PROCEEDINGS OF SCIENCE

Optimizing the usage of multi-Petabyte storage resources for LHC experiments

Mr. BARREIRO MEGINO, Fernando
CERN
E-mail: fernando.harald.barreiro.megino@cern.ch

Mr. CINQUILLI, Mattia
CERN
E-mail: Mattia.Cinquilli@cern.ch

Dr. GIORDANO, Domenico
CERN
E-mail: Domenico.Giordano@cern.ch

Dr. GIRONE, Maria
CERN
E-mail: Maria.Girone@cern.ch

Dr. KARAVAKIS, Edward
CERN
E-mail: Edward.Karavakis@cern.ch

Dr. MAGINI, Nicolo
CERN
E-mail: Nicolo.Magini@cern.ch

Mrs. MANCINELLI, Valentina
CERN
E-mail: valentina.mancinelli@cern.ch

Dr. SPIGA, Daniele*
CERN
E-mail: Daniele.Spiga@cern.ch

In the last two years of Large Hadron Collider (LHC) [1] operation, the experiments have made a considerable usage of Grid resources for the data storage and offline analysis. To achieve the successful exploitation of these resources a significant operational human effort has been put in place and it is the moment to improve the usage of the available infrastructure. In this respect, the Compact Muon Solenoid (CMS) [2] Popularity project aims to track the experiment's data access patterns (frequency of data access, access protocols, users, sites and CPU), providing the base for the automation of data cleaning and data placement activity on Grid sites. As well, the popularity-based Site Cleaning Agent (Victor) has been developed to monitor the evolution in time of the used and pledged space and remove unused data replicas at full Tier-2s.

PoS(EGICF12-EMITC2)107

PROCEEDINGS OF SCIENCE

EGI Community Forum 2012 / EMI Second Technical Conference,
26-30 March, 2012
Munich, Germany

*Speaker.

1. Introduction

The CMS experiment at the LHC has developed its computing model [3, 4] based on a distributed system of computing resources and services relying on Grid technologies. The resources are made available to CMS through the Worldwide LHC Computing Grid Project (WLCG) [5]. The CMS Computing Model uses a hierarchical structure of the collaboration's various computing centers, called Tiers, following the implementation proposed by the Models of Networked Analysis at Regional Centres (MONARC) project [6]. The hierarchy is composed by the following levels:

- a single Tier-0 centre at the European Organization for Nuclear Research (CERN) for prompt data reconstruction and custodial responsibility;
- 7 Tier-1 centres having custodial responsibility for raw and reconstructed data. They also provide services for reprocessing, calibration, skimming and other data intensive analysis tasks;
- 50 Tier-2 centres where a fraction of the reconstructed data and high level analysis objects are replicated for user analysis; they also provide Monte Carlo simulation facilities.

A CERN Analysis Facility (CAF) is also used for low latency activities requiring access to raw data: it combines flexible CPU resources with rapid access to the entire CMS data collection for fast analysis [7].

Grid analysis in CMS is data driven and analyis jobs are directed to the Tier-2 sites where the needed data is available. A prerequisite is that data is already distributed to the sites. A data transfer system, PhEDEx [8], is responsible for moving data, both real and simulated, between sites reliably.

The CMS experiment makes a considerable usage of Grid resources for the data storage and offline analysis. As shown in Figure 1 and Figure 2 an average of 250 daily users submit 200k jobs per day, reaching peaks of 500k jobs that access distributed data.

Given the scale of the CMS Grid infrastructure, it is a complex problem to control and optimize the usage of the storage. In this context the CMS Popularity Service has been developed to track over time the usage of the collaboration's data. The aim of the service, originally inspired by the ATLAS [9] experience [10], is to monitor which data is actually being used measuring its popularity based on various metrics. The system is also designed to serve data to external services as currently it is happening with Victor the Site Cleaning Agent which has the role to suggest data that can be safely removed. In this paper we describe the CMS Popularity System providing details about its architecture, the system performance and the monitoring perspective. We give an insight of the workflow of the popularity-based Site Cleaning Agent Victor.

2. Popularity Service

Aiming both to avoid duplication of functionality with the already existing services and mainly to implement a sustainable system we have defined a modular architecture for the Popularity Service. This approach is extremely important to allow to gather file usage from several data services, either experiment specific or not.

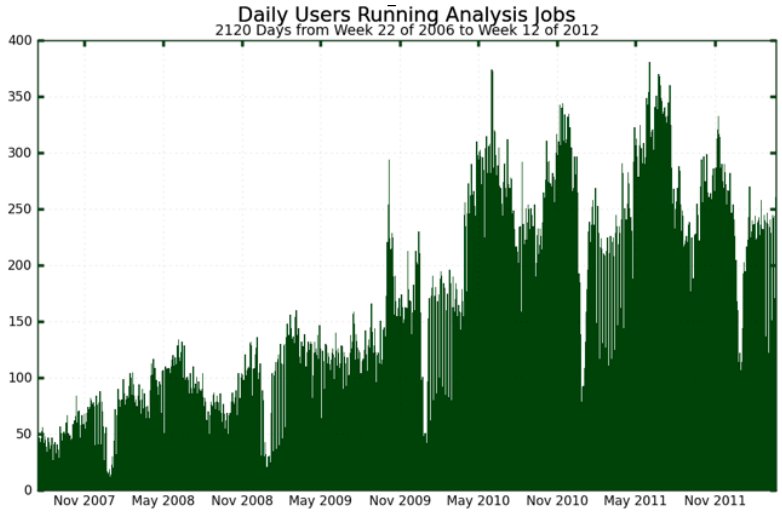


Figure 1: Number of daily users running analysis jobs in CMS during the past four years.

