Monitoring data transfer latency in CMS computing operations

Bonacorsi D[1], Diotalevi T[1], Magini N[2], Sartirana A[3], Taze M[4], and Wildish T[5]

[1] University of Bologna
[2] Fermi National Accelerator Laboratory
[3] Ecole Polytechnique of Paris
[4] Cukurova University
[5] Princeton University

E-mail: Daniele.Bonacorsi@bo.infn.it, Tommaso.Diotalevi@studio.unibo.it, nicolo.magini@cern.ch, meric.taze@cern.ch, sartiran@llr.in2p3.fr, awildish@princeton.edu

Abstract.

During the first LHC run, the CMS experiment collected tens of Petabytes of collision and simulated data, which need to be distributed among dozens of computing centres with low latency in order to make efficient use of the resources. While the desired level of throughput has been successfully achieved, it is still common to observe transfer workflows that cannot reach full completion in a timely manner due to a small fraction of stuck files which require operator intervention.

For this reason, in 2012 the CMS transfer management system, PhEDEx, was instrumented with a monitoring system to measure file transfer latencies, and to predict the completion time for the transfer of a data set. The operators can detect abnormal patterns in transfer latencies while the transfer is still in progress, and monitor the long-term performance of the transfer infrastructure to plan the data placement strategy.

Based on the data collected for one year with the latency monitoring system, we present a study on the different factors that contribute to transfer completion time. As case studies, we analyze several typical CMS transfer workflows, such as distribution of collision event data from CERN or upload of simulated event data from the Tier-2 centres to the archival Tier-1 centres. For each workflow, we present the typical patterns of transfer latencies that have been identified with the latency monitor.

We identify the areas in PhEDEx where a development effort can reduce the latency, and we show how we are able to detect stuck transfers which need operator intervention. We propose a set of metrics to alert about stuck subscriptions and prompt for manual intervention, with the aim of improving transfer completion times.

1. Introduction

The CMS experiment [1] at the LHC accelerator is concluding the first Long Shutdown (LS1) after a successful first run of data taking (Run-1), with Run-2 starting in Summer 2015. In the original CMS Computing model [2], one of the main concepts of the data management [3] was that jobs go where the data is, and no data moves in response to job submissions. In such model, the importance of adequate policies and tools for data placement is vital. Over

the years, this motivated the CMS Computing project to design, build and operate a robust and reliable solution to perform transfers of massive volumes of data among computing centres of the Worldwide LHC Computing Grid (WLCG) [4, 5], called PhEDEx [6, 7, 8, 9]. PhEDEx is a reliable and scalable dataset replication system based on a central database on an Oracle instance running at CERN and a set of highly specialised, loosely-coupled, stateless software agents distributed at sites. In production for CMS since more than 10 years, PhEDEx moved 150 PB during Run-1, and it is currently moving about 2.5 PB per week among about 60 sites. The PhEDEx design aims at providing the highest possible transfer completion rate, despite possible infrastructural unreliabilities, achieved via intelligent fail-over tactics and automatic retrials. During the several years of its operations, including the first LHC data taking period (Run-1) and the first LHC Long Shutdown (LS1), a large set of data concerning the latencies observed in all transfers between all Tiers has been collected. The study of this data set is allowing a categorisation of the different root sources of such latencies and shaping the strategy to attach this problem and increase the overall performance of the PhEDEx system.

2. Instrumenting PhEDEx to collect latency data

The atomic unit for transfer operations in PhEDEx is the file block: an arbitrary group of O(100-1000) files in the same dataset. To achieve scalability, PhEDEx doesn't keep a permanent record of the states of individual files: all information is aggregated at the level of block after transfers are completed.

This level of detail is sufficient for replica location, but it is not enough to identify problems that increase latency in block transfers: for example, there is 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 is dominated by a few stuck files in the "transfer tail".

