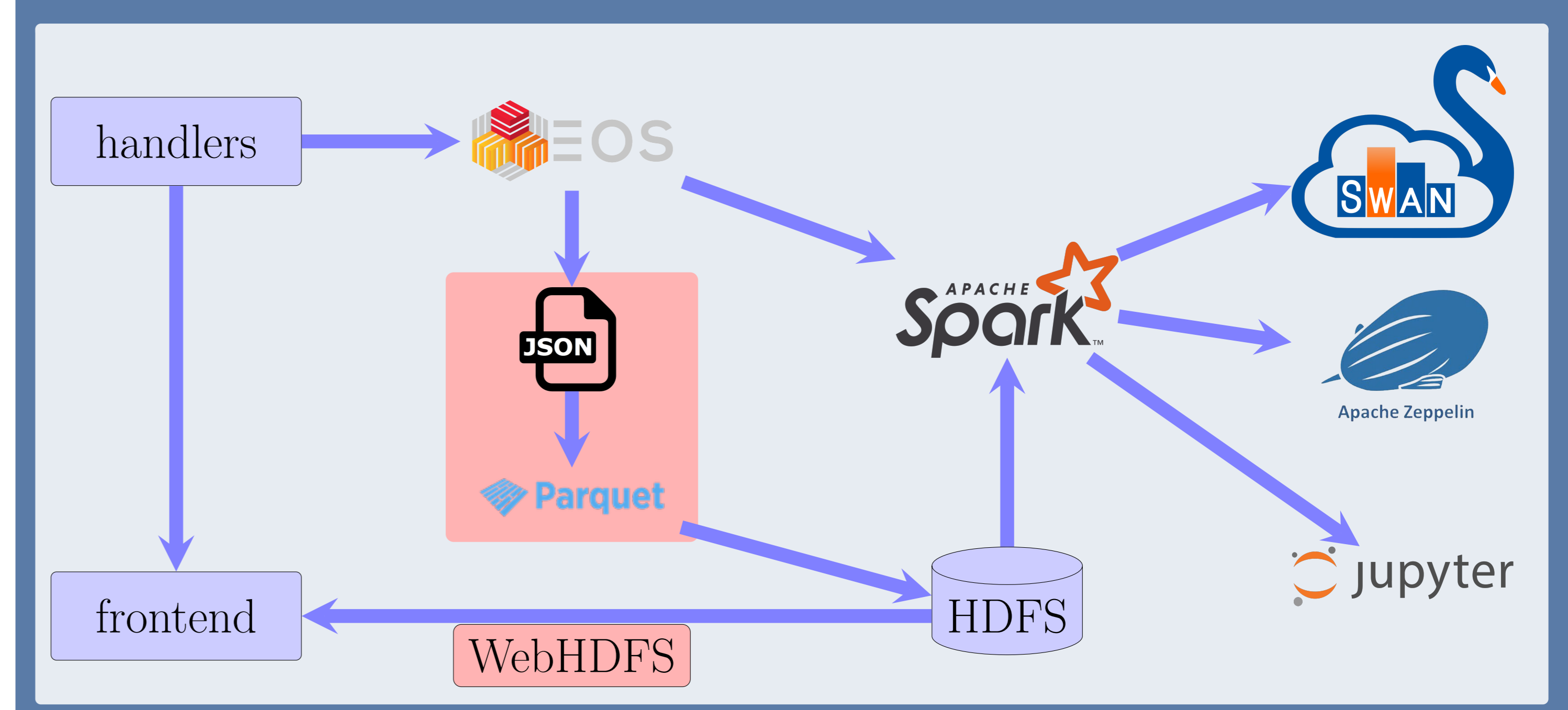
Why big data tools?

- Increasing **data volume**
- Interactive** exploration
- Easier and faster** access to data
- Flexible** reports
- Collaboration, reproducibility** owing to integrated notebooks

