No file left behind - monitoring transfer latencies in PhEDEx

T Chwalek[a], R Egeland[b] [1], O Gutsche[c], C-H Huang[c], R Kaselis[d], M Klute[e], N Magini[f], F Moscato[c], S Piperov[g], N Ratnikova[a,h], P Rossman[c], A Sanchez-Hernandez[i], A Sartirana[j], T Wildish[k], M Yang[e], S Xie[e]

[a] KIT - Karlsruhe Institute of Technology, DE
[b] University of Minnesota, Twin Cities, US
[c] Fermi National Accelerator Laboratory, US
[d] Vilnius University, LT
[e] Massachusettes Institute of Technology, US
[f] CERN, CH
[g] Bulgarian Academy of Sciences, BG
[h] ITEP, RU
[i] Centro Invest. Estudios Avanz. IPN, MX
[j] Ecole Polytechnique, FR
[k] Princeton University, US

E-mail: nicolo.magini@cern.ch, natalia.ratnikova@kit.edu

Abstract. The CMS experiment has to move Petabytes of data among dozens of computing centres with low latency in order to make efficient use of its resources. Transfer operations are well established to achieve the desired level of throughput, but operators lack a system to identify early on transfers that will need manual intervention to reach completion. File transfer latencies are sensitive to the underlying problems in the transfer infrastructure, and their measurement can be used as prompt trigger for preventive actions. For this reason, PhEDEx, the CMS transfer management system, has recently implemented a monitoring system to measure the transfer latencies at the level of individual files. For the first time now, the system can predict the completion time for the transfer of a data set. The operators can detect abnormal patterns in transfer latencies early, and correct the issues while the transfer is still in progress. Statistics are aggregated for blocks of files, recording a historical log to monitor the long-term evolution of transfer latencies, which are used as cumulative metrics to evaluate the performance of the transfer infrastructure, and to plan the global data placement strategy. In this contribution, we present the typical patterns of transfer latencies that may be identified with the latency monitor, and we show how we are able to detect the sources of latency arising from the underlying infrastructure (such as stuck files) which need operator intervention.

1. Introduction

The CMS Experiment [1] is one of 4 large particle physics experiments that are recording proton-proton and lead ion-lead ion collisions at the LHC accelerator at CERN, Geneva, Switzerland. To archive and analyze its data, CMS and the other LHC experiments depend on the Worldwide

[1] now at Montana State University

LHC Computing Grid (WLCG) [2], a worldwide distributed data grid of over 150 compute and storage clusters varying in size, from tens of TBs to several PBs. The CMS computing model [3] has three tiers of computing facilities:

- Tier-0 at CERN (T0), used for prompt processing, tape backup and export to Tier-1 centers of CMS data.
- 7 Tier-1 (T1) centers, used for the tape backup and large-scale reprocessing of CMS data, and distribution of data products to the Tier-2 centers. The Tier-1 centers are typically at national laboratories with large computing facilities and archival storage systems.
- 50 Tier-2 (T2) facilities, where data analysis and Monte Carlo production are primarily carried out. These centers are typically at universities and do not have tape backup systems, only disk storage. Each Tier-2 is "associated" for support to a Tier-1, usually on the basis of geographical proximity, but is not restricted to sharing data with this Tier-1 center.

These sites are interconnected by high-speed networks of 1-10 Gbps. In addition, many institutions that do not pledge resources to WLCG provide a Tier-3 facility for local users, which may be also used opportunistically by CMS.

CMS is routinely producing tens of petabytes of data per year. Event data from the detector and simulated data are packed in files with an average size of a few GB, in order to reduce scaling issues with storage systems and data catalogs. In order to further simplify bulk data management, files are grouped together into "file blocks" of sizes of the order of 1 TB. File blocks are further grouped in datasets with sizes up to 100 TB defined by their physics content.

2. PhEDEx

In order to meet the CMS data distribution requirements, the Physics Experiment Data Export (PhEDEx) [4] project was designed to facilitate and manage global data transfers over the grid.

PhEDEx is based on a high-availability Oracle database cluster hosted at CERN (Transfer Management Data Base, or TMDB) acting as a "blackboard" for the system state (data location and current tasks). PhEDEx software daemon processes or "agents" connect to the central database to retrieve their work queue, and write back to TMDB the result of their actions.