For this reason, in 2012 we instrumented the central agents of PhEDEx with a detailed latency monitoring system [10], collecting file-level information on transfers in historical monitoring tables to complement block-level information, and providing corresponding Data Service APIs to retrieve the monitoring data for further analysis.

The BlockAllocator agent that is responsible for monitoring subscriptions records the timestamps of the main events related to block completion in the t_dps_block_latency table: the time when the block was subscribed and the time when the last file in the block was successfully replicated at destination. The difference between these two timestamps can be defined as the total latency for block replication as experienced by users, which may also include time spent while the subscription was manually suspended by an operator. In this case, the agent also measures and records in the table the time elapsed during the suspension, and subtracting this value from the total latency we measure what we define as the "PhEDEx latency", i.e. introduced by PhEDEx itself and the underlying transfer infrastructure rather than human intervention.

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, including the time when the file was first activated for routing, the time of the first transfer attempt and of the final successful transfer attempt, 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 attempts (which may be different if the transfer was rerouted).

For performance reasons, the entries in these live tables are archived to historical tables after the transfer is completed: file-level statistics into t_log_file_latency, which is eventually cleaned up after 30 days, and block-level statistics into t_log_block_latency, in which they are kept indefinitely and integrated with additional events related to block completion:

- 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 at destination

- the times when 25%/50%/75%/95% of the files in the block were replicated at destination
- the time when the last file in the block was successfully replicated at 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.

3. Cleaning data and defining variables

In the last two years, thanks to the instrumentation detailed in the previous session, the PhEDEX system collected records about the latency of roughly 3 million block subscriptions. This data include several useful informations:

- block informations: block, number of files, size, timestamps of the block opening and closing;
- subscription informations: destination site, custodiality flag, total time of suspension, creation date of the subscription, priority;
- source sites informations: site from which most files where transferred and number of files transferred from this site;
- informations about the transfer execution: timestamps of the first request, the first transfer, the 25%/50%/75%/95% completion and of the last transfer. The total number of transfers attempts.

This huge amount of records represents the raw input of our analysis. Before starting the actual analytic work, however, this set of data has to undergo some cleaning and processing steps in order to remove the items of no interest and define useful derived observables.

First of all we cleaned ill defined data, that is records with missing or inconsistent entries. These are mostly issued from test transfers and represent roughly the 5% of the total amount.

In the remaining set there are roughly 780,000 transfers that took place while the block was still open and growing in size. These would actually be well defined targets for our analysis but their treatment may render the whole process uselessly complex. Therefore, for the time being, we decided to remove these entries from the set. For the same reason we decided to remove all the 62,000 transfers entries which have been suspended during their execution.

We also defined a cutoff of 3 hours on the transfer time. The idea behind this choice is that, seen the typical time scale of data transfers in CMS, if a transfer takes less than 3 hours from the subscription to the completion we can, arbitrarily but sensibly, argue that it is not a candidate for having latency problems. This cutoff removes roughly 960,000 items.

At this point we are left with 1 million transfers records among which we have to point out those having latency issues. In order to determine the signal of a latency problem we defined few "skew" variables showing the transfer rate ratio between the time spent in a small portion (last 5% or first 25%) and the X percent from the beginning or to the end of a transfer. More precisely, we define the following 4 sets of variables

- skew X variables:

$$Skew_X = \frac{\text{(time spent transferring the LAST 5 percent of the files)}}{\text{(time spent transferring the FIRST } X \text{ percent of the files)}} * \frac{X}{5} \qquad (1)$$

  where $X = 25, 50, 75, 95$;

- skew last X variables:

$$SkewLast_X = \frac{\text{(time spent transferring the LAST 5 percent of the files)}}{\text{(time spent transferring the LAST } X \text{ percent of the files)}} * \frac{X}{5} \qquad (2)$$

  where $X = 25, 50, 75$;

- reverse skew X variables:

$$RSkew_X = \frac{\text{(time spent transferring the FIRST 25 percent of the files)}}{\text{(time spent transferring the FIRST } X \text{ percent of the files)}} * \frac{X}{25} \tag{3}$$

  where $X = 50, 75, 95$;
- reverse skew last X variables:

$$RSkewLast_X = \frac{\text{(time spent transferring the FIRST 25 percent of the files)}}{\text{(time spent transferring the LAST } X \text{ percent of the files)}} * \frac{X}{25} \tag{4}$$

  where $X = 5, 25, 50, 75$.