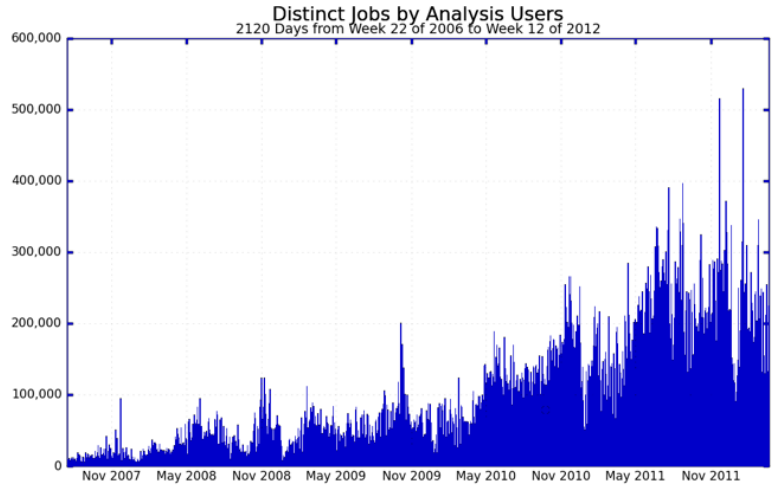


Figure 2: Number of jobs submitted daily to the Grid via CMS Remote Analysis Builder (CRAB) [11] during the past four years.

The CMS Computing Model foresees that the end user runs its data analysis algorithms on the Grid. Particularly the Tier-2 sites have the role to support the end user activities, both providing CPUs and storage where data of interest are placed. The activity monitored by the CMS Popularity System refers to the physics analysis carried out on the data produced by the experiment and spread around the Tier-2s. The CMS users access data for their analysis mostly through the user-friendly interface CRAB to handle data analysis in a local or distributed environment, hiding the complexity of interactions with the Grid and other CMS services.

CRAB has been enabled to report to the CERN Dashboard service [12] a summary information for all job activities (submission, status, failures). This information includes also the file-based

statistics used by the CMS Popularity Service that remain only temporarily archived in the Dashboard, just for the time needed to transfer them into the Popularity System. The Popularity Service has been completely decoupled from Dashboard and uses this latter as one of the possible data sources of the file based information. This choice is due to the different goals of the Dashboard monitoring, more focused on the realtime monitoring of the jobs respect to the aggregation and data mining of the patterns of usage of the data accessed.

In fact, the system CRAB-Dashboard is not the only possible data source for the Popularity System. Any system providing patterns of access and usage of the files on a storage system can be used as an external source. We have recently developed the plugin to collect this information from a storage data service accessed by means of the XRootD [14] software framework. XRootD is a fully generic suite for fast, low-latency and scalable data access, which can serve natively any kind of data, organized as a hierarchical filesystem-like namespace, based on the concept of directory. Nowadays several Tier-2 sites implement the remote access of their stored files through the XRootD protocol. In such a way an alternative solution to the data access is provided when the local access of a same file in another site fails: the effect is an improvment of the job success rate. The information of the remote file accessed as fallback solution is propagated also from CRAB to the Popularity Service, and allows to monitor such behaviour.

In addition the CMS CAF at CERN is equipped with a storage data service (EOS [13]) based on XRootD. It provides monitor information about the storage activity, such as file open and close, read bytes, username. This information is exposed through UDP packets sent from the XRootD servers and collected by dedicated plugins into the Popularity Service.

2.1 Design and architecture description

As shown in Figure 3, the CMS Popularity System is composed by the following components:

- Popularity plugins to interface the different data sources;
- Popularity database;
- Popularity Web Frontend;
- Popularity API for external systems (e.g. Site Cleaning Agent).

By design the Popularity System has been conceived completely decoupled from the data sources; hence it gathers information from them by means of dedicated plugins. This approach guarantees flexibility and the easy extension of the monitoring system to different providers of information (such as CRAB-Dashboard and XRootD).

Each plugin interrogates a data source, using dedicated APIs provided by the data service. For the CRAB-Dashboard case, a Dashboard web API exposes the data cached in the Dashboard database in the last day of operation. For the XRootD case, it is through an ActiveMQ [15] API that the data are retrieved from a message queue. The usage of a message queue allows to decouple the collection of the UDP packets from the Popularity plugin that can consume asynchronously the collected information.

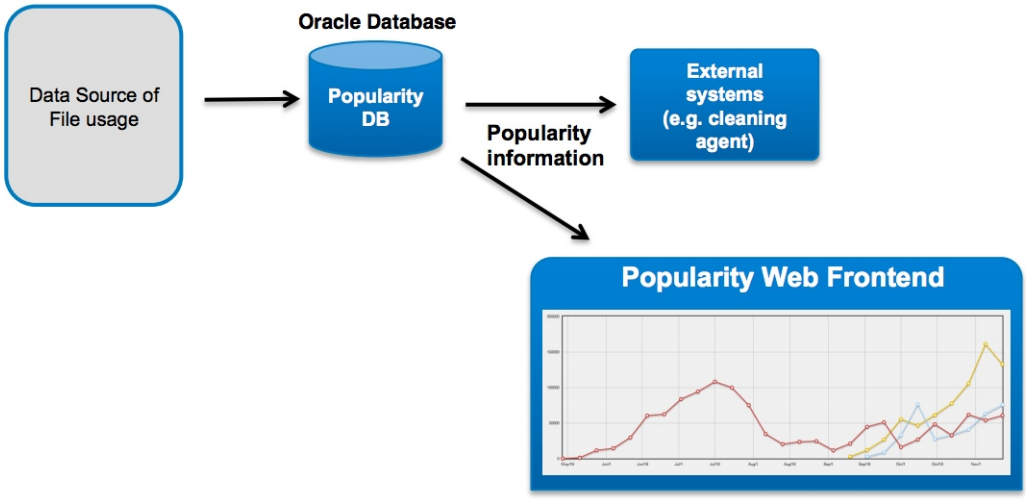


Figure 3: Popularity data flow. Data source can be either gathered from CRAB and Dashboard service or from the XRootD UDP packets collected and published in the Messaging System for Grid.

