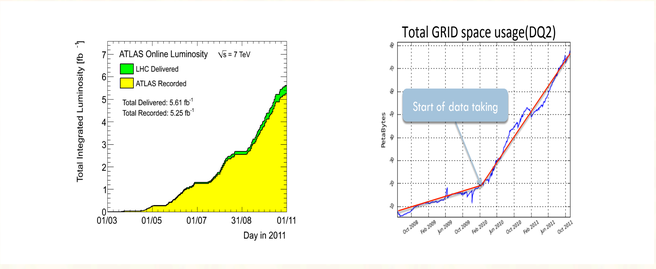
Monitoring of computing resource utilization of the ATLAS experiment

Ilija Vukotic, David Rousseau, Gancho Dimitrov, Osman Aidel, RD Schaffer, Solveig Albrand

Ilija Vukotic on behalf of the ATLAS collaboration, Laboratoire de l'Accelerateur Lineaire, Bat 200, BP 34, 91898 Orsay, France
E-mail: ivukotic@cern.ch

Introduction

Due to the good performance of the LHC accelerator, the ATLAS experiment has seen higher than anticipated levels for both the event rate and the average number of interactions per bunch crossing.
The current and future usage of CPU, memory and disk resources has to be monitored, understood and acted upon.
This requires data collection at a fairly fine level of granularity: the performance of each object written and each algorithm run, as well as a dozen per-job variables, are gathered for the different processing steps of Monte Carlo generation and simulation and the reconstruction of both data and Monte Carlo. We present a system to collect and visualize the data from both the online Tier-0 system and distributed grid production jobs.