On a transfer which is ideally running at a constant rate all these variables have value one. Thus, one or more of the "skew" variables that significantly differs from unity can be considered a good signal of a transfer having latency issues. In particular this is the signal of a latency "tail" in a generalized sense, that is a section of the transfer which has been significantly slower than the others.

This led us to a further data cleaning. Tails analysis requires in fact defined values for the "skew" variables as well as data blocks which are big enough to have a "bulk" and a "tail". Thus, data for this analysis was further skimmed keeping only blocks with more than 5 files, size larger than 300GB and with defined values for all the "skew" variables. This leaves us with a final set of roughly 42,000 transfers entries which are the actual basis of our latency tails analysis.

4. Types of latency

At this point our analysis requires defining the different ways data transfers may be affected by latency issues and, if possible, the signatures of these latency types. We make no claim to completeness and, at this stage, we just focus on three main cases.

Tails First of all let's consider block transfers that have "tails". That is: one or few files that take much longer to get transferred than the rest of the block. We can look for this type of latency by selecting transfers in which the time needed for moving the last 5% of bytes is larger than a given threshold δ. To prevent very large blocks from being included even if they have no real latency issues we may add to δ an offset which depends from the size of the block and a reference speed parameter v. In formulas

$$\Delta T_{last5\%} > \delta + \frac{S}{20v} \tag{5}$$

where $\Delta T_{last5\%}$ is the time of the last 5% of replicas and S the size of the block. Sensible values for the parameters are $v = 5MB/s$ and $\delta = 10h$.

Early Stuck Then we can consider block transfers that begin with serious performance issues and start flowing properly only after some time, presumably once such issues have been fixed. We may call such transfers "early stuck". To find this type of latency we can select transfers in which the time for the first replica is larger than a given threshold δ. An offset which depends from the average size of the files and the reference rate parameter v will prevent blocks with very big files to show up even if they have no latency issues. In formulas

$$\Delta T_{1st} > \delta + \frac{\langle S \rangle}{v} \tag{6}$$

where ΔT_{1st} is the time of the first replica and $\langle S \rangle$ the average file size in the datablock. We can use again the values above as good estimations of v and δ.

Many Small Blocks If we take large datasets with many small (few files) blocks, we can consider the latency caused, at the dataset level, by some stuck blocks. This is much like the latency tails discussed above but replacing files by blocks and blocks by datasets. This is another important latency type that we will not be able to analyze since it affects blocks of a type that we ruled out of our sample (they are below the size limit).

Despite this list of data types is clearly incomplete, we will see that it covers most of the real-life cases. In what follows we will in fact proceed in a more detailed analysis of the transfers affected by the latency types defined in this section. We will see which are their causes and their consequences on the CMS operations.

5. Latency of big blocks with long tails

The extended CMS data production and processing tasks that run over the distributed WLCG infrastructure may occasionally produce corrupted outputs. That is, one or few files in a datablock which are missing - resulting from a non-handled failure in the stageout to grid storage - or have wrong size/checksum. PhEDEx has been designed to deal with such data corruption events and is able to detect the corrupted files - relying on FTS internal checking mechanisms and on pre/post validation scripts - and tag the corresponding transfers as failing. Moreover PhEDEx tries to minimize the impact of the corrupted files on the data placement operations - and in particular on the transfers throughput - by re-queuing such files with low rank while it keeps on transferring the rest of the data. As the transfers of the corrupted files keep on failing systematically, PhEDEx suspends them for a longer time.