The plugins run every day populating a table of the Popularity database containing raw data, generating about 360k records per day. Each record includes a variable number of attributes, depending on the data source. Some of these attributes are common to any data source, such as the timestamps identifying the start and end of an access, the file name accessed and the site name from which it was accessed as well as the user name of who accessed the file. For the CRAB-based record other attributes are collected and refer to the job execution, such as exit codes, luminosity sections analyzed and local/remote access.

Considering the amount of data collected a database system is necessary to store, structure and extract the needed information in short processing time. For this purpose we have chosen to use a relational database implemented through Oracle with an organized set of materialized views to optimize the query operations. Materialized views are used to extract summaries at different levels of granularity and correlate them with other significant attributes for the popularity analysis (site, user, number of users, access time in CPU hours). The materialized views have a hierarchical structure with a first level that performs aggregations on the raw table and a second level of views that use the first level ones to furthermore refine the information on different details. As both the raw table population and the materialized views update are executed on a daily base, the information contained in the database is always up to date to the previous day.

The user interface of the service is implemented as a web service. It provides a graphic visualization, with interactive tables and plots, of the information through a web site, so that the application is easy to access and to use with every browser, without needing toinstall any software. The web service also provides a web API that allows to retrieve the popularity information in JSON formatted data that can be further used as a data source by other services (e.g. the Site Cleaning Agent).

The web service is developed as a Django application. Django [16] is an open source web application framework, a tool to build web application fast and easily. In order to improve the

recurrent retrieving of large data volume, a memory-based data caching system - memcached [17] - is used. Django allowed us to adopt this system in an easy and fast way thanks to a robust cache framework that provides backends for various cache services.

The CMS Popularity System has been structured to follow a modular approach: the services developed for the application share the same common core and the same structure resulting in a homogenous interface and a solid base that facilitates code maintenance, optimization, and the development of others modules. The core contains python modules that provide functionality like user input validation, http comunication, database connection. A common templating model and javascript library for the graphic data representation is applied in order to reuse tools without code replication.

2.2 Security

The security aspect has been highly considered during the development of the CMS Popularity System. The Popularity web service is deployed behind the CMS Web Redirector [19]. A web redirector is a reverse proxy used in front of other web services in order to mitigate potential threats coming from the underlying network, managed and unmanaged clients and hosts, potential untrustworthy users. Thanks to this service, the CMS Popularity System is equiped with a Web Application Firewall (WAF) through the Apache ModSecurity [20] module and SSL based authentication. Regarding the code, it has been implemented following the recommendations from the CERN security rules [21]. In order to improve the code maintainability, the input validation logic has been implementd in a python library and used in the python scripts. The role of the library is to take care of the retrieval of the needed information from the database and generate the output for the web interface and the API. Furthermore all the user input forms in the web interface are made to provide and accept only valid choice values.

2.3 Performance of the system

The CMS Popularity Service is steadily collecting data since May 2011. At the time of writing the majority of the collected data have been taken through the CRAB-based source plugin. It has been the first plugin implemented while the XRootD one went in production recently. That said, in this section we evaluate the performace referring to the data coming from the CRAB-based plugin. It is important to observe that the used plugin doesn't affect the performance of the whole system. The amount of data uploaded every day concerns the activity of about 100,000 jobs accessing in total approximately 360,000 files per day. The upload procedure for the raw table takes on average 30 minutes per day while the procedure to update all the materialized views takes on average 15 minutes per day. The size of the stored records is about 120 GB for the raw table and about 2.5 GB for the materialized views. The update runs in parallel on all the first level views, a process that takes most of the measured time, and aftewards the second level views get updated accordingly in a matter of few seconds.

The performance of the web interface highly benefits from the use of the materialized views to extract information from the database. Combining the low query processing time with the implemented cache system, the loading of the web pages results really performing. The average size of a http response from the web service is of about 50 KB but reaches peaks of 3 MB for some

resources. What affects more the loading time of the web pages is the time needed by python modules to extract and format data, followed by the javascript execution for the tables and plots creation. The average waiting time for the data response is of about 1 second for the medium size result and goes up to a maximum of about 15 seconds for the biggest results. Thanks to the caching system of the Django application combined with the caching of the results performed by the Oracle database system, once the results have been cached the waiting time decrease by a factor of two for the average size response and goes down to about 1 second with response up to 2 MB of size.

So far the database size and the data updating procedures times are excellent for the amount of data managed with no problems or bottlenecks observed in both workows. The system appears performant and enough exible to cope with a realistic increase of the number of records daily inserted.