PhEDEx provides two interfaces for data operations management, and for transfer monitoring: an interactive web site, and a web Data Service to retrieve information from TMDB in machine-readable formats. Performance metrics are constantly recorded in TMDB, summarizing snapshots of the TMDB state into dedicated status monitoring tables at regular intervals. Historical information is further aggregated from the status tables into time-series bins of data on transfer volume, transfer state counts, and number of failures. Since the website and data service only access these monitoring tables instead of the live tables to provide monitoring information, the performance of the user monitoring and of the transfer system are largely decoupled.

Since the beginning of the LHC physics run at 7 TeV in March 2010, CMS has been steadily transferring data with PhEDEx with peaks in global speed between all sites exceeding 4.5 GB/s, distributing 65 PB of replicas over all sites [5].

3. PhEDEx transfer workflow

The fundamental unit of data for PhEDEx data transfers is the file block. Users can request transfers of file blocks (usually grouped in datasets) through the web interfaces, by placing a subscription for the blocks to a given destination node. Typical use-cases for subscriptions are the transfer of data to their custodial archival location, or to a site where CPU resources are available for processing and analysis. To avoid data loss and ensure that data are fully available for processing, PhEDEx is designed to aim for eventual completion of all transfer subscriptions.When a new subscription is created, the central agents of PhEDEx running at

CERN expand the block into its current file replicas, and calculate the path of least cost for each file from the available sources to the destination. The download agents running at the destination sites receive these tasks and initiate the data transfers using technology-specific backends interacting with the WLCG middleware. In case of transfer success, the central agents record the new file replica in TMDB; in case of transfer failure, the central agents automatically exclude the file from the transfer queues for a short time (about 1 hour), then reschedule the transfer, potentially rerouting it from a different source. This backoff policy leaves slots available for transfers that have a chance to succeed, optimizing for bulk utilization of the available network bandwidth (still a limiting and unreliable resource when PhEDEx was originally designed in 2004) rather than fast delivery of individual files. After all files in the block have been replicated to the destination, the central agents collapse again the file replicas into the block, thereby reducing the amount of data location information that needs to be tracked in TMDB.

The intelligent automation of retry policies ensures that most transfer subscriptions are completed without operator intervention, even on links with low transfer quality. However, a large amount of operator effort is still needed to identify transfers that are permanently stuck and need manual intervention to reach full completion. A common reason for this is a failure or misconfiguration of the storage or the network; this is usually easy to identify as it leads to a drop of transfer quality to 0%. Another common issue preventing the completion of block subscriptions are so-called “transfer tails”, referring to the observation that blocks are left in an incomplete state at the site for a long period of time, even if the majority of the files in the block have been transferred successfully. This usually is an indication that a file in the block is permanently stuck or corrupt at the source sites, and operator intervention is needed to recover the file or invalidate it declaring it lost. This type of failure is less apparent in the monitoring, since it may affect only a few files on links with an otherwise good transfer quality.

4. Block completion latency monitoring in PhEDEx

Since the deployment of version 2.6.2 in 2008, PhEDEx has been instrumented with a system to monitor the latency of block transfer completion. The BlockAllocator agent that is responsible for monitoring subscriptions also recorded the timestamps of events related to block completion in the t_log_block_latency database table. The events recorded were:

- the time when the block was subscribed
- the time when the first file in the block was routed for transfer
- the time when the first file in the block was successfully replicated to the destination
- the time when the last file in the block was successfully replicated to the destination

The total latency for block replication could be defined as the time elapsed between the subscription and the successful arrival of the last replica at destination. If the subscription was suspended by a user while the transfers were in progress, the agent also took care of measuring and recording in the table the time elapsed during the suspension. Subtracting this value from the total latency, we could measure the latency introduced by the system itself, i.e. by PhEDEx and the underlying transfer infrastructure, excluding any additional latency introduced by human intervention.

A prototype web page was deployed to display the overall distribution of block completion latencies by destination site; however, the level of detail recorded in the database table was considered insufficient for many applications. In particular, it provided little insight into the reason for a perceived high latency in a particular transfer: for example, there was no way to distinguish between the case of a high-latency transfer proceeding at a low regular rate, and the case of a transfer for which the latency was dominated by a few stuck files in the “transfer tail”.

