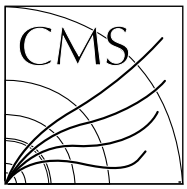
Available on CMS information server

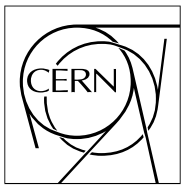
CMS NOTE 2007/016



The Compact Muon Solenoid Experiment

CMS Note

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland



July 2, 2007

Integration and operational experience in CMS Monte Carlo production in LCG

J. Caballero, P. Garcia-Abia, J.M. Hernández

Particle Physics Division, CIEMAT, Madrid, Spain

Abstract

This note describes integration and operational aspects of the CMS Monte Carlo production in the LHC Computing Grid (LCG). In 2005 the McRunjob MC production system was ported to LCG-2 in order to make use of the distributed computing and storage resources available in LCG for CMS. The full production chain (generation, simulation, digitization with pile-up, reconstruction, injection in the data transfer system and publication for analysis) was implemented. Experience gained during the implementation and operation of the production system in LCG has been used to build ProdAgent, the new MC production system. ProdAgent takes also advantage of the new CMS event data model, event processing framework and data management services. Integration and operational experience with ProdAgent is also described in this note.

1 Introduction

An efficient and performant Monte Carlo (MC) production system is crucial for delivering the large data samples of fully simulated and reconstructed events required for detector performance studies and physics analyses in CMS. The LHC Computing Grid (LCG) makes available a large amount of distributed computing, storage and network resources for data processing. While LCG provides the basic services and resources for distributed computing, reliability and stability, both in global Grid services and at the sites, are still key issues, given the complexity of services and heterogeneity of resources in the LCG Grid. A robust, scalable, automated and easy to maintain MC production system that can make an efficient use of the LCG resources is mandatory.

CMS has developed such a system during the last few years. We describe in this note the experience gained in integrating and operating the MC production system in LCG. We also briefly introduce the production frameworks and workflows in order to present the complexity of the system an its implications in the integration and operations tasks. In section 2 production in LCG with McRunjob, the production system until mid-2006, is summarized. An important pioneering work in integrating and running production in LCG was done porting McRunjob to LCG and establishing the full production chain. In section 3 we describe ProdAgent, the succesor of McRunjob as production system. ProdAgent has largely profited from the experience with McRunjob and has provided automation, fault tolerance, scale and efficiency in CMS MC production. The result has been an increase in about an order of magnitude in production yield. The new CMS event data model, processing framework and data management services, which have significantly simplified the production workflow, have also contributed to this increase in production scale.

2 Production with McRunjob

McRunjob [1] was until mid-2006 the framework for batch oriented large scale Monte Carlo production in CMS. It was designed to cope with multiple cooperating applications generating and processing a flow of data, to wrap these applications in a form by which they could be submitted as batch jobs, and to track or facilitate the tracking of these jobs. A modular architecture was adopted with customizable interfaces to allow for easy extension to new environments and applications. Each distinct application, database or external service was modeled internally as a Configurator that kept the relevant metadata describing the application parameters, results of database queries and service invocations. The Configurators were also responsible for generating jobs to achieve a specific data processing workflow. They were maintained in a container called the Linker that coordinated the actions of the Configurators, facilitated the communication among them and collected application specific job scripts into a unified linked job including all applications. Another Configurator abstracting the execution services could then submit the jobs. To drive the workflow, the user provides a script containing macros that are interpreted by the Linker into calls on the Linker and Configurator APIs.

McRunjob was originally employed in CMS for running large scale MC production in local farms. In order to utilize the large amount of computing, storage and network resources made available by the LHC Computing Grid (LCG [2]) McRunjob was extended to work in the LCG Grid environment. The initial implementation on LCG-1 [3], which included generation and detector simulation jobs, was extended to include the digitization with pile-up events and reconstruction processing steps as well as data harvest by the CMS data transfer system -PhEDEx [4]- and data publication for analysis [5]. In addition, as described in this note, the LCG implementation has been made more robust to cope with the inherent unreliability of the LCG distributed system resulting in a significant increase in performance.

ORCA [6] was until mid-2006 the CMS object-oriented application for digitization and reconstruction. It was based on the COBRA [7] framework as the interface for basic services. For a given dataset, in order to run the digitization step, ORCA requires COBRA event metadata describing the whole collection of input simulated events. When running reconstruction the metadata of both the simulated and the digitized events are required. The generation of metadata (the so-called metadata attachment) requires direct (posix-like I/O) access to the event data files.This operation is not suitable for distributed systems since the output of the jobs is potentially distributed among several Storage Elements with no POSIX-like I/O access from the metadata generating process. Therefore, metadata attachment was the main show-stopper for porting the MC production system to LCG beyond the generation and simulation steps. To solve this problem the concept of atomic attachment was introduced [5]. The metadata attachment is done on the Worker Node at run time only for the run to be processed. This way as soon as there are input data for a given step, the processing can start without waiting for the whole input collection to be completely available.

2.1 Framework and workflow

The basic unit in MC production is the dataset, a given physics process with a well defined set of parameters. The production chain starts with the generation of the events followed by detector simulation, mixing with minimum bias events to account for multiple interactions and pile up events, electronics response simulation (digitization) and reconstruction of physics objects. Each processing step is run separately in McRunjob and produces a different data tier (detector hits in the simulation, detector digis in the digitization and physics objects in DST format in the reconstruction). Each step is processed with a defined geometry and CMS software version. Detector and physics groups request a number of events of a specific dataset/processing step. For practical reasons requests are split into small assignments composed of runs (jobs of about 1000 events).

The generation jobs have no input and generate a small output (10 to 50 MB ntuples). They are, therefore, pure CPU jobs, taking few minutes to complete, although a few hours could be needed if hard filtering is present. The detector simulation jobs need the small input created during the generation step. They are CPU and memory intensive jobs, requiring 24 to 48 hours of processing, and generate a large output, $\sim$500 MB in three files (EVD files), where the smallest one is only $\sim$100 kB. The digitization jobs have lower CPU and memory requirements. Typically 5 to 10 hours are sufficient to carry out the work. However, they intensively perform read operations from the storage system, since continuous direct access to the PU samples through the LAN is needed. A large output is obtained, with similar size to the simulation one. Finally, the reconstruction jobs need even less CPU ($\sim$5 hours) and create a smaller output ($\sim$200 MB, in two EVD files).

Figure 1 shows the production workflow with McRunjob. For job creation McRunjob queries the CMS MC Reference Database (RefDB [8]). This global database is the source of production requests and configuration. It also acts as the event data bookkeeping system. McRunjob accesses RefDB via a web interface and downloads the list of runs, job templates, application data-cards, input file specification, etc. With these templates the application script is created. McRunjob reports back to RefDB about the data produced so that it can be used for further processing steps and for data discovery by physicists.