Thus the effect of file corruption during data production is observed, at data placement level, as a latency tail issue in PhEDEx transfers. The impact of these missing/corrupt files can only be seen in the last few percentages of the transfer of the whole block. Hence, while transfer rates in the first 95% have higher values, rate in the last 5% drops off to quite low values as shown in fig. 1. In a similar way, $RSkewLast_5$ values (see eq. 4) are much lower than $Skew_{95}$ values (see eq. 1) in fig. 2.

The underlying reasons for file production failures have been extensively investigated by the CMS PhEDEx team. It was observed that they are mostly due to transient storage problems, which explains why only a few files are affected from this issue. Tier-2's grid sites are particularly exposed to this source of latency as Tier-1's generally have more reliable storage systems. This can be seen in fig. 3.

Despite only few files for each data sample are concerned, this type of latency issues can have a very important impact on the CMS operation. In most cases, in fact, the transferred data is useless to CMS jobs until a 100% transfer completion is reached. Thus, it is quite important to find the stuck files being the root cause of the latency, and fix the identified problems as soon as possible. In most cases, solving this problem requires a manual expert operator intervention, consisting of either replacing the file (if it has other valid replicas) or otherwise invalidating it and announcing it as lost.

6. Latency of blocks that get stuck early

The overall CMS production resources consist of a highly interconnected pool of WLCG sites of different capacities and belonging to different Tier levels. All of them are actively used in processing/production activities, and an efficient and closely monitored data transfer system on this complex topology is essential. It is not unexpected that, in exploiting this heterogeneous set of resources over long periods of time, some permanent or transient errors due to hardware/software problems are experienced. Although CMS tries hard to detect the problematic sites in advance and proactively takes measures to use them only if they are safe both for processing and for data transfers, it is not always possible to select with 100% purity on

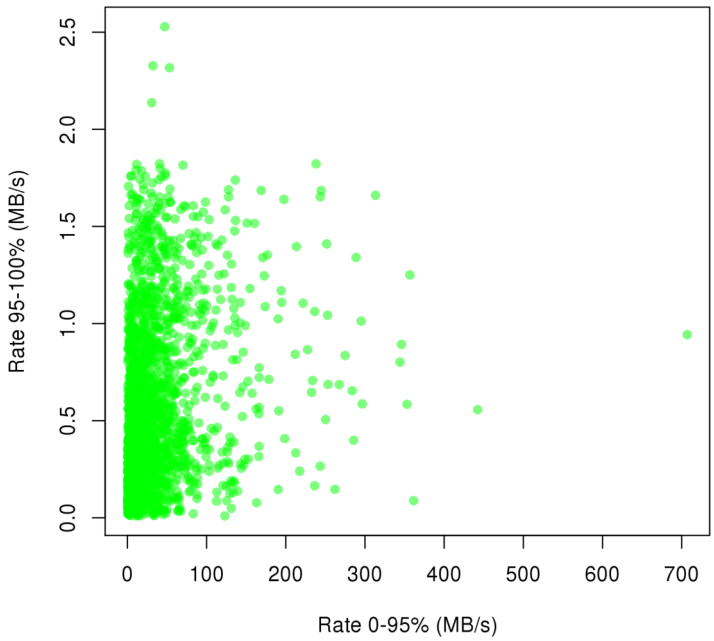

Figure 1. Transfer rate of the last 5% of the block vs rate of the first 95%. For transfers tagged as having latency tails according to eq. 5.

long periods of time a subset consisting of only sites in perfect shape. Hence, it is expected that the overall transfer system needs to always deal with a bunch of problematic sites that should nevertheless be used as either source or destination of some data transfer tasks. In many of these cases, as the problem may just be at the infrastructural site level, these kind of transfers show up as stuck in the very beginning, i.e. even in the transfer of the very first file.