For this reason, a script was developed to measure transfer latencies for individual files within a block. Since the details of file transfer tasks are cleaned up from the database immediately after task completion and are only recorded in the log files of the central agents of PhEDEx, the script needed to parse the log files as input and could not be used to provide a live monitoring. Nevertheless, the script proved useful in the post-mortem analysis of several transfer campaigns during the computing challenges in 2008 and 2009, revealing how the retry and rerouting policies implemented in the PhEDEx FileRouter were working in practice. The results were used to guide the optimization of the FileRouter agent in version 3.3.0 in 2010.

5. Development of the new latency monitoring system

With the accumulation of ever-increasing volumes of data since the startup of LHC collisions in 2009, the CMS data placement policy is progressively evolving from the original static pre-placement model. Thanks to the increasead network reliability, it is now feasible to optimize dynamically the use of computing resources, transferring datasets to sites with CPU resources available for processing and analysis. The focus for transfer operations is therefore shifting to a fast delivery of "hot" data, making a detailed monitoring of transfer latencies more important than ever.

For this reason, in August 2011, the PhEDEx development team gathered in a focused "codefest" workshop to develop a prototype for an extension of the latency monitoring system. We decided to complement block-level information with details on the transfer history of individual files within the block. The central agents of PhEDEx now record this information in two new tables: the BlockAllocator agent records the timestamps of events related to block completion in t_dps_block_latency, while the FilePump agent responsible for collecting the results of transfer tasks records a summary of the transfer history of each file in the block in the t_xfer_file_latency table. The events recorded in the file-level table include:

- the time when the file was first activated for routing
- the time when the file was first staged to disk for transfer (for transfers from T1 archival tape systems)
- the time of the first transfer attempt
- the time of the most recent transfer attempt
- the time of the final successful transfer attempt
- the time when the file was successfully migrated to tape (for transfers to T1 archival tape systems)

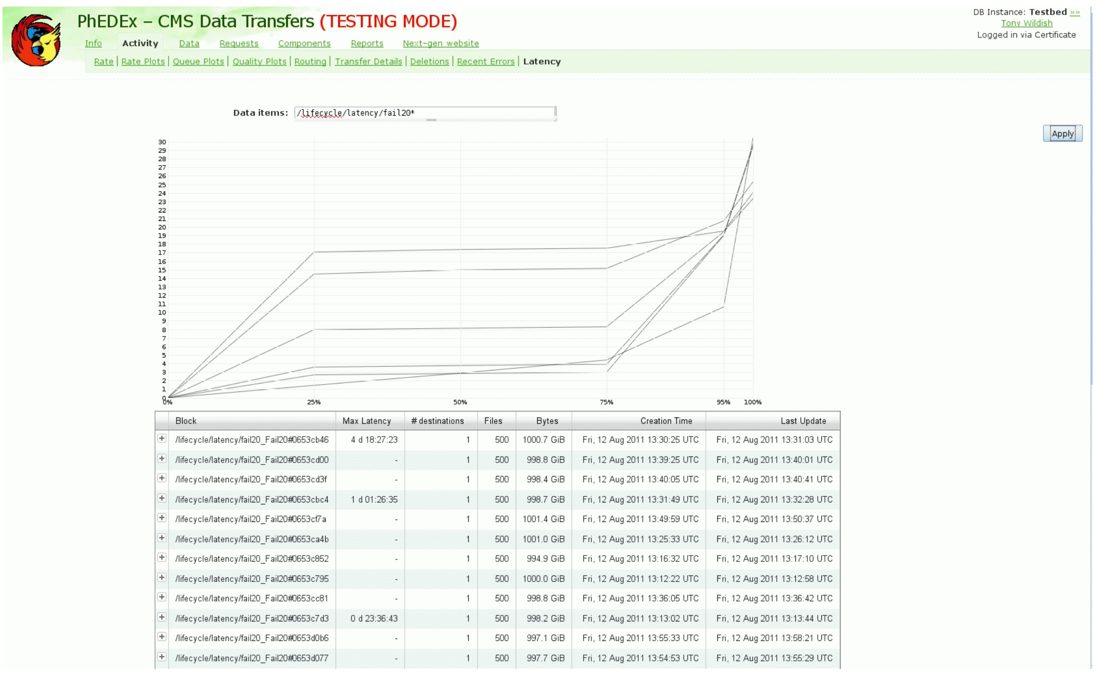
The FilePump agent also records the number of transfer attempts needed for the file to arrive at destination, as well as the source node of the first and last transfer attempt (which may be different, if the transfer was rerouted from a different source after the first attempt). To prevent a performance degradation of the FilePump agent, we decided to keep in the new tables only the information for blocks currently in transfer, archiving entries for completed blocks from t_dps_block_latency to the t_log_block_latency table, in which the block-level statistics are kept indefinitely, and from t_xfer_file_latency to a new t_log_file_latency table, from which the file-level statistics are eventually cleaned up after 30 days. With the file transfer details available, we also extended the t_log_block_latency table to record additional events related to block completion:

- the time when 25% of the files in the block have been replicated to the destination
- the time when 50% of the files in the block have been replicated to the destination
- the time when 75% of the files in the block have been replicated to the destination
- the time when 95% of the files in the block have been replicated to the destination

In addition, we record the total number of transfer attempts needed to transfer all files in the block, as well as the source node for the majority of files in the block.

Finally, complementing the measurement of transfer latencies, we also provided a table to record the estimated time of arrival for blocks still in transfer, calculated by the FileRouter when choosing the optimal source for file transfers. If no estimate is possible, for example when no valid path from the source to the destination node exists, the reason for the missing estimate is given instead, to provide the operators with a suggestion on the external interventions needed to complete the transfer.

6. Visualisation of latency monitoring information


PhEDEx – CMS Data Transfers (TESTING MODE)
DB Instance: Testbed »»
Tony Wildish
Logged in via Certificate
Info
Activity
Data
Requests
Components
Reports
Next-gen website
Rate | Rate Plots | Queue Plots | Quality Plots | Routing | Transfer Details | Deletions | Recent Errors | Latency
Data items: /lifecycle/latency/fail20*
Apply
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
0% 25% 50% 75% 95% 100%

	Block	Max Latency	# destinations	Files	Bytes	Creation Time	Last Update
+	/lifecycle/latency/fail20_Fail20#0653cb46	4 d 18:27:23	1	500	1000.7 GiB	Fri, 12 Aug 2011 13:30:25 UTC	Fri, 12 Aug 2011 13:31:03 UTC
+	/lifecycle/latency/fail20_Fail20#0653cd00	-	1	500	998.8 GiB	Fri, 12 Aug 2011 13:39:25 UTC	Fri, 12 Aug 2011 13:40:01 UTC
+	/lifecycle/latency/fail20_Fail20#0653cd3f	-	1	500	998.4 GiB	Fri, 12 Aug 2011 13:40:05 UTC	Fri, 12 Aug 2011 13:40:41 UTC
+	/lifecycle/latency/fail20_Fail20#0653cbc4	1 d 01:26:35	1	500	998.7 GiB	Fri, 12 Aug 2011 13:31:49 UTC	Fri, 12 Aug 2011 13:32:28 UTC
+	/lifecycle/latency/fail20_Fail20#0653cf7a	-	1	500	1001.4 GiB	Fri, 12 Aug 2011 13:49:59 UTC	Fri, 12 Aug 2011 13:50:37 UTC
+	/lifecycle/latency/fail20_Fail20#0653ca4b	-	1	500	1001.0 GiB	Fri, 12 Aug 2011 13:25:33 UTC	Fri, 12 Aug 2011 13:26:12 UTC
+	/lifecycle/latency/fail20_Fail20#0653c852	-	1	500	994.9 GiB	Fri, 12 Aug 2011 13:16:32 UTC	Fri, 12 Aug 2011 13:17:10 UTC
+	/lifecycle/latency/fail20_Fail20#0653c795	-	1	500	1000.0 GiB	Fri, 12 Aug 2011 13:12:22 UTC	Fri, 12 Aug 2011 13:12:58 UTC
+	/lifecycle/latency/fail20_Fail20#0653cc81	-	1	500	998.8 GiB	Fri, 12 Aug 2011 13:36:05 UTC	Fri, 12 Aug 2011 13:36:42 UTC
+	/lifecycle/latency/fail20_Fail20#0653c7d3	0 d 23:36:43	1	500	998.2 GiB	Fri, 12 Aug 2011 13:13:02 UTC	Fri, 12 Aug 2011 13:13:44 UTC
+	/lifecycle/latency/fail20_Fail20#0653d0b6	-	1	500	997.1 GiB	Fri, 12 Aug 2011 13:55:33 UTC	Fri, 12 Aug 2011 13:58:21 UTC
+	/lifecycle/latency/fail20_Fail20#0653d077	-	1	500	997.7 GiB	Fri, 12 Aug 2011 13:54:53 UTC	Fri, 12 Aug 2011 13:55:29 UTC