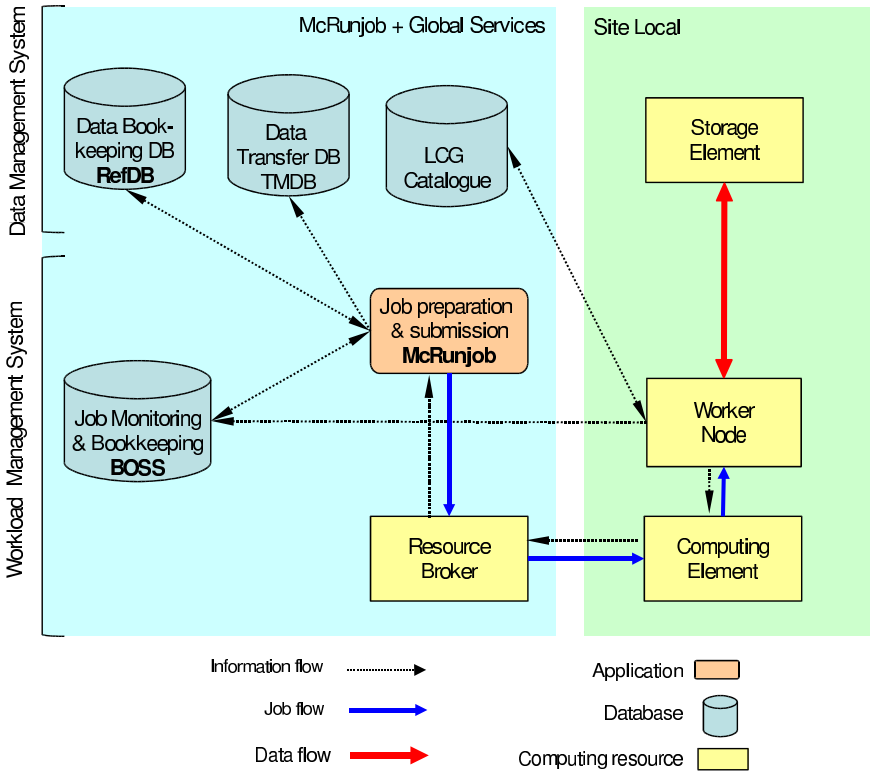


Figure 1: Production workflow with McRunjob in LCG

Jobs prepared by McRunjob based on specifications in RefDB are submitted to the Grid using the LCG Workload Management system (WMS). McRunjob is executed in a LCG User Interface (UI) machine with the WMS client applications installed. A file in JDL (Job Description Language) syntax describing the requirements of the job

(CPU, memory, ORCA software version, etc) together with all files needed to run the job are sent to a Resource Broker (RB) which submits the job to the appropriate site matching the requirements. The gateway for Grid jobs at a site is a Computing Element (CE) which submits jobs to the local batch system. The RB through the Grid information system has a global view of available sites with their CPU and storage resources. A Grid site is able to accept jobs only if the needed CMS software version is available. Production software is packed in standalone archives (the so-called DAR files) which contain all necessary to run CMS jobs included system libraries. Software is pre-installed at sites by the CMS software manager who submits Grid installation jobs. A software tag is published in the site Grid information system upon successful completion of the software installation job.

Jobs at run time at the Worker Node (WN) query the LCG global file catalogue to locate the Storage Element (SE) hosting the input data. For the input logical file names the LCG catalogue returns the storage URLs which are used by the LCG replication tools to download the input data into the WN. Event Metadata are generated at the WN for the input files before processing them. Atomic metadata attachment at the WN introduces a negligible overhead, since the needed EVD files have been already copied into the working area from the remote SE. Job output EVD files are uploaded into a destination SE (set in the job specifications) and registered in the LCG catalogue. When the job finishes, a summary file describing the produced data is retrieved by McRunjob via the RB. This file is uploaded in RefDB for production bookkeeping.

BOSS (Batch Object Submission System [9]) is used for job submission, monitoring and bookkeeping. A BOSS task running in parallel with the job collects job status information by parsing the process log file. This information is stored in the BOSS database, either remotely in real-time from the WN or locally at job completion time when the job BOSS summary information is retrieved via the RB together with the job summary file.

To make production data available for analysis, COBRA metadata have to be generated for the whole collection. The metadata generation process requires direct access to the event data. Thus, production files, potentially spread among many LCG sites, have to be harvested into a single site. McRunjob has been coupled to the CMS data transfer system PhEDEx in order to perform the data harvest automatically. Production files are made known to PhEDEx using the information in the summary files. A PhEDEx agent processes summary files, extracts the appropriate information (dataset name, logical file name, file size and checksum) and injects those attributes into the PhEDEx central transfer management database (TMDB). Files are atomically injected into PhEDEx on a per job basis as they become available from production in order to minimize the latency between production and transfer. PhEDEx transfers take place between nodes (sites) which are interconnected according to a certain transfer topology. A *virtual* PhEDEx node (comprising all LCG production sites) is used for collecting production data. Production files are assigned to the LCG PhEDEx node and an export agent provides in TMDB transfer URLs for production files. Transfer URLs are built getting from the central LCG catalogue the storage URL associated to a given logical filename. Physical File names (PFN) of downloaded files at a site are registered in a local PhEDEx catalogue.

For a given dataset, COBRA metadata have to be generated for each of the production steps at every site hosting the data for analysis. The McRunjob publication service (CMSglide [10]) produces COBRA metadata files by reading the event data files registered in the local PhEDEx catalogue at the data harvesting site. In the metadata generation process CMSglide uses dataset information from RefDB, generates local XML file catalogues and registers the dataset in the local data location database (PubDB [11]). Figure 2 shows the data harvest workflow in McRunjob as well as the metadata generation workflow. Analysis tools inspect RefDB/PubDB for data discovery (RefDB stores URLs of PubDBs) and submit jobs to the appropriate site hosting a given dataset. Analysis jobs use at runtime the XML file catalogues generated by CMSglide for accessing the data locally.

For the digitization step, signal events are normally mixed with minimum bias pile-up events and the resulting hit distribution is digitized. Pile-up samples are pre-located at the production sites with the corresponding XML catalogue placed in a standard directory location providing information about the path and protocol to read pileup events.

2.2 Operation and performance

After porting McRunjob to LCG-2, production operations using LCG resources started in mid-2005. Initially operations were driven from CIEMAT by about 1.5 FTE and other production operators joined the effort few months later. Initial productions were quite inefficient resulting in a low production yield. Several problems were identified: job submission rate was quite low due to the large size of the input sandbox and job retrieval time was quite high due to the large size of the output sandbox; local configuration problems at the sites (software installation problems, unstable access to the software via NFS) resulted in a high job failure rate; instability in LCG services

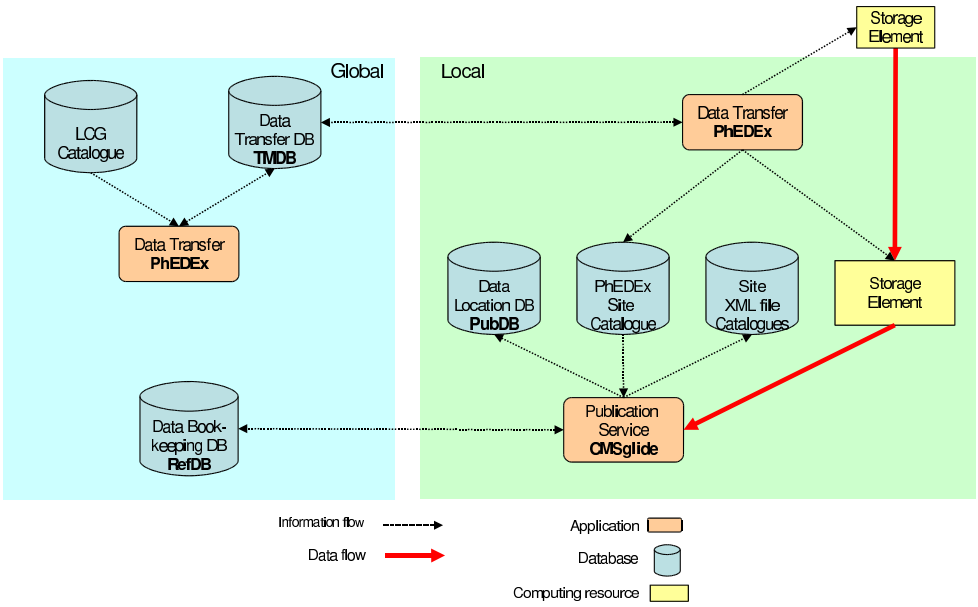


Figure 2: Data harvest and publication workflow in MC production in LCG

(RB, information system, central LCG catalogue) significantly affected job efficiency; the file stage-in/stage-out procedure was unreliable (instabilities in SEs, no retries, no fall back to file replicas at other sites, no automatic input data pre-stage from tape). In addition, operations were hard to conduct due to lack of proper monitoring and poor error report by the production framework. Typically log files had to be visually inspected in order to determine the cause of job failures. As a result, job re-submission was very hard to automate.