These transfers are hence reported as stuck at 0% completion for hours as shown in fig. 5, so it is far easier to detect them with respect to any other latency type.

In addition, $SkewLast_{75}$ values (see eq. 2) are quite small while $Skew_{25}$ (see eq. 2) have quite large values in fig. 4 as expected.

The solution, however, is not straight-forward: it might require site admin intervention at the source/destination site, or even central operators/experts involvement. The price to pay if not promptly identified is high: only a quick problem identification, attack and fix can avoid to pile up delays and additional work load at a later stage.

7. Latency of datasets with many small blocks

In all CMS processing activities, datasets - to be intended as collections of files with specific physics content - are produced via the contribution of multiple WLCG sites supporting the CMS workflows. In most cases, the contributing sites are not the final destination where the dataset is supposed to be stored. Hence, a PhEDEx subscription is made to ship all data to a tape endpoint - and possibly to some disk endpoints - chosen according to CMS policies. However, some of the blocks might be located at sites having temporary network or storage problems which can cause a latency in the overall data placement task.

Big datasets composed of many small blocks (few files per block) are particularly exposed to

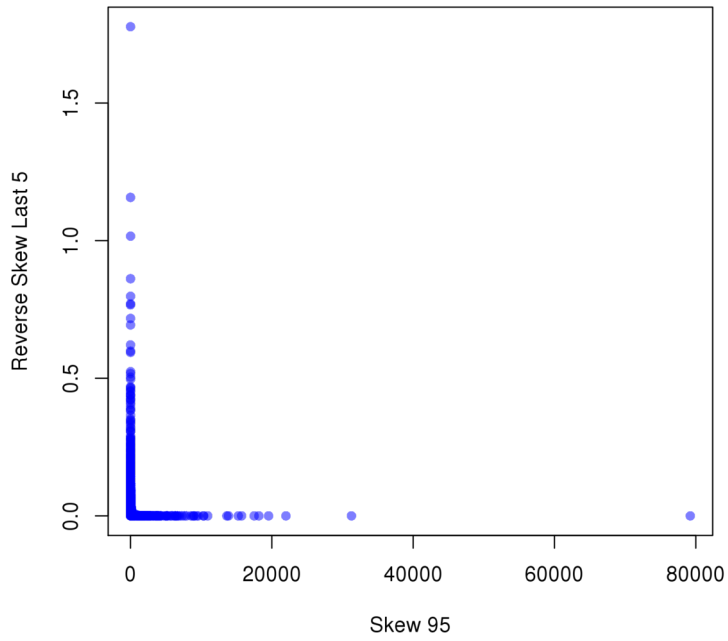

Figure 2. $RSkewLast_5$ versus $Skew_{95}$ as defined in eq. 4 and eq. 1. For transfers tagged as having latency tails according to eq. 5.