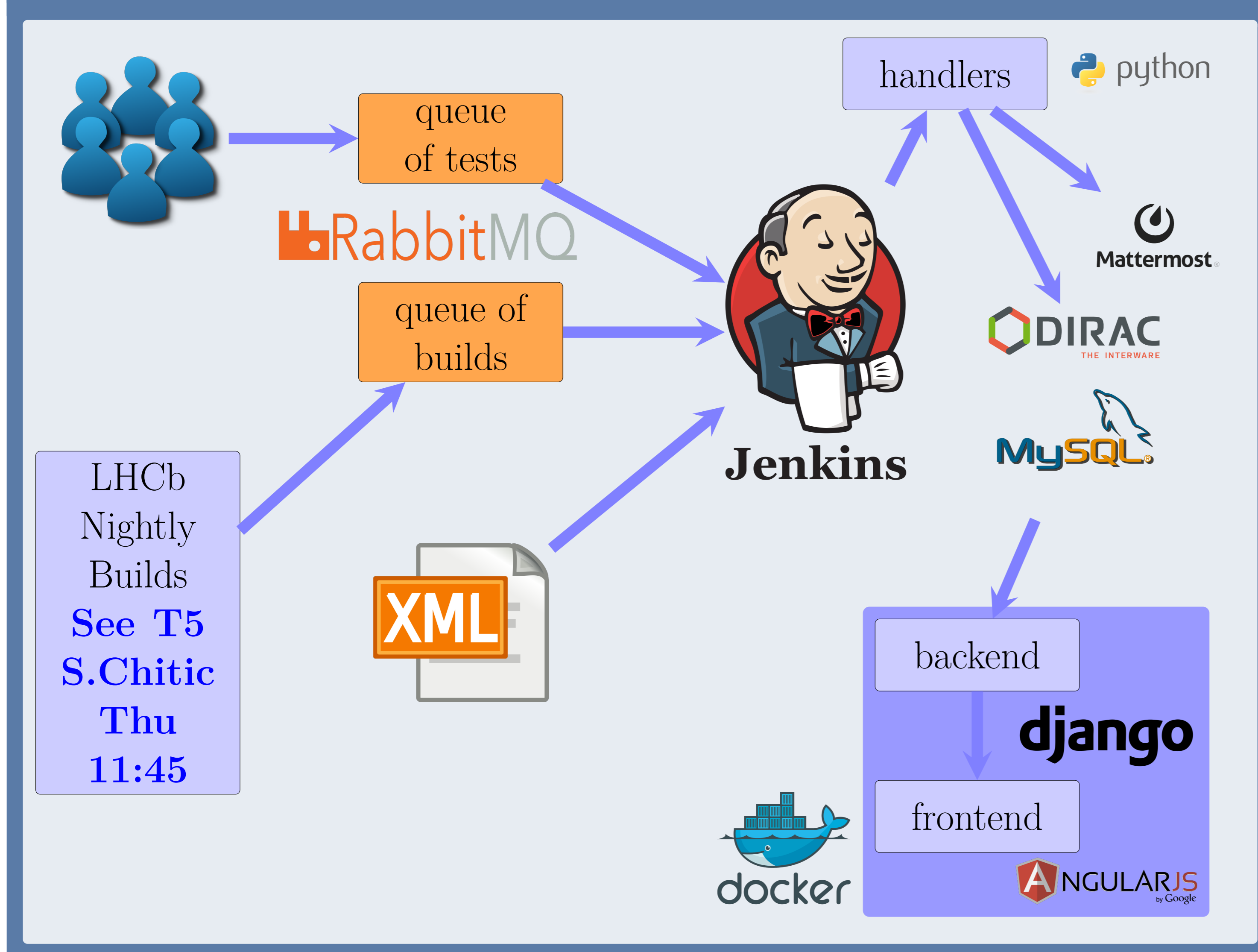
Technologies of choice

- Apache **Spark** for data processing engine
- Apache **Parquet** data format
- Apache **Zeppelin** and **SWAN**: notebooks with PySpark interpreter

Enhancing LHCbPR framework with Hadoop



Integration with messaging system



Running as a service

- 5 LHCb applications (e.g. for reconstruction, simulation, triggering), 67 option files
- ~50 tests daily
- Tests run from ~minutes up to ~10h
- Tens of MB daily
- ~8 GB collected up to now (~1.5 years, zip files with JSON and ROOT files)
- LHCbPR** has shown to be **versatile framework** useful for the Collaboration

Infrastructure

- Tests **sensitive to timing** running on dedicated machines ensuring that the other load is minimised
- Results of the tests **parsed by the handlers** to save metrics
- Zip file sent to the DB through **Dirac Storage Element**
- Web front-end** `l1lhcpr.cern.ch` with i.e. generic ROOT files viewer, trend analysis, see T5, R.Currie, Thu 12:00

Prototype

- Based on CERN IT Hadoop service
- Porting analysis modules** to notebooks using both Scala and PySpark
- Widgets** for interactive analysis
- Using **spark-root** for reading ROOT files