Several improvements were introduced in the production system in order to make it more robust. A significant improvement in the job submission and job retrieval rates was attained by largely reducing the size of the input and output sandboxes. Job metadata and XML catalogues were removed from the input sandbox while log files were removed from the output sandbox. Instead, those files were made available through a storage element. The stage-in/stage-out procedures were made more robust. Several retries with delays in between were introduced to copy from/to storage elements to cope with temporary unavailability of SEs or the global LCG catalogue. Wherever possible, pre-stage[1)] of input data from tape was manually run at the sites to reduce access latency to the data and avoid timeouts from the processing application when waiting for the data to be staged in. Fall-back SEs were used in case of stage-out failure of the output files. In order to reduce the number of copy operations per job, the output EVD files were packed together into an uncompressed zip archived which is stored in the SE. CMS job processing applications (including the metadata generation service) are able to read EVD files from inside zip archives directly from the SE without unpacking them. Packing files into a large archive has also the advantage of preventing storing small files into the storage systems. Zipping of EVDs was widely adopted later in CMS outside production. Having larger files greatly benefits data transfers, data archiving to tape and data staging to disk, as well as reduces the number of entries in the catalogues.

A dedicated LCG File Catalogue (LFC [12]) was setup for production purposes. This measure helped to reduce job failures when looking up in the catalogue for input files or registering output files. In order to reduce failures due to Grid Services or site problems, it was critical to strengthen the contact with site and grid administrators responsible of running the data and workload services. In order to cope with experiment software access or configuration problems the job wrapper was instrumented to install the CMS software in the working area of the job at run time. This operation causes little overhead compared to the typical long processing time of the jobs. It also allows to run CMS production jobs at opportunistic sites where CMS support is not available.

Table 1 shows job failure percentages due to various reasons before and after implementing the improvements described above. The total error rate significantly decreased from 34% to 8.5%.

[1)] In the CMS computing model, MC production is designed to run only at the Tier-2 sites, where the storage system is entirely based on disk storage. However, given the availability of CPU and storage resources at the Tier-1 centers in this phase of the experiment where no real data is yet available, MC production was also run at the Tier-1 sites, where data can be migrated to tape and deleted from disk.

Failure reason	Phase 1	Phase 2
Input data stage-in	4.8%	0.5%
Output data stage-out	7.4%	0.3%
Access to the LCG global catalogue	2.8%	1.5%
Experiment software configuration	10.1%	0.2%
Access to local data	8.3%	2.1%
Grid site configuration failures	0.3%	0.4%
Unclassified	0.3%	3.5%
Total	34.0%	8.5%

Table 1: Job failure rate before (Phase 1) and after (Phase 2) the implementation of improvements in the production system described in the text.

Figure 3 shows the number of trials needed to successfully complete a job, for different production steps (simulation, digitization and reconstruction) and for both production phases (before and after improvements in the production system). Job efficiencies got higher for all production steps in the second phase. In average the global job efficiency increased from 66% to 86%.

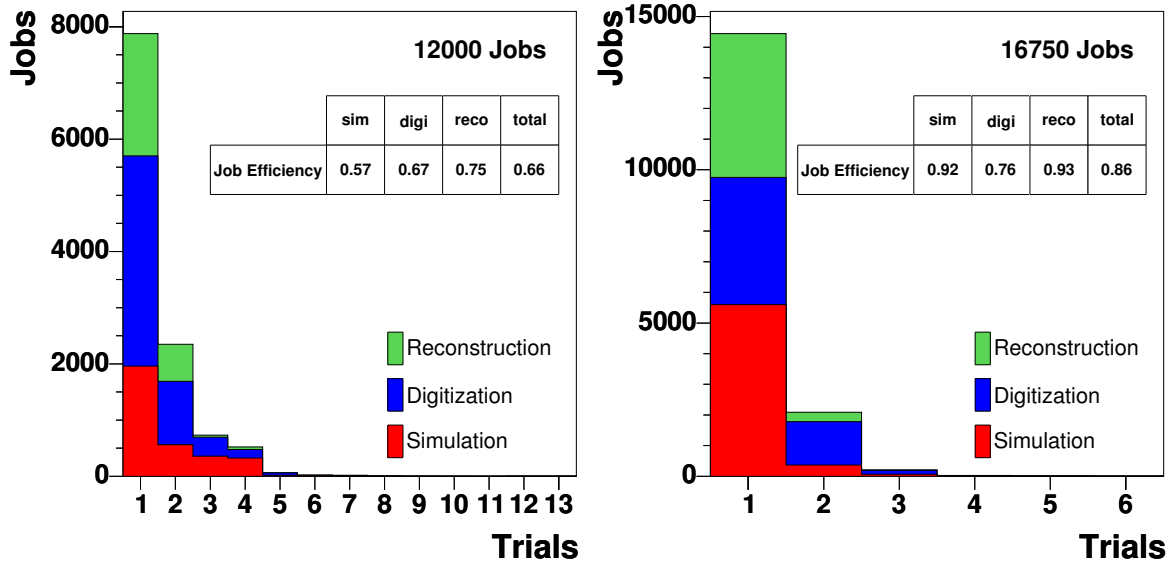


Figure 3: Distribution of trials needed to successfully complete a job before (left) and after (right) the implementation of improvements in the production system. The distribution is shown for different production steps and job efficiencies derived from the distributions are given in the table.

Production efficiency critically depends on the stability and reliability of LCG services (information system, resource broker, file catalogue, etc) and the workload and storage systems at the sites. Production was run using a white list of sites, in which size, robustness and responsiveness of local site administrators were the elements taken into account to include a site in the production white list. Figure 4 shows the distribution of jobs run at the sites used for production. Jobs were executed at 42 sites over 14 countries along the whole production period. About 90% of the jobs were run only at 13 sites over 5 countries. Digitization with pile-up processing was run in a smaller number of sites, those hosting the pile-up samples. Reconstruction was usually run at the sites hosting digitized data in order to reduce data transfers over the network. Only those sites with continuous high performance and low failure rate were massively used. A large fraction of the production was run at the Spanish sites where MC production operators had local access to the workload and storage systems making it easier and faster to detect problems and contact site admins to fix them.

The submission strategy was driven by data placement. Jobs were preferentially submitted to sites hosting the input data to be processed in order to minimize data transfer over the network. For digitization with pile-up processing jobs were restricted to run at the sites hosting the pile-up minimum bias sample. These sites publish a special tag in the Grid information system so that the Resource Broker submits digitization jobs to only those sites.

Figure 4: Distribution of jobs per site: total (upper left), simulation jobs (upper right), digitization jobs (lower left) and reconstruction jobs (lower right).

Figure 5 shows the number of successfully completed jobs per day (left) and the accumulated number of produced events (right), separately for simulation, digitization and reconstruction. In total, over 16 million simulated events, 11.5 million digitized events and 6.5 million reconstructed events were processed in LCG in essentially one year of production. Typically about few hundred jobs were completed per day with spikes of more than one thousand jobs. The production rate was not continuous due to several reasons: lack of requests by physics groups, temporary unavailability of central Grid services, delays waiting for a new release of the CMS software to become available, lack of automation of job submission, tracking and resubmission heavily relying on MC operator interventions, etc. It can be seen that at the end of the production period, after the implementation of various improvements in the production system, the production yield greatly increased producing 2.5 million events in one week.