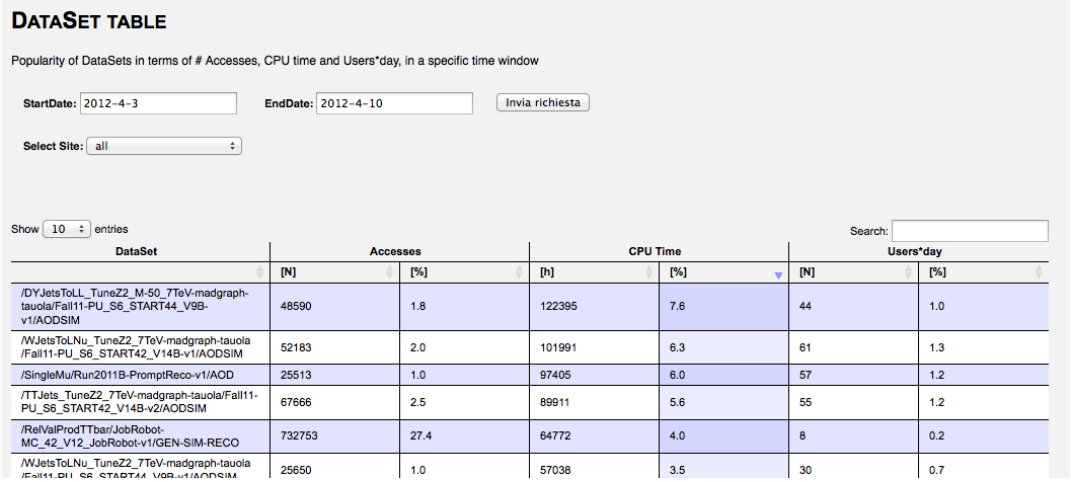
2.4 Popularity data exposure

In the CMS Computing Model an event consists of the detector electronic footprint of the particles produced in the P-P (or Pb-Pb) collision at the crossing of the bunches. Starting from the raw data produced by the online system, successive degrees of processing (event reconstruction) refine this data, and create higher-level physics objects. Multiple files representing a coherent sample are collected in a set of data called dataset. The datasets are defined primarily by processing history and event selection criteria. In turn, event information from each step of the simulation and reconstruction chains is logically grouped into what we call a data tier. The most important data tiers from a physicist's point of view are RECOnstructed data (RECO) (all reconstructed objects and hits) and Analysis Object Data (AOD) (a smaller subset of RECO which is needed by analysis).

The Popularity Service provides usage statistics about datasets and datatiers accessed by the users. The information is presented as:

- **tables**: a complete set of statistics and aggregations for a specific time window;
- **charts**: historical view of the values by giving a visual representation of the metrics evolution over time;
- **JSON formatted API [22]**: other than serve external system, the API allows users who may want to run their own specific analysis, to extract and furthermore manipulate the data.

The collected statistics are highly customisable in terms of metrics reported, period of time covered, geographical location of data. We use DataTables [23], a plug-in for the jQuery Javascript library for the creation of dynamic tables. Therefore all tables are interactive allowing the user to order the result by different colums and to filter the content of the tables by fields. For the plots we use Highcharts [24], a charting library written in pure JavaScript that offers intuitive, interactive charts for web applications. Charts shows the popularity metrics along time for datasets and datatiers. The metrics used to value the popularity of an element can be selected between CPU hours, number of accesses or number of users and refers to a specific time window and time aggregation that can be configured through the user interface. The plots show both the selected time windows and the complete history of the metric value in the long run and not only limited to the present situation. It is also possible to monitor the usage of specific datasets selected by the user from a base of all the datasets known by the database.



DATASET TABLE

Popularity of DataSets in terms of # Accesses, CPU time and Users*day, in a specific time window

StartDate: 2012-4-3 EndDate: 2012-4-10 Invia richiesta

Select Site: all

Show 10 entries Search:

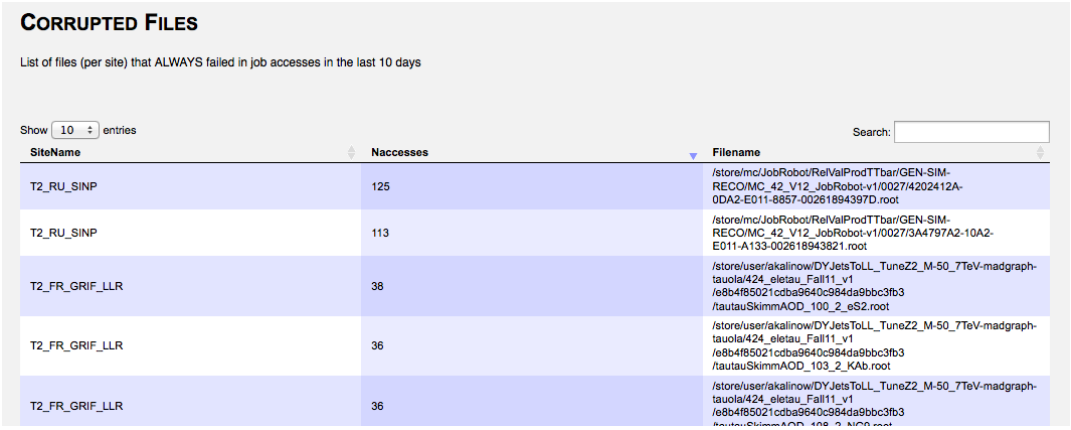
DataSet	Accesses		CPU Time		Users*day	
	[N]	[%]	[h]	[%]	[N]	[%]
/DYJetsToLL_TuneZ2_M-50_7TeV-madgraph-tauola/Fall11-PU_S6_START44_V9B-v1/AODSIM	48590	1.8	122395	7.6	44	1.0
/WJetsToLNu_TuneZ2_7TeV-madgraph-tauola/Fall11-PU_S6_START42_V14B-v1/AODSIM	52183	2.0	101991	6.3	61	1.3
/SingleMu/Run2011B-PromptReco-v1/AOD	25513	1.0	97405	6.0	57	1.2
/TTJets_TuneZ2_7TeV-madgraph-tauola/Fall11-PU_S6_START42_V14B-v2/AODSIM	67666	2.5	89911	5.6	55	1.2
/RelValProdTTbar/JobRobot-MC_42_V12_JobRobot-v1/GEN-SIM-RECO	732753	27.4	64772	4.0	8	0.2
/WJetsToLNu_TuneZ2_7TeV-madgraph-tauola/Fall11-PU_S6_START44_V9B-v1/AODSIM	25650	1.0	57038	3.5	30	0.7

Figure 4: The Dataset table shows the most popular datasets with related usage values, with the user input to configure the query for the statistics.