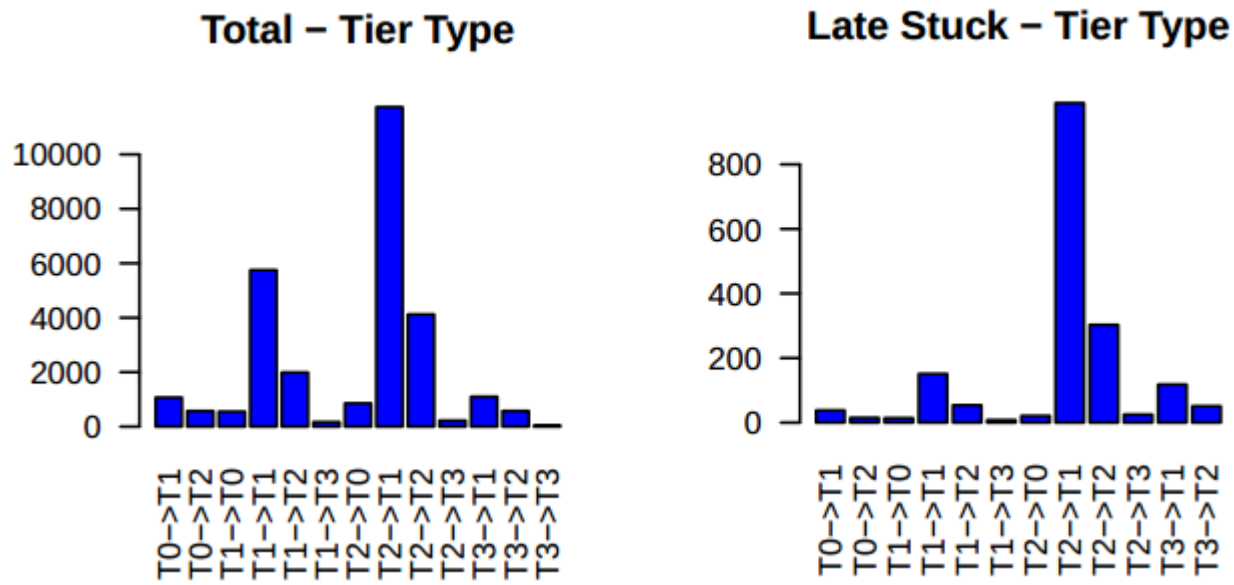

Figure 3. Block transfers counting, grouped by source and destination Tier type. For all transfers (left) and for transfers tagged as having latency tails according to eq. 5 (right). We can see that the contribution of Tier-0/1 sources is much lower in the sample with latency issues.

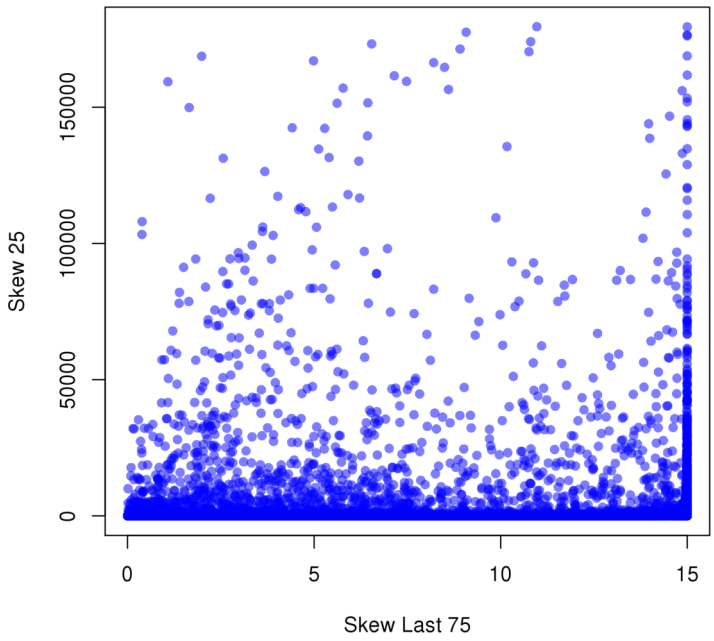

Figure 4. $Skew_{25}$ versus $SkewLast_{75}$ as defined in eq. 1 and eq. 2. For transfers tagged as being "early-stuck" according to eq. 6.