Figure 6 shows an example of processing of one dataset through all production steps. The plot shows the number of jobs in execution in each time bin, in green those that completed successfully while those that eventually failed are plotted in red. Large spikes of failed jobs are due to problems in central grid services (typically the LCG central catalogue or the RB) or at sites (storage system). The overlap between the processing of the three production steps is small in this example due to the lack of automation in job submission when data from the previous processing step become available. The gaps between periods of job completion within a given processing step are also due to the manual job submission and non-optimal job distribution between sites.

Figure 7 shows the distribution of event processing times (left) and the distribution of event sizes (right) for the different productions steps. In average few minutes are needed to simulate one event while digitization and reconstruction usually takes less than a minute. The shortest event processing time corresponds to digitization without pile-up. Larger event sizes correspond to simulation jobs. Table 2 summarizes the relevant numbers for processing times and data sizes together with the total number of processed events.

Jobs that perform digitization with pile-up need to read from the storage system many minimum bias events per

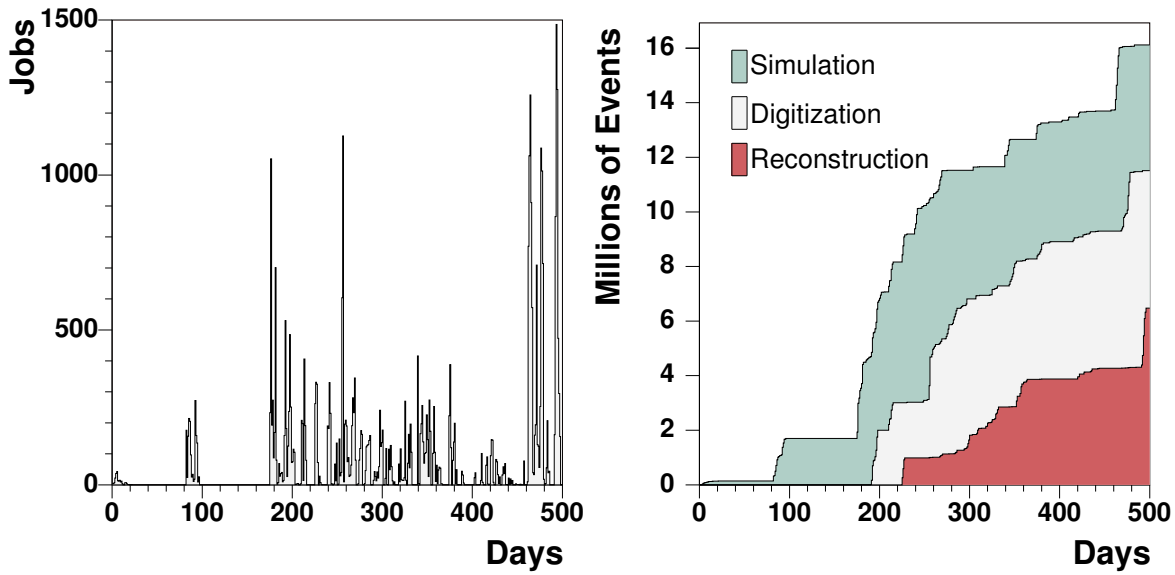


Figure 5: Number of successfully completed jobs per day (left) and accumulated number of simulated, digitized and reconstructed events (right).

	Simulation	Digitization	Reconstruction
Processed jobs	14900	14200	8900
Processed events	16 M	11.5 M	6.5 M
Processed events per day	44500	37000	23500
Mean time per job	28 h	13 h	8 h
Mean time per event	3 min	1.5 min	1 min
Mean size per job	360 MB	425 MB	260 MB
Mean size per event	475 kB	530 kB	380 kB

Table 2: Production yield, processing times and data sizes for the different production steps.

signal event. This poses a particular load in the storage system at some sites and jobs become I/O dominated. This effect can be seen in figure 8 left where the total job processing time over the job CPU time is plotted for different production steps. Digitization with pile-up jobs have a larger ratio indicating that the CPU is idle during a large fraction of the time reading pile-up events from storage. The effect depends on the performance of the storage system at the sites as can be seen in figure 8 right.

In order to improve the performance and reliability of digitization with pile-up processing and with the aim of extending digitization to sites not hosting pile-up samples, a strategy based on copying a fraction of the pile-up sample into the worker node was developed. At run time the job wrapper selects a random subset of the pile-up sample from the different replicas available at several sites. The size of the data subset is determined by the total number of events to be processed and the number of pile-up events per signal event required. Metadata generation for the selected pile-up subset is run at the worker node. This procedure was only implemented at the end of the production period and therefore was not widely tested at a large scale, but its feasibility was demonstrated.

Figure 9 shows the distribution of the time elapsed from job submission to job execution. When computing resources are available this time is dominated by the overhead introduced by the grid services for job submission (copy of the input sandbox, resource brokering, submission to the Computing Element of the selected site). Depending on the resource broker, this overhead varies from 30 seconds to about 100 seconds (see Fig. 9 right). The long tail in the distribution is due to jobs queuing at the local batch system at sites waiting for CPU resources to become available. The relatively small number of jobs sitting in this tail indicates that most of the time there were free CPU resources at the LCG sites running production.

Despite all deficiencies in the McRunjob framework for LCG and the unreliability of LCG services and sites, the pioneering work of extending the production framework to the LCG distributed system opened to CMS the possibility of using a large amount of computing resources for production [13]. It was also the seed for an improved

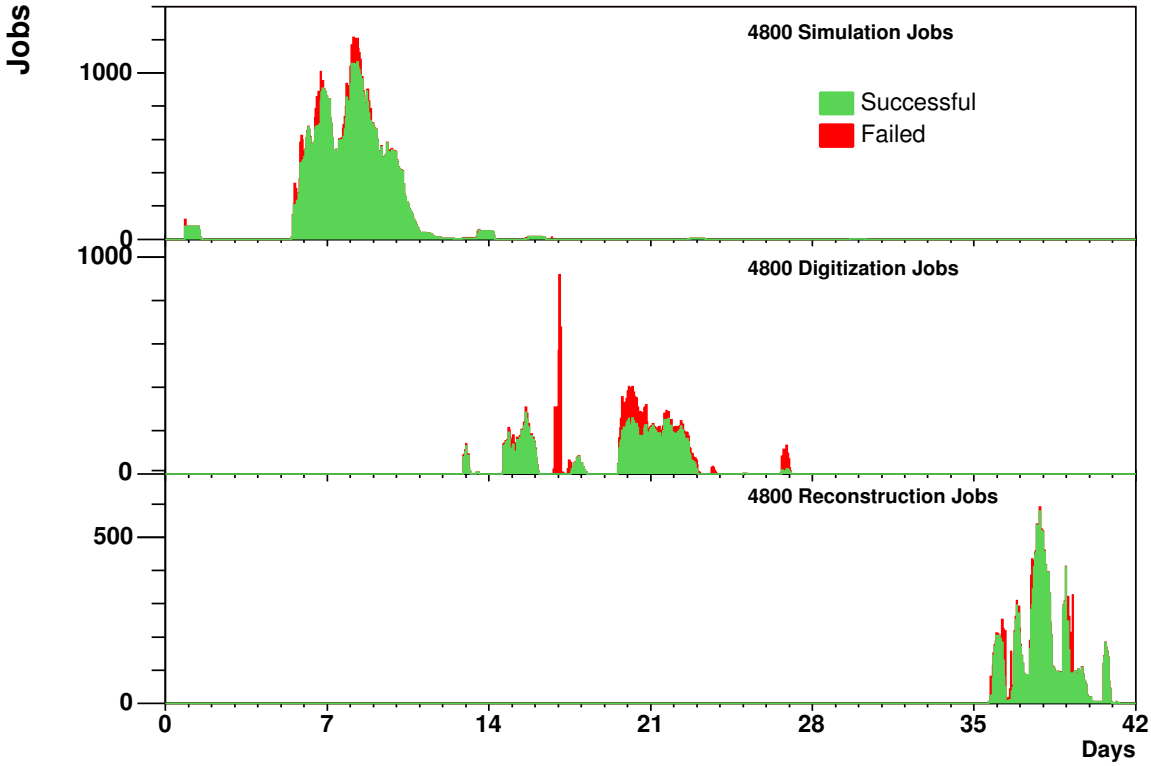


Figure 6: Example of processing of one dataset through all production steps.