As previously said, all the information visualized through tables and charts is available as JSON formatted data via the HTTP API provided by the Popularity Service. Also every web page with plots contains the URL and the specified parameters needed to request the JSON of the visualized data.

2.5 Interpretation of results

In this paragraph we aim to highlight some of the views currently implemented. Regarding the user related information the system provides tables to extract users activities on each data tier, reporting statistics in terms of accesses, CPU hours of processing, dataset name, number of users, with the possibility to select a specific site to narrow the query.



CORRUPTED FILES

List of files (per site) that ALWAYS failed in job accesses in the last 10 days

Show 10 entries Search:

SiteName	Naccesses	Filename
T2_RU_SINP	125	/store/mc/JobRobot/RelValProdTTbar/GEN-SIM-RECO/MC_42_V12_JobRobot-v1/0027/4202412A-0DA2-E011-8857-00261894397D.root
T2_RU_SINP	113	/store/mc/JobRobot/RelValProdTTbar/GEN-SIM-RECO/MC_42_V12_JobRobot-v1/0027/3A4797A2-10A2-E011-A133-002618943821.root
T2_FR_GRIF_LLR	38	/store/user/akalinow/DYJetsToLL_TuneZ2_M-50_7TeV-madgraph-tauola/424_eletau_Fall11_v1/e8b4f85021cdba9640c984da9bbc3fb3/tautauSkimmAOD_100_2_eS2.root
T2_FR_GRIF_LLR	36	/store/user/akalinow/DYJetsToLL_TuneZ2_M-50_7TeV-madgraph-tauola/424_eletau_Fall11_v1/e8b4f85021cdba9640c984da9bbc3fb3/tautauSkimmAOD_103_2_KAb.root
T2_FR_GRIF_LLR	36	/store/user/akalinow/DYJetsToLL_TuneZ2_M-50_7TeV-madgraph-tauola/424_eletau_Fall11_v1/e8b4f85021cdba9640c984da9bbc3fb3

Figure 5: The corrupted file table shows the files that always failed in job accesses, with the number of access attempts per site.

2.5.1 Identification of corrupted files

The CMS Popularity Service allows to discover corrupted files that cause the failure of the CRAB jobs in a given site and, as a consequence, lead users to move away from submitting jobs on that site for specific datasets. Corrupted files account for about 3% of the CRAB job failures identified by the Popularity Service. The prompt identification of the corrupted files is a very useful source of information for site administrators. Identifying corrupted files on time allows to fix the site problem by re-transferring the file, before that other users experience the same problem. This identification is generally not possible through the systems monitoring the CRAB job status because in that case the failures are reported at the level of job, in terms of exit codes, without any information identifying the file that caused the I/O failure. The file level information is instead available in the Popularity Service. Moreover a data mining of the failures is needed to provide a reliable feedback and avoid false alarms. This task is appropriate for the Popularity Service. Corrupted files are found checking if their access was always unsuccessful in the last days of operation. The checked time range, currently of 10 days, is configurable and guarantees to not ag as corrupted those files that were temporary unavailable because of storage glitches. As shown in figure 5, the corrupted files table shows, for each Tier-2 site, all the files that always failed in job accesses in the last 10 days, ordered by number of access attempts.

2.5.2 CPU time and number of access

The service implements dedicated views to monitor the amount of CPU time spent to access datasets and datatiers (fig. 6) as well as the number of their accesses. This information is extremely useful for the CMS computing managers to monitor the usage of the computing resources of the experiment and take decisions about future strategies for the resources allocation. In addition this information allows to detect wrong behaviors of the users through the aggregation by user-id.

2.5.3 Dataset lifetime

Figure 7 is a graph created with the custom plot feature. It shows the evolution of two different datasets: the green one, created in the summer of 2011 reaches a high usage in August and then gradually loses popularity down to the point of being replaced by the blue one, created in fall of 2011 that reaches its peak in March 2012. In the graph of Figure 8 it is possible to see the same alternation between two datasets that contain the same type of data but that went through different processing, as the physicist at first keeps using the one created in the summer 2011 for their analysis but then replaces it with the one created in fall 2011.

3. Victor Site Cleaning Agent

Given the scale of the CMS grid infrastructure, it is a complex problem to control and optimize the usage of the storage. The CMS physics community consists of over 20 physics-groups that have pledges on over 50 Tier-2s, resulting in over 124 physics-group Tier-2 associations. At the moment it takes considerable human effort to control the evolution of the space and to verify that the groups are not exceeding their pledges. We have deployed a fully automated, popularity-based Site Cleaning Agent in order to scan Tier-2 sites that are reaching their space quota and suggest obsolete, unused data that can be safely deleted without disrupting analysis activity.