Data ingestion

- Automated daily copies (cron job)
- JSON files merged once a day and copied to HDFS (`hdfs dfs -put`)
- Conversion into Parquet format (compression ratio wrt JSON: ~ 16)
- Significant **speed-up when data partitioned** properly

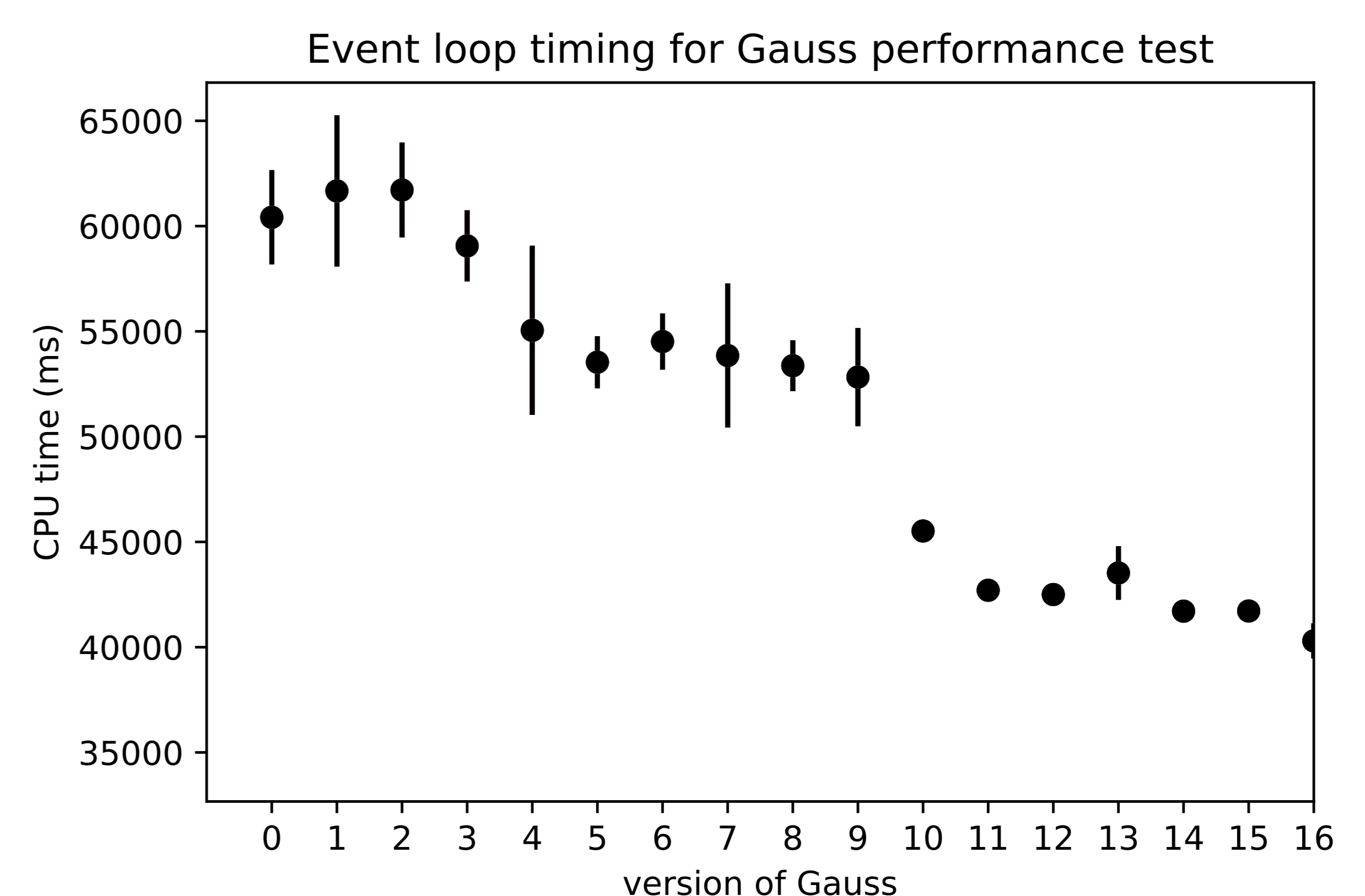


Figure 1: Time spent in the event loop by the benchmark. Labels on the x-axis correspond to different versions of Gauss. Decrease of the time for versions 4 and 10 is due to changes in the RICH code specifically to speed it up which were introduced in corresponding software versions. See T2, D.Popov, Thu 12:15