Grid production system as described in the next section.

3 Production with ProdAgent

McRunjob production in LCG suffered from the lack of the proper automation, monitoring and error handling necessary to deal with processing in a distributed environment with inherent instability and unreliability. As a consequence the production system was not efficient and robust enough, did not scale beyond running about thousand jobs in parallel and was manpower-intensive. There was therefore the need for a new production system to automatically handle job preparation, submission, tracking, resubmission and data registration into the various data management databases (data bookkeeping, location and transfer systems). In addition, the new event data model and processing framework made it possible to simplify the quite complex processing and publication workflow in McRunjob. In particular, the lack of support for multi-step processing in a single job and the need of metadata generation in order to make the data available for analysis were two sources of big overhead. The implementation of a new production system was also an opportunity to incorporate into a new system the very valuable operational experience gained with McRunjob in LCG.

In 2006 the system for large scale MC production in CMS was changed. McRunjob was replaced by ProdAgent [14]. The new system was designed aiming at automation, ease of maintenance, scalability, avoid single points of failure and support multiple Grid systems. In addition, ProdAgent integrated the production system with the new CMS event data model, data management system and data processing framework.

The new processing framework, CMSSW [15], is centered around the concept of an Event. A data processing job is composed of a series of algorithms that run in a particular order. The algorithms only communicate via data stored in the Event. There is one executable and many plug-in modules which run the algorithms. The same executable is used for High-Level Trigger, simulation, reconstruction and analysis. The CMSSW executable is configured at run time by the user's job-specific configuration file. This file tells the executable which data to use, which modules to run, which parameter settings to use for each module, and in what order to run the modules. Required modules are dynamically loaded at the beginning of the job. The Framework provides ways to guarantee reproducibility by automatically maintaining and recording sufficient provenance information for all application results. The new data model and framework do not make use anymore of event metadata, unlike the old ones. This considerably

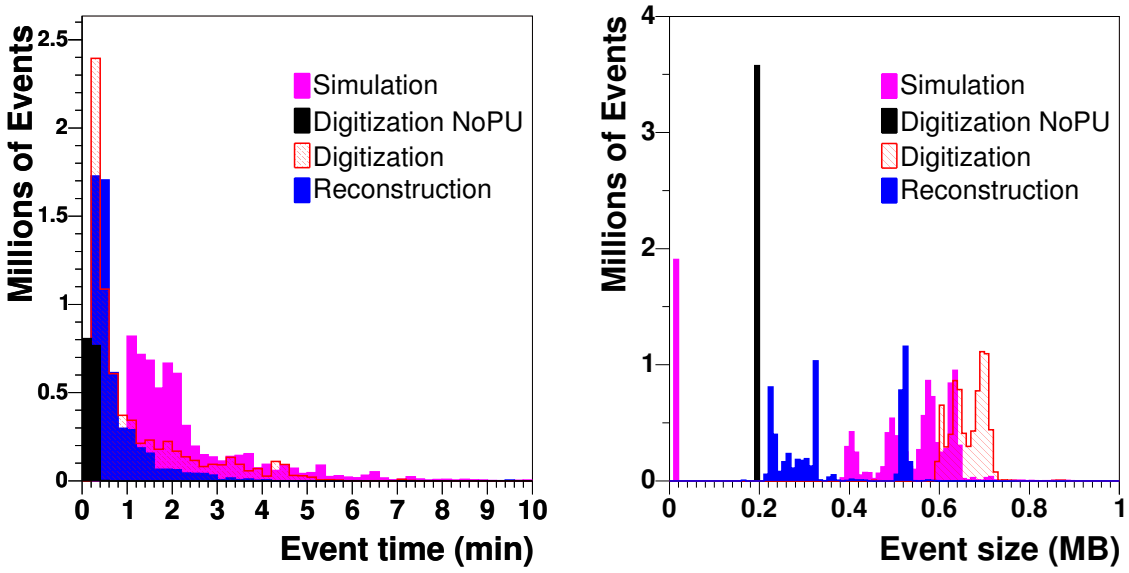


Figure 7: Event processing time (left) and event size (right) for simulation, digitization and reconstruction jobs.

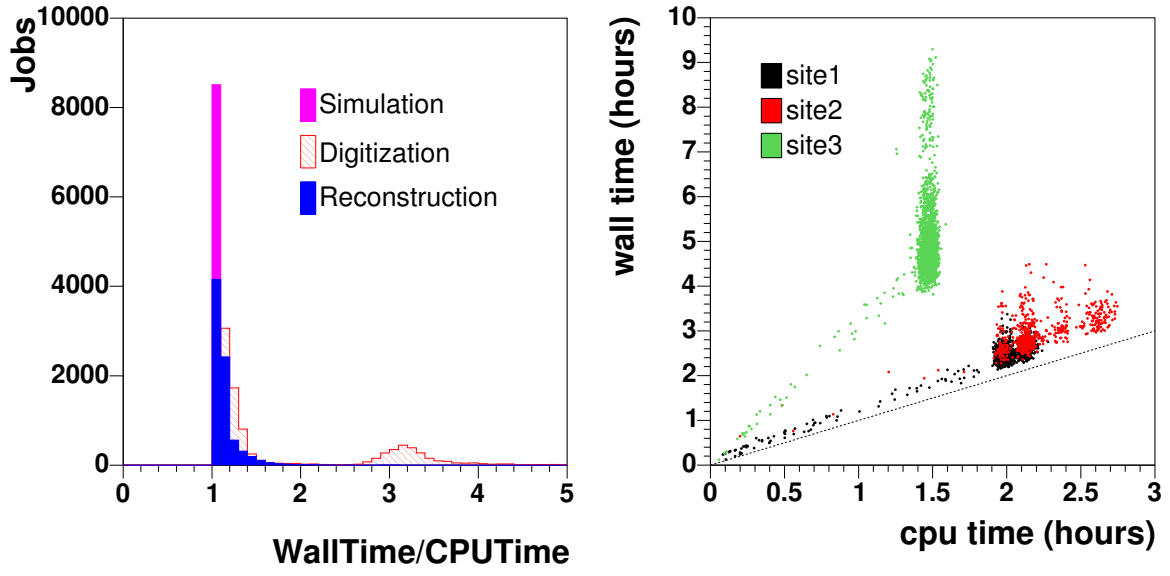


Figure 8: Total processing time over CPU time for different production steps (left) and total processing time versus CPU time for digitization with pile-up jobs run at three different sites.

eases the production workflow.

The new Data Management system (DMS) is based on a set of loosely coupled components that manage the large amounts of data produced, processed and analysed in the CMS distributed computing environment. The Dataset Bookkeeping System (DBS [16]) describes the existing data. It replaces RefDB in the old DMS. The Data Location Service (DLS [17]) tracks the location of the data. It replaces PubDB in the old system. Finally, local physical file names and access protocols at the sites are provided to the jobs through a local Trivial File Catalogue (TFC [18]), a set of logical to physical file name conversion rules. The TFC replaces the site XML POOL file catalogues in the old system. The new DMS introduces the concept of file block, a set of files which are tracked and replicated together.

