Draft version 1.0

ATLAS NOTE

April 6, 2012

Monitoring of computing resource utilization of the ATLAS experiment

David Rousseau[a], Gancho Dimitrov[b], Ilija Vukotic[a], Osman Aidel[c], RD Schaffer[a], Solveig Albrand[d]

[a]*LAL, Univ. Paris-Sud and CNRS/IN2P3, Orsay, France*
[b]*CERN, Geneva, Switzerland*
[c]*Domaine scientifique de la Doua, Centre de Calcul CNRS/IN2P3, Villeurbanne Cedex, France*
[d]*Laboratoire de Physique Subatomique et de Cosmologie, Universit e Joseph Fourier and CNRS/IN2P3 and Institut National Polytechnique de Grenoble*

Abstract

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. In order to respond to these changing requirements, 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. Around 40 GB of performance data are expected from up to 200k jobs per day, thus making performance optimization of the underlying Oracle database of utmost importance.

1 Introduction

The main goals of this system is to provide the following information:

size of all the objects that ATLAS stores in all officially produced data files. timing of all the algorithms, tools and services used during official data production, simulation, reconstruction. CPU time, wall clock time, memory leakage and other per job information

This information will be used:

by performance experts to identify places where optimization will be most beneficial by management to determine current resource needs and to extrapolate for future running parameters by reco shifters to continuously monitor changes in resource consumption example1: find all the persistent objects not written out in last 1 year and remove them from the code. example2: find all the unneeded persistent objects written out. example3: find all the algorithms, tools or services not needed in a particular transformation step. example4: estimate resources needed for a particular MC/real data production/reconstruction.

It is important to notice that in some cases even significantly less than 100% of the events will be taken into account. Still this should not matter for most of applications.

2 Sources of information

While this can be changed for now we settled at:

stored object size is obtained from PoolFile.py root based tool (also used by checkFile.py) that opens root file and returns names, sizes and number of entries of all the collections stored. algorithm timings are currently obtained from PerfMonSD. (Data collected before 1st April were obtained through ChronoSvc). per job information is obtained from PerfMonSD. This in turn obtains this information from operating system.

Code collecting the information is situated in the doPostRunActions function of the Tools/PyJobTransformsCore/trf.py

3 Data transport

Tier-0 collected data are sent directly to Oracle DB in Lyon. Authentication is done via environment variable TZAMIPW. In case that the default db is unreachable Oracle DB in CERN is used to temporarily store the data, from where data are moved automatically to Lyon. prodSys (grid) jobs data are sent using special AMI commands. Authentication is done via VOMS. Only jobs having '/atlas/Role=production' in output of voms-proxy-info -fqan will try to send data.

Currently there is no backup solution in case of problems with AMI. In both cases it is guaranteed that if upload fails for whatever reason it will fail quietly and jobs will finish normally. There is a window of 60 seconds in which all of the data collection and delivery has to be done. All the code concerning data transport is in Tools/PyJobTransforms.

4 Data storage

Database scheme may be found here. For now there is no documentation describing all the stored procedures that are running on the db and which summarize the data received. In order to ease certain common tasks a table with run info is included in the data base. It contains:

mu - average mu - from the plots like these lumi - luminosity time - duration of the run use - flag saying if this run is ok to use. It is manually set. It can be used to exclude runs where information is messed up. ReadyForPhysics - flag obtained from AMI db. All the runs having this flag set to True are distributed.

This table is updated by the script perfStatusOfAmi.py run by cron job every day at 13:00.

5 Data mining

How to use it? Classify objects/algorithms Independently what type of visualization you will use, and unless you are interested in only job performance data, you should always first make sure objects/algorithms are all properly classified. This is done through AMI interface. Path to it:

Start from the AMI home page (AMI home). In the menu find: Applications-¿Atlas-¿AMI admin Now menu has changed and you have in it also Database option Click on it and you have a large list of the databases. You need the one named:

`COLL_SIZES_01`

Click on it (not on "+" sign) and than bookmark this page as you'll need it. This bookmark is not the one from your browser but the one from AMI (you'll see it in the menu) Now click on

`ATLAS_AMI_COLL_SIZES_01_LAL`

in 'database name' will give you list of all the relevant tables. For the details on the table schema click on table name. To look at the data in the table click on the link (Browse). For the classification purposes two table are important ones AlgoRef and object.

Initial object classification was done by hand. Objects are separately classified for all the data formats (AOD,ESD,DPD,...) Algorithms are classified by a script Tools/PerformanceReports/updateAlgoCategories.py. The script is kept up to date by ThomasKittelmann. Currently the only persons having necessary rights to make changes in classification are DavidRousseau, IlijaVukotic, ThomasKittelmann and RD Schaffer.

NEVER CHANGE INFO IN OTHER TABLES UNLESS YOU KNOW EXACTLY WHAT ARE YOU DOING - for example removing category or stream name will delete all the data ever classified in this category/stream.

Place your content here. Policy [1]

Place your content here Appendix A

Place your content here Appendix B

Place your content here Appendix C

6 Performance

here on what percentage of info was collected. what is time to upload what is cpu load produced

7 Conclusion

Do we find this system to work ok for this kind of task and this data size. How much it could expand.

References

[1] *ATLAS Publication Policy*, Available from the atlas publication committee web page:
Http://atlas.web.cern.ch/atlas/groups/general/scinotes/scinotes.html.

Appendices

A Database accounts

accounts are : ... servers are: ... which ones are used: ...

B Database schema

schema looks like this: figure

C Database optimization

all of the partitionings, row movements, re-indexings