Figure 1. Prototype of the web page used for latency monitoring visualisation.

Data Service APIs to retrieve the contents of the block latency tables were developed together with the initial prototype of the system. Taking into account the different use-cases for the generation of latency reports, the APIs accept several filter options: data can be selected on block name or destination node, to single out specific subscriptions; on the subscription time and several other timestamps, to generate global performance reports for specific time periods; or on latency, to identify outstanding transfers. A prototype web page to access latency information was also developed (Fig. 1), displaying data retrieved through the API in a nested data table and a sample plot.

7. Validation of the latency monitoring system

The new latency monitoring system was deployed to a dedicated PhEDEx Testbed running against a snapshot of the Production instance TMDB. To validate the system in a realistic workflow, we started a simulation of the Production infrastructure, running the central agents and site agents for the T0, 7 T1s, and some T2 sites. The site download agents were set up to execute fake transfers with a configurable bandwidth and failure probability. To simulate the ongoing production of new CMS data, we injected new files regularly using the “lifecycle

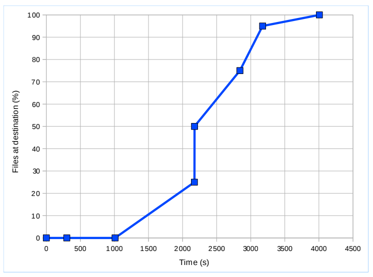


Figure 2. Percentage of files at destination as a function of time since subscription, for a block transferred with perfect transfer quality.

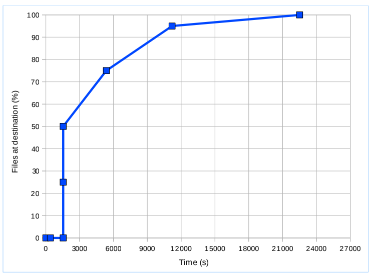


Figure 3. Percentage of files at destination as a function of time since subscription, for a block transferred with 80% transfer quality.

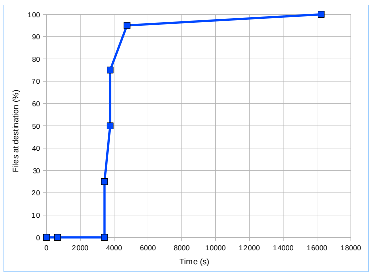


Figure 4. Percentage of files at destination as a function of time since subscription, for a block with a few permanently stuck files (“transfer tail”).

agent” [6]. The agent was updated for this occasion, adding several new features needed to reproduce typical operational issues during the validation, such as the ability to mark a specific fraction of files within a block as “stuck”, emulating hard failures on the source files. To validate the functionality of the system, we configured the lifecycle agent to inject file blocks that would produce different patterns of transfer failures in the download agents: blocks in which all files would be transferred successfully, blocks in which each file transfer attempt had the same random probability to fail, and blocks in which most of the files would transfer on the first attempt but some of the files would persistently fail. Sample plots of number of files in the block at destination as a function of time for each category are shown in Fig. 2, 3 and 4. The different patterns in the block completion plots are visible.

We also verified the scalability of the new system, configuring the “lifecycle agent” to inject new files at an accelerated rate. The scale test was useful to identify and solve a few performance bottlenecks in the initial prototype of the latency monitor, and the final version showed no significant performance degradation running simulated transfers at a rate up to 10 times the scale of Production transfers.