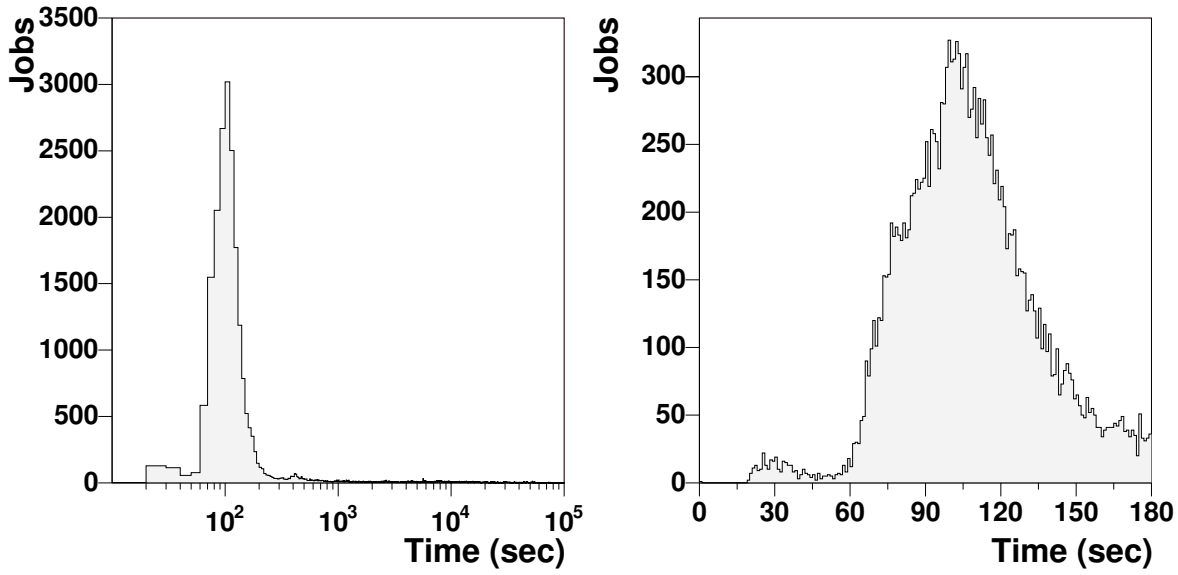


Figure 9: Distribution of the time elapsed from submission to job execution (left). Zoom of the interval (0,180 secs) (right).

3.1 Framework and workflow

ProdAgent is built as a set of loosely coupled components that cooperate to carry out production workflows. Components are python daemons that communicate through a mysql database. Components use an asynchronous subscribe/publish model for communication. Their states are persistently recorded in the database. Work is split into these atomic components that encapsulate specific functionalities.

Scaling is achieved by running any number of ProdAgent instances. Every instance makes use of a local database for operation and monitoring of the components as well as a local DBS/DLS instance for data bookkeeping. Produced data are published into the data transfer system database and into the global DBS/DLS instance to make them available for transfer and to the collaboration for analysis.

ProdAgent includes components for job creation, submission and tracking, error handling and job cleanup, data merging and publication into global DBS/DLS/data transfer system. The interface to the different Grid flavours is done via plug-in's for the job creation, submission and tracking components. It is fairly simple to add new systems and customization.

Each Job returns a detailed XML record of processing to the ProdAgent for accounting. It includes details of inputs, outputs, job information, any errors and diagnostics, etc. ProdAgent extends the job report with additional information like dataset, data tiers, site information, file checksums, etc. This report file is passed to components upon job completion for triggering data registration and data merge. The merge component triggers jobs based on the availability of data in the local DBS instance.

Figure 10 depicts the production workflow with ProdAgent. Processing jobs are sent to sites and store data at the local storage elements by means of the local TFC. Jobs report back to ProdAgent which triggers data merge jobs sent to the sites hosting the unmerged data. Job resubmission on error is automated. Data processing, bookkeeping, tracking and monitoring occurs in the local scope databases of the ProdAgent instance. After successful processing data bookkeeping and location information is promoted to the global scope databases and the data transfer system.

Compared to the workflow in McRunjob, there are significant differences. Jobs do not use anymore a specific software distribution for production (DAR files for McRunjob); they use the standard software repository also used by analysis jobs. There is no stage-in of input files into the WN's local disk but files are directly opened on the storage element via appropriate local posix-like IO access protocol (rfio, dcap). Output files are staged out into the local storage system; stage out into a remote SE is only done as fall-back when local stage out fails. Local storage systems implement a structured name space so that a simple TFC can be used; there is no need anymore for specific XML local file catalogues for each dataset. A global LCG catalogue is not used anymore for production; each ProdAgent use a local scope DBS/DLS instance for data bookkeeping and tracking. In ProdAgent there is an additional data merge step and a data information promotion step from local scope to the global DBS/DLS

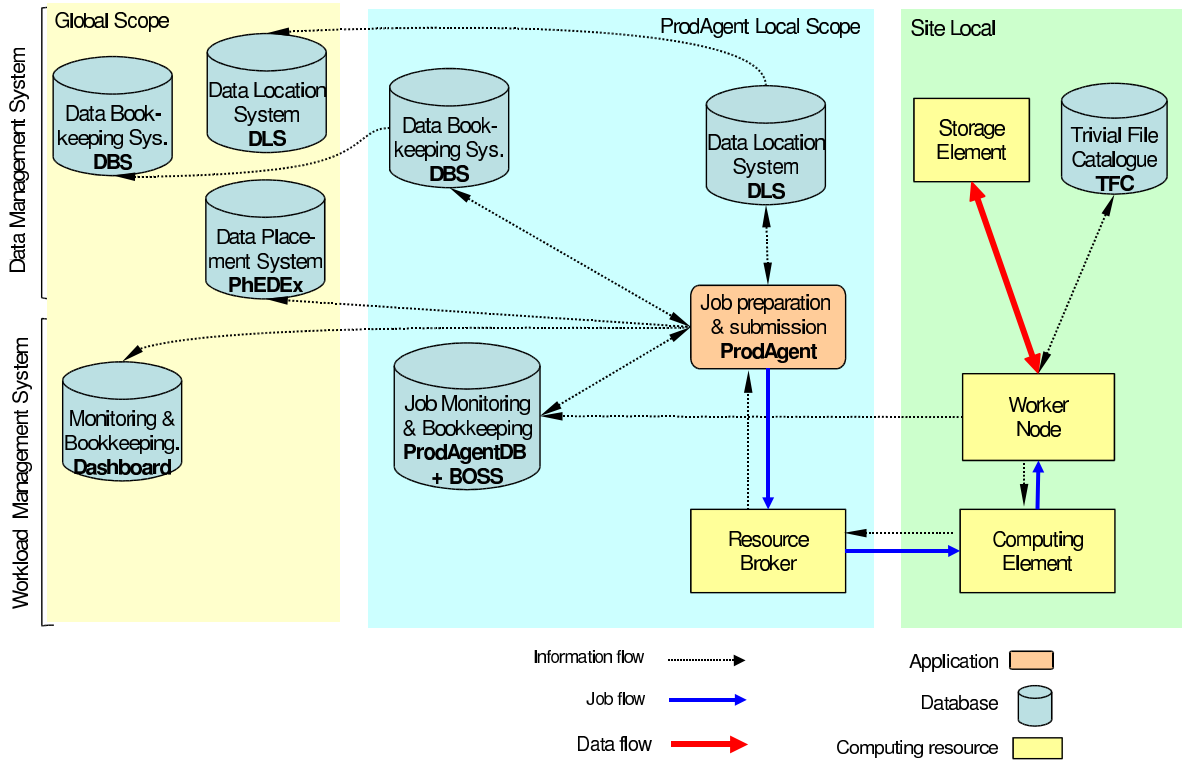


Figure 10: Production workflow with ProdAgent in LCG

instance. The complex data publication workflow in McRunjob is very much simplified, in particular, there is no need for metadata generation for each dataset (something that had to be done at every analysis site in the old EDM model). In ProdAgent no data harvest via a PhEDEx virtual node is needed. PhEDEx TMDB keeps a mapping between SEs and PhEDEx nodes so that during data registration in the transfer system data are associated to the PhEDEx nodes where they reside.