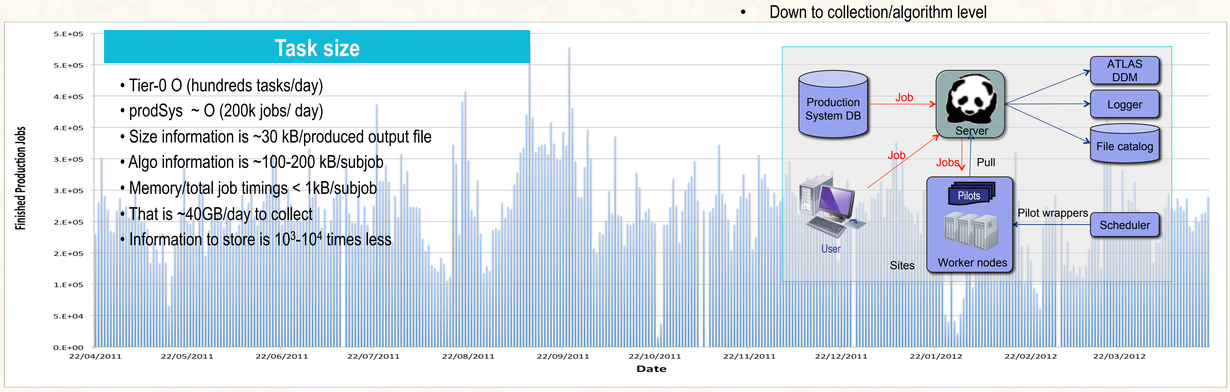


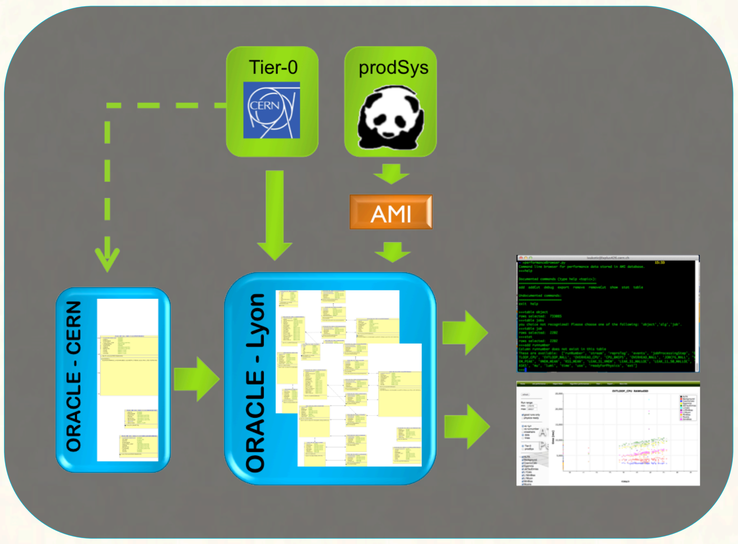
Requirements

- **Fast feedback** is essential if we want to reduce unnecessary resource usage due to misconfiguration of a sub-detector, trigger, job or even algorithm
- Reliable, **full information** is needed as important changes can happen in the middle of the run. on our resource usage
  - Disk size
  - CPU/real time usage
  - Memory usage
- System must be **robust, easy to support**
- It must be **easy to extract the info** at each granularity level.
- System must be **transparent** in a sense that no job should ever fail due to problem in monitoring. The monitoring can introduce only very **small overhead**.
- Way to **throttle percentage of jobs monitored**. This is specially important for MC jobs where there we need just to collect statistically significant sample
- ALL the officially produced data
  - both T0 and prodSys(grid)
  - each reprocessing
  - ALL the streams
  - ALL the formats
  - ALL the runs
  - Most of the events
  - Down to collection/algorithm level

Task size

- Tier-0 O (hundreds tasks/day)
- prodSys ~ O (200k jobs/ day)
- Size information is ~30 kB/produced output file
- Algo information is ~100-200 kB/subjob
- Memory/total job timings < 1kB/subjob
- That is ~40GB/day to collect
- Information to store is 10^3-10^4 times less





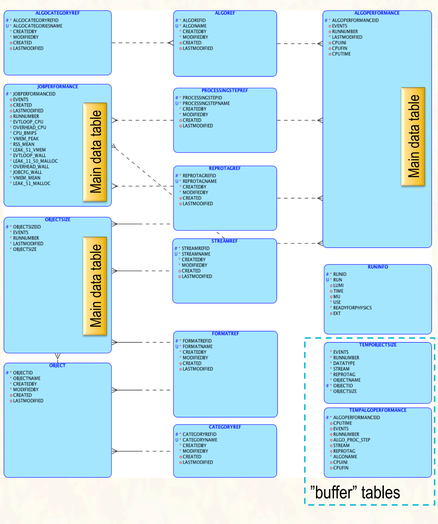
Data collection

Sources:
- Object sizes from ROOT/POOL based script
- Algo/Tools/Service timings – Gaudi (ChronoSvc) or inhouse developed PerfMonSvc.
- Per task information: <μ>, luminosity, data quality flags, etc. mashed up from AMI and other databases.

Major problem: separation of official and user jobs.

Data transport

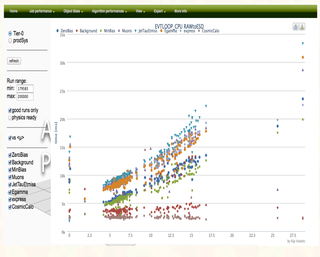
A lot of options: CHIRP, JEM, etc. For Tier-0 we choose direct Oracle access. prodSys jobs upload through AMI frontend. Dedicated optimization of AMI procedures: fast authentication/authorization using VOMS. Extreme robustness and failover required.



Data storage

A special design and detailed optimization of the database schemes. The DB summarizes the data received per: the input dataset, transform executed and set of processing variables. This results in reduction of information stored by factor usually more than thousand. Oracle database schemes for both Tier-0 and prodSys data are identical. The most important database design decision was to have "buffer" tables for both object size and algorithm timing data. These tables have no indices and no constrains of any kind. This makes bulk inserts into them extremely fast, and also evens the load on the DB servers. Once a week tables are resized for performance reasons. Information collected is added to appropriate rows of the "real" tables using stored procedures executed every 2 minutes. Stored procedures are optimized by changing row based with table based sql commands. The two largest "real" tables are optimized by partitioning them according to run number.

Data Mining



Can be done using command line tools, AMI, and web interface.
http://ivukotic.web.cern.ch/ivukotic/performance/index.asp
(ASP, jQuery, highcharts)

Results

In the year 2011, the system collected information on more than 98% of all events. Time to upload all the information from a single job ranges from 150-300 ms. To test potential of the system to accept the data from the production system, we stressed the system by uploading dummy data with twice the expected upload rate (equivalent of 500 000 concurrent production jobs). Even that was still giving less than 10% load on the DB instance dedicated to the system.

Conclusion

The system described proves that even at the scale of ATLAS experiment with hundreds of thousands of concurrent, world-wide distributed jobs, a relatively simple but highly optimized system with an Oracle DB back-end, can do the job without resorting to much more complicated solutions (both to set up and to support).