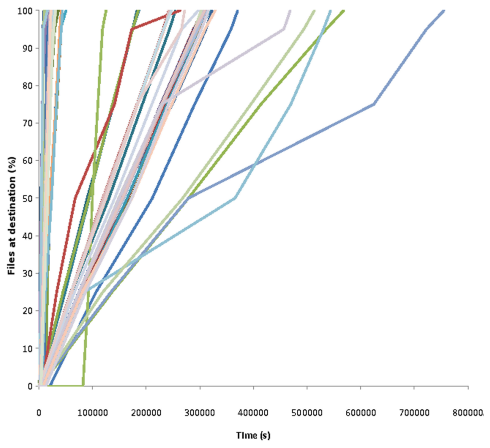


Figure 5. Percentage of files at destination as a function of time since subscription, for several blocks transferred in the Debug instance of PhEDEx.

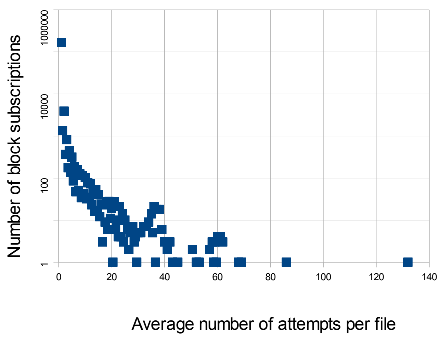


Figure 6. Distribution of the average number of transfer attempts per file needed to complete a block subscription.

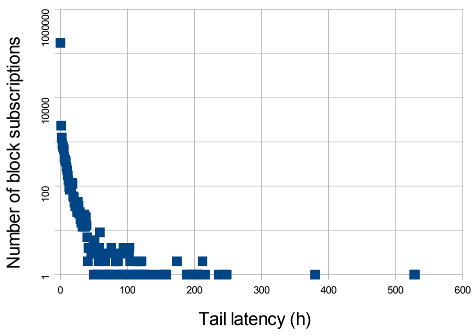


Figure 7. Distribution of the time needed to transfer the "tail" of a block, defined as the last 5% of the files in a block subscription.

8. Deployment of the latency monitoring system

The new latency monitoring tables and agents were released with version 4.1.0 of PhEDEx in Spring 2012 and deployed to the PhEDEx Debug instance used to test link performance with transfers of real files. Since then, it has collected full logging information for tens of thousands of block transfers and hundreds of thousands of file transfers; Fig. 5 shows block completion curves for several blocks transferred in the Debug instance.

Deployment to the Production instance of PhEDEx was performed in early Summer 2012. At the time of writing, transfer operators are still gathering experience with the new latency monitoring system, so we can only perform a preliminary analysis of the data, without drawing any definitive conclusion. Examples of the statistics that can be generated with the latency monitor are shown in Fig. 6 and 7.

In Fig. 6, the average number of transfer attempts per file is histogrammed for all block subscriptions (about 180000) completed in a period of one month. While most of the block subscriptions are completed transferring every file on the first attempt, in about 10% of the cases at least one retry is needed to transfer all files in the block, and in about 0.5% of the cases at least 10 attempts per file are needed to complete the subscription.

In Fig. 7, a histogram of the "tail latencies", defined as the time needed to transfer the last 5% of the files in a block, is shown for all subscriptions completed in the same period of one month: in about 1% of the cases, the files in the "tail" are transferred with a delay of more than 12 hours.

Based on a more detailed analysis in the upcoming months, we will finalise the metrics used to alert about stuck subscriptions and prompt for manual intervention.

References

[1] The CMS Collaboration 2008 The CMS experiment at the CERN LHC JINST **3** S08004
[2] Knobloch J *et al.* 2005 LHC Computing Grid Technical Design Report CERN-LHCC-2005-024
[3] Bonacorsi D 2007 The CMS computing model *Nucl. Phys.* B *(Proc. Suppl.)* **172** 53-56
[4] Egeland R, Wildish T and Metson S 2008 Data transfer infrastructure for CMS data taking PoS(ACAT08)033
[5] Kaselis R, Piperov S, Magini N, Flix J, Gutsche O, Kreuzer P, Yang M, Liu S, Ratnikova N and Sartirana A 2012 CMS data transfer operations after the first years of LHC collisions, these proceedings
[6] Sanchez-Hernandez A, Egeland R, Huang C-H, Ratnikova N, Magini N and Wildish T 2012 From toolkit to framework - the past and future evolution of PhEDEx, these proceedings