3.2 Operation and performance

The main operational reasons to migrate to a new production system were to incorporate a much higher level of automation and proper error handling in order to make a better use of the available resources for production and therefore to reach a larger production scale from few hundred CPUs to few thousand CPUs per production system instance. ProdAgent has successfully incorporated all those ingredients. Automation in job creation, submission, tracking, error handling, resubmission, data registration and bookkeeping is realized by means of dedicated component daemons in ProdAgent.

One can see in figure 11 the production scale reached by one ProdAgent instance during the summer 2006 production campaign in preparation for the CMS CSA06 computing challenge [19]. One ProdAgent instance can submit up to 3000 jobs per day, limited by the serial submission of jobs in the LCG workload management system. It takes 20-30 seconds to submit a job to a LCG RB. This includes authentication, upload of the input sandbox, interaction with RB network server, etc. Given a typical job duration of 12-24 hours corresponding to a 500-events job with a typical event processing time of 2-3 minutes, a ProdAgent instance can process up to 1.5 million events per day. The desired production scale is attained by running any number of parallel ProdAgent instances. Currently ProdAgent is in the process of implementing the bulk submission feature of the gLite [20] workload management system (the successor of LCG) to boost the number of jobs a ProdAgent instance can handle.

Production resources are distributed among production teams so that each team runs jobs at specific sites. This way each team is in charge of contacting specific site administrators to follow up potential problems with production at the sites. Only sites with a proved record of reliability and sufficient resources are included in the production team's white list. Jobs are submitted carrying a production extension in the Grid proxy which allows to run them at the sites under local production user IDs which can be prioritized at the local batch system with respect to other

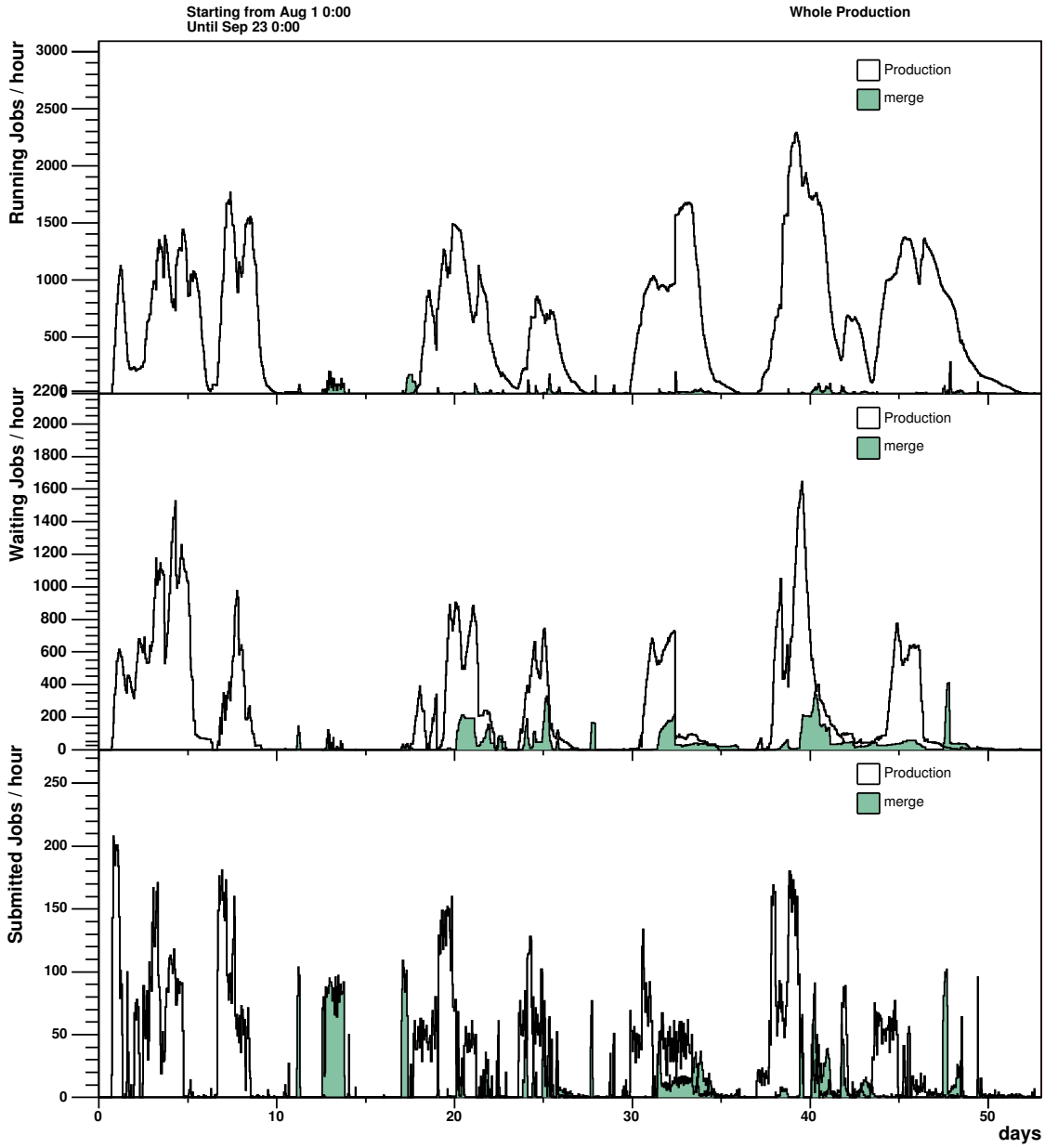


Figure 11: Number of jobs running (upper), queued at sites (middle) and submitted by a ProdAgent instance as a function of time for the summer 2006 MC production.

user jobs of the same virtual organization. When a processing step requires of an input dataset, jobs are sent to the sites hosting the input data. Unlike in McRunjob, the new production system cannot remotely download input data into the processing nodes. The locality of the Trivial File Catalogue prevents from remotely resolving the file SURL from the LFN. In case a given input dataset has to be processed at a different site than that hosting the data, the dataset has to be replicated by the data transfer system prior to submitting the processing jobs. In ProdAgent input files are directly read from the local storage system via the posix-like I/O implementations of the data access protocols of the local storage system. In McRunjob input files were downloaded into the local disk of the worker node. Compared to the old EDM used in McRunjob, the number of input files files for processing job is reduced from 3 EVD files and several metadata files to just one event data file. Output files are staged out into the local storage system of the processing site. As in McRunjob, for robustness the stage out is retried several times in case o failure and fall-back remote SE can be used if local stage out fails. The alternate remote storage URL to be used in case of stage out failure is configured in the local TFC at the sites.