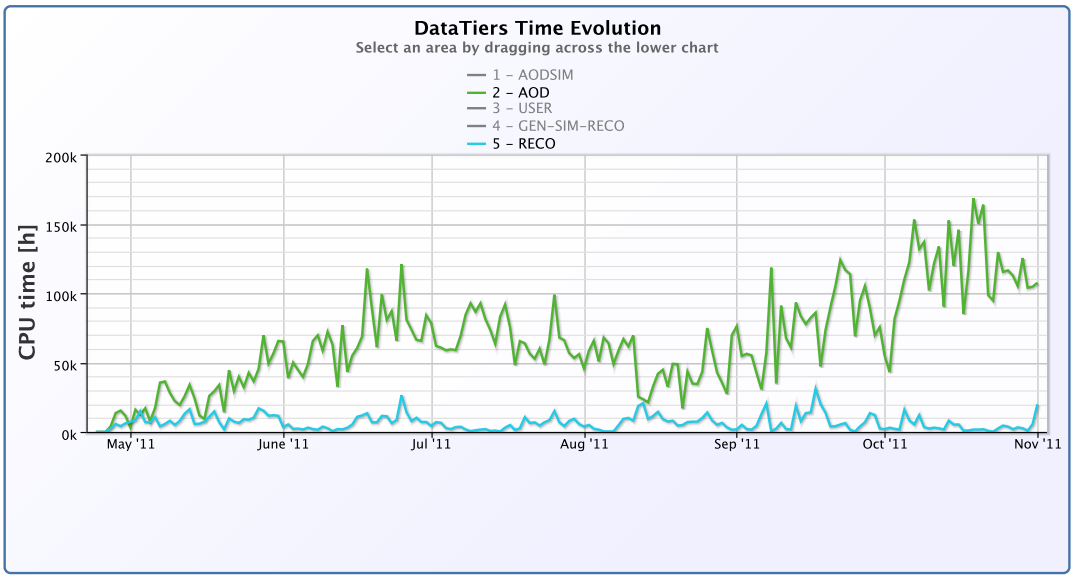


Figure 6: CPU time spent by the user analyses running on two of the most popular datatiers: AOD (green) and RECO (blue). The increasing popularity of the AOD datatier is evident since May 2011, and is a proof of the successful transition realized by the CMS Collaboration in accessing AOD instead of RECO datatiers for the analysis.

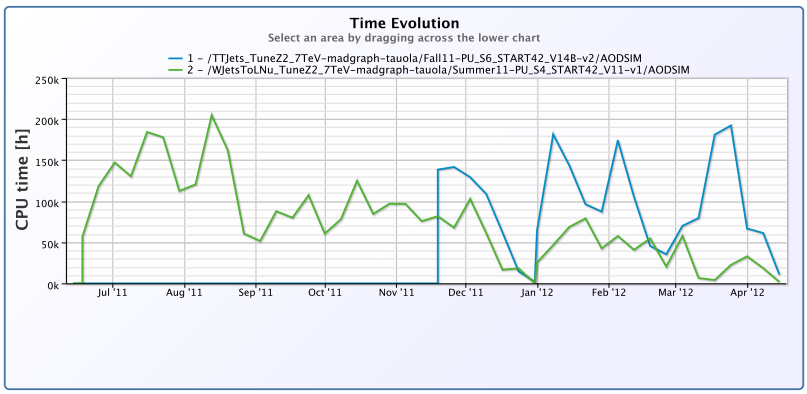


Figure 7: The Popularity custom plot shows the most popular dataset in August 2011 (green) as gradually get replaced by a different dataset (blue) that becomes the most popular, expecially in the month of March 2012.

3.1 Workflow

Victor is an agent that runs on a daily basis on a dedicated machine to discover unpopular replicas for groups reaching its quotas at a given site.

Figure 9 shows the Victor workflow. The first step is to identify groups reaching its quota. The information about space usage is taken from PhEDEx [8], the data management and transfer system in CMS. The pledges are taken from a catalogue of the resources of all CMS computing sites. The thresholds are configurable, but as default a site is considered full if it has less than 10%

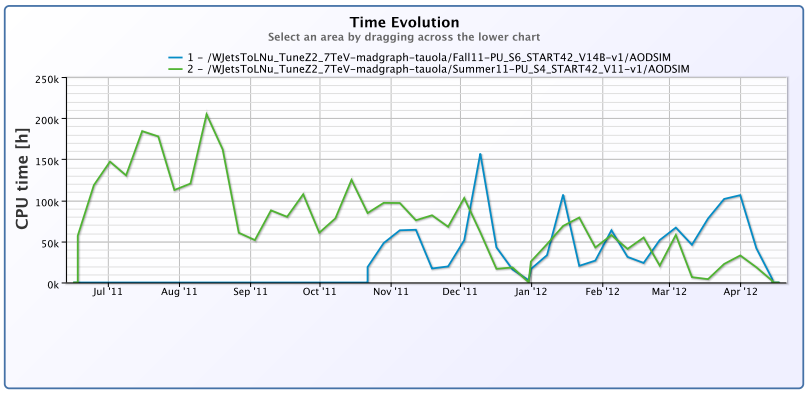


Figure 8: The Popularity custom plot shows the trend of the CPU time of two different processing of the same set of data as the one created in fall 2011 (blue) gradually replace the previous one created in summer 2011 (green)

and 15 TB of free space. In the second step Victor will try to select replicas to clean until the site has 30% or 25TB of free space and will stop then. In order to choose the unpopular replicas, Victor will get a dump of the replicas owned by the groups at each site from the Popularity service and PhEDEx, and will apply the following steps:

1. Sort blocks from oldest to newest
2. Apply a series of conditions to filter the replicas to delete
   (a) Blocks must be older than 90 days, since we want to prevent deleting fresh data that still didn't have the opportunity to become popular
   (b) Blocks can't be custodial and must have a custodial copy in a Tier-1, in order to avoid deleting a unique copy of the data
   (c) The number of file accesses in the last 30 days must be lower than [1, 10, 100]. The threshold of file accesses is increased in these steps as we need to relax the conditions to free up space on a site.