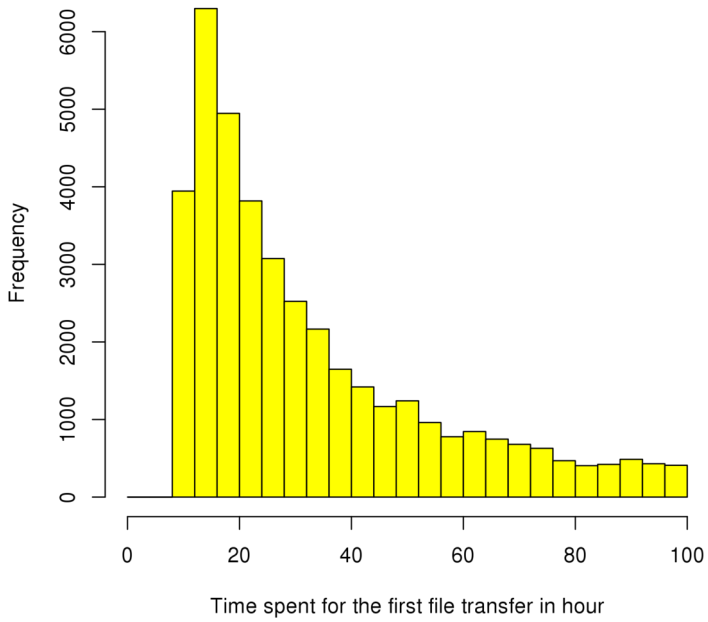

Figure 5. Frequency histogram of the first file replica time (in hour) for the transfers tagged as being "early-stuck" according to eq. 6.

this kind of issues as the sources of blocks are multiple and may be particularly heterogeneous. This is much similar to the latency issues described in sec. 5 but on dataset/fileblock level rather than on fileblock/file level.

As we said in sec. 4, the way our final set of latency data was selected does not allow us to study this kind of latencies. However, one way to detect these kind of latencies among the raw latency data provided by PhEDEx may be to select small group, aggregate information by dataset, calculate the transfer rate distribution of bocks within a dataset and look at the slow tails of such distributions. We leave this analysis for future work.

8. Conclusions

PhEDEx is one of the crucial components of the CMS Computing system. Profiting of data collection and filtering since Run-1, the CMS Computing project has a large set of data concerning the latencies as observed in all transfers happening between all Tiers. The study of this data set has just started, but it is already allowing a categorisation of the different root sources of such latencies and shaping different strategies to attach the observed problem, and this increase the overall performance of the PhEDEx system for CMS.

References

[1] CMS Collaboration, *"The CMS experiment at the CERN LHC"*, JINST **3** S08004 (2008)

[2] Grandi C, Stickland D and Taylor L 2005 The CMS Computing Model *CERN-LHCC-2004-35/G-083, CMS note 2004-031*

[3] M. Giffels, Y. Guo, V. Kuznetsov, N. Magini and T. Wildish, *"The CMS Data Management System"*, J. Phys.: Conf. Ser. 513 042052, 2014

[4] J. D. Shiers, *"The Worldwide LHC Computing Grid (worldwide LCG)"*, Computer Physics Communications 177 (2007) 219-223

[5] WLCG: http://lcg.web.cern.ch/lcg/

[6] T. Barrass, D. Bonacorsi, J. Hernandez, J. Rehn, L. Tuura, J. Wu, I. Semeniouk, *"PhEDEx high-throughput data transfer management system"*, CHEP06, Computing in High Energy and Nuclear Physics, T.I.F.R. Bombay, India, February 2006

[7] T. Barrass *et al*, *"Software agents in data and workflow management"*, Proc. CHEP04, Interlaken, 2004. See also http://www.pa.org

[8] R. Egeland *et al.*, *"Data transfer infrastructure for CMS data taking"*, XIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT'08), Erice, Italy, Nov 3-7, 2008 - Proceedings of Science, PoS (ACAT08) **033** (2008)

[9] L. Tuura *et al.*, *"Scaling CMS data transfer system for LHC start-up"*, presented at Computing in High Energy and Nuclear Physics (CHEP'07), Victoria, BC, Canada, September 2007 - J. Phys.: Conf. Ser. **119** 072030 (2008) doi: 10.1088/1742-6596/119/7/072030

[10] T Chwalek *et al.* 2012, *"No file left behind - monitoring transfer latencies in PhEDEx"*, J. Phys.: Conf. Ser. 396 032089 doi:10.1088/1742-6596/396/3/032089