3.2 Results

All the obtained information and cleaning suggestions are stored in an Oracle database, which can be visualized through the web interface. Therefore this web page presents a basic storage accounting system (fig. 10) and also provides the interface for physics-group and site administrators that can select the suggested replicas in a copy&pastable way (fig. 11) to forward the request to the PhEDEx deletion service. Additionally, it presents a set of graphs that visualize the status of the used and total space for every site-group association. Other plots show the evolution of the space usage by a group on all the pledged sites, as well as all groups on a given site (fig. 12). Victor runs once per day, accessing fresh information on data popularity from the dedicated service. The time needed to run the full procedure, from discovering groups reaching their quota to providing

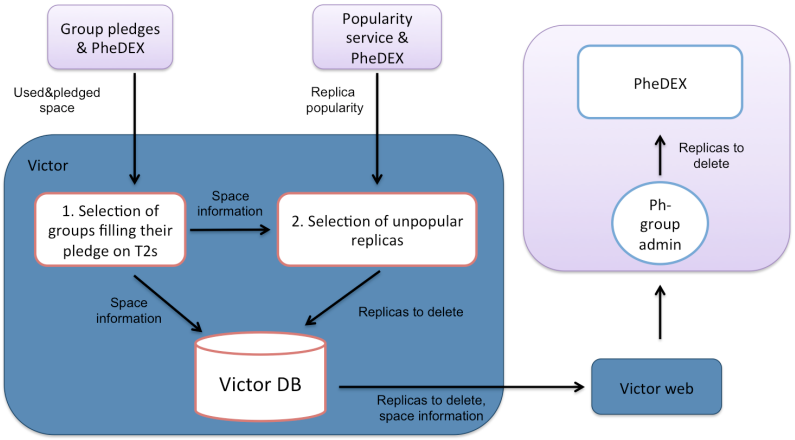


Figure 9: Site cleaning workflow.

the full list of replicas to be deleted, takes about three hours and varies on the basis of the number of selected groups. The amount of data suggested for deletion by Victor depends on several factors such as the status of the analysis activity in each physics group, the proximity to major conferences, the availability of new data processing campaigns as well as the batch deletion of obsolete processing campaigns. As an indicative feedback about the usefulness of Victor consider that in the first days of operation Victor identified about 1.5 PB of obsolete data that could be safely removed from CMS Tier-2 sites.

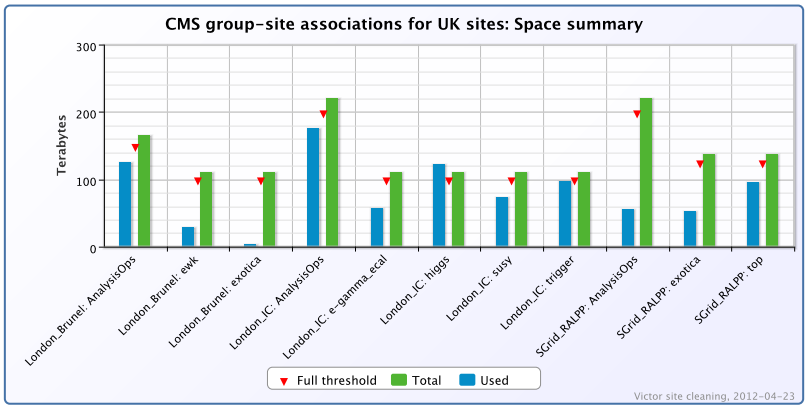


Figure 10: Space usage overview for the UK Tier-2 sites showing for each group-site association the allocated space (green), the used space (blue) and the space limit (red) at which Victor starts to clean. Similar overview plots are provided for all the other CMS Tier-2 sites, aggregated by countries.

4. Impact

The computing environment of CMS is a complex system composed by a multiplicity of different resources (CPU, sites, data) and users, it is therefore crucial to have a efficient monitoring to

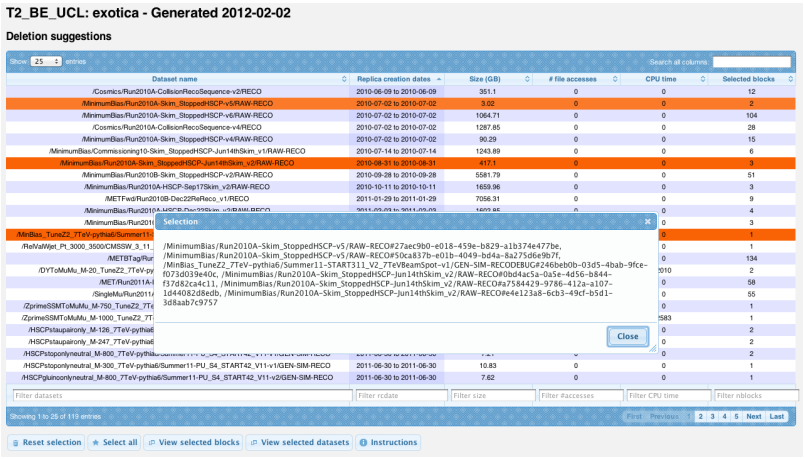


Figure 11: Selection table that shows the suggestions of unpopular, old replicas for deletion.

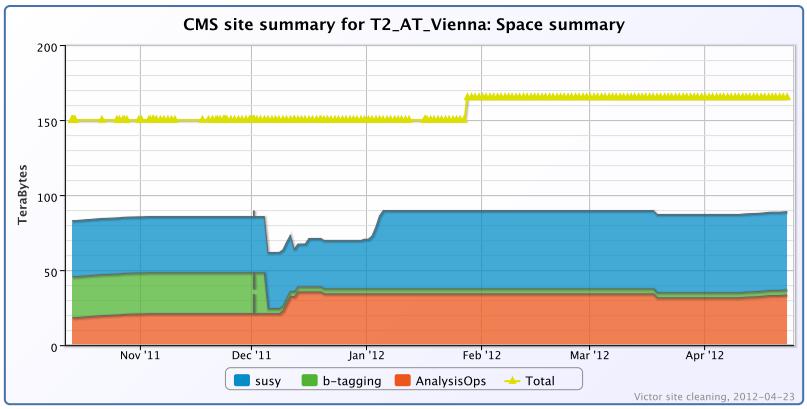


Figure 12: Evolution of the space used by groups with pledges in a Tier-2 (T2_AT_Vienna).

gain feedback and take actions. The Popularity Service can play an important role in the management of the data and to help taking strategic decision for the data placement. It has already been used by the CMS computing coordination to verify the physics analysis have switched from RECO to AOD datatier and to account the amount of unpopular datasets that have been created and made available to the users but have been rarely or never used.

Current work is to demonstrate dynamic data placement functionality based on this popularity service and integrate it in the data and workload management systems: as a consequence the pre-placement of data will be minimized and additional replication of hot datasets will be requested automatically.

The ideas in this contribution can be extended to other scientific domains that use the Grid for their data analysis and that want to learn how their community is making use of the available data and eventually implement automatic strategies to optimize their distribution. The information provided by the Popularity Service can serve as foundation for the evolution of their data placement.

5. Conclusions

This contribution presented the Popularity Service developed to monitor the usage statistic of data in the CMS computing system by gathering information from the CRAB and XRootD services. The described system has been deployed for the CMS use case although it is not a CMS specific tool. A key point of its design is to be completely decoupled by the computing services of the collaboration aiming to became a common solution for the LHC experiments. The system is showing excellent performance without any problem found in the workflow. The service provides graphical and JSON interfaces to access the collected statistics. The tools have been used to discover useful information for strategic and data placement decision.

Experiment and user activities keep increasing and oblige the experiments to ensure the future scalability of the system by automating manual operations and optimizing the usage of available resources. The strategies we are presenting go exactly in this direction. The popularity and cleaning systems are the first step towards the implementation of an optimized data placement model, where the number of dataset copies kept on grid sites is directly related to their popularity.

References

[1] TLS Group "The Large Hadron Collider Conceptual Design", CERNAC-95-05 Preprint hep-ph/0601012, 1995.

[2] The CMS Collaboration, S Chatrchyan et al, "The CMS Experiment at CERN LHC", JINST 3 S08004, 2008.

[3] The CMS Collaboration, "CMS: The computing project." Technical design report, CERN-LHCC-2005-023, ISBN 92-9083-252-5 (2005)

[4] Grandi, C., Stickland, D., Taylor, L., et al.: "The CMS computing model", CERN-LHCC-2004-035/G- 083 (2004)

[5] "LCG Computing Grid - Technical Design Report", LCG-TDR-001 CERN/LHCC 2005-024, 2005.

[6] M. Aderholz et al., "Models of Networked Analysis at Regional Centres for LHC experiments (MONARC) - Phase 2 Report," CERN/LCB 2000-001 (2000).

[7] O Buchmuller et al 2010 "The CMS CERN Analysis Facility (CAF)" J. Phys.: Conf. Ser. 219 052022

[8] R. Egeland et al., "Data transfer infrastructure for CMS data taking", Proceedings of Science, PoS(ACAT08)033 (2008).

[9] ATLAS Collaboration, JINST 3 (2008) S08003.

[10] Angelos Molfetas, et al., "Popularity framework to process dataset tracers and its application on dynamic replica reduction in the ATLAS experiment2, CHEP, Taipei, Taiwan, October 18-22, 2010

[11] Codispoti, G. et al., "CRAB: A CMS Application for Distributed Analysis", Nuclear Science, IEEE Transactions on Volume 56, Issue, Part 2, Oct. 2009 pp:2850 - 2858

[12] J. Andreeva et al., "Dashboard for the LHC experiments", J.Phys.: Conf. Ser. 119 062008, 2008.

[13] Andreas J. Peters et al., "Exabyte Scale Storage at CERN" 2011 J. Phys.: Conf. Ser. 331 052015

[14] The Scalla Software Suite: xrootd/cmsd, http://xrootd.slac.stanford.edu/

[15] The ActiveMQ project: http://activemq.apache.org/index.html

[16] The Django framework https://www.djangoproject.com/

[17] Memcached http://memcached.org

[18] S.Metson D.Bonacorsi, M.Dias Ferreira, R.Egeland, SiteDB: marshalling people and resources available to CMS

[19] The Web Redirector https://twiki.cern.ch/twiki/bin/view/LCG/WLCGWebRedirector

[20] Apache ModSecurity http://www.modsecurity.org/documentation/

[21] CERN Computer Security https://security.web.cern.ch/security/home/en/index.shtml

[22] The application/json Media Type for JavaScript Object Notation (JSON) RFC 4627

[23] DataTables http://datatables.net/

[24] Highcharts http://www.highcharts.com/