Workload injection into ProdAgent is still a manual operation. For a given workflow (input and output datasets, number of events, software version, configuration, production steps, etc) the production operator injects into ProdAgent certain number of jobs appropriate for the number of CPUs available to the ProdAgent instance. The lack of automatic injection of jobs into the system translates in periods of time where available resources are not fully exploited, as can be seen in figure 11. Hence, job submission and number of running jobs were not uniform during the summer production. Currently ProdAgent is in the process of implementing a new component to queue jobs at ProdAgent level, monitor the available amount of resources and release jobs accordingly. This will lead to a much efficient use of the resources.

Figure 12 shows the accumulated number of events with time for processing and merge jobs for the ProdAgent instance run by the Spanish production team. Part of the unmerged sample could not be merged due to a disk loss in one of the sites. Figure 13 shows the distribution of produced events among the associated sites to the ProdAgent Spanish team instance. Above two thirds of the production was done at CERN and one quarter at the Spanish sites. The missing merged events in figure 12 stem from storage problems at IN2P3. Note that above-mentioned pictures represent only about one fifth (there were four additional production teams) of the production done over summer 2006.

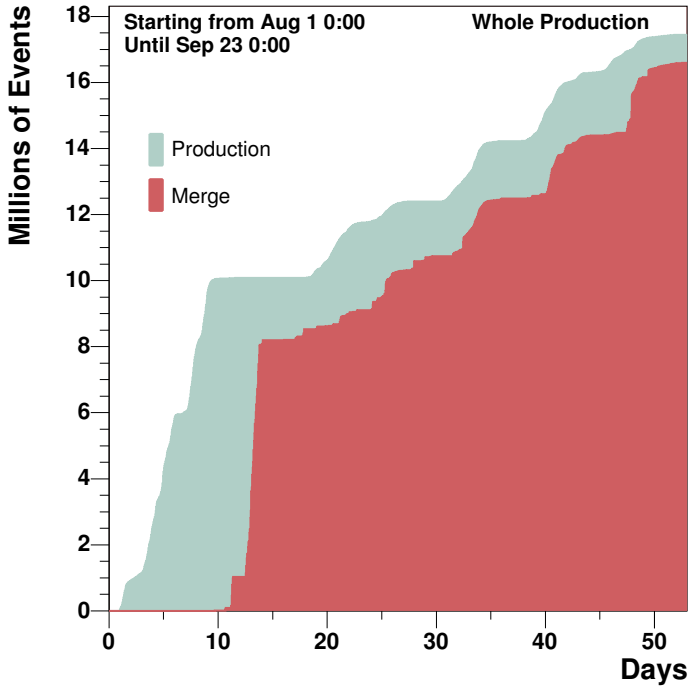


Figure 12: Accumulated number of produced events for processing and merge jobs.

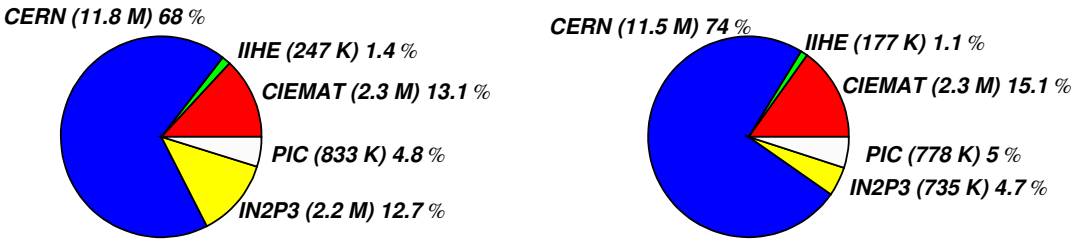


Figure 13: Distribution of produced events between the processing sites for unmerged (left) and merged data (right).

In figure 14 the event processing time for various datasets for processing (left) and merge jobs (right) is plotted. The processing of minimum bias event is typically 4 times faster than the processing of signal events. The same

applies for merge jobs since the minimum bias event size is 4 times smaller and the duration of merge jobs (I/O dominated) is proportional to the amount of data to read. The two distinct peaks in the minimum bias sample for merge jobs correspond to the different performance of the storage system at two different sites.

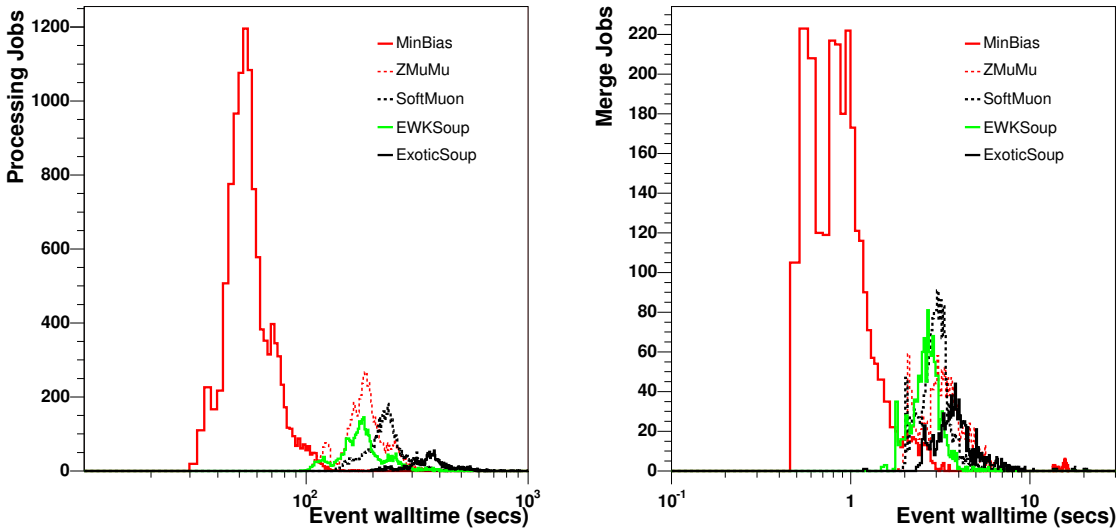


Figure 14: Wallclock event processing time for various datasets for processing jobs (left) and merge jobs (right).

Table 3 shows job failure percentages (relative to all submitted jobs) for processing and merge jobs for different failure reasons. The largest number of failures correspond to application failures and Grid inefficiencies. Application failures include failures reading the input data. In fact, there is an increased number of application failures for merge jobs (10.6% vs 5.5%), which access a much larger number of input files than the processing jobs. Grid related failures include failures in global Grid services (RB, information system) and failures of Grid services at the sites (CE, WNs). Grid job failures typically occur when the exit code of the job cannot be communicated back to the RB (e.g. jobs that get killed due to hardware failures at the WNs or killed by the batch system after expiring the allocated time slot) or at job submission time when the destination site temporarily disappears from the Grid information system. This last reason explains the increased Grid failure rate for merge jobs. These jobs are submitted to a specific site (the site hosting the unmerged input data) while processing jobs can be run at any of the sites in the white list. If a particular site temporarily disappears from the the information system, merged jobs submitted to that site will fail while processing jobs will run in any of the remaining sites. Failures in storing the output file into the local storage account for about 4% of the errors. Around 2% of the jobs fail when accessing the experimental software from a shared repository (typically via NFS).

Failure reason	Processing jobs	Merge jobs	All jobs
Application failure (including data access)	5.5%	10.6%	6.6%
Output data stage-out	4.0%	3.2%	3.8%
Experiment software configuration/access	2.6%	1.3%	2.3%
Grid related failures	5.2%	11.0%	6.7%
Total	17.3%	26.1%	19.4%

Table 3: Job failure rate for processing and merge jobs.

In figure 15 the number of trials for a job to succeed is plotted for production and merge jobs. The efficiency for production jobs is 83% and 74% for merge jobs, yielding an overall efficiency of about 80%.

One important feature of the new CMS event processing framework is that it can chain production steps. Typically either production is run in one step (generation, simulation, digitization with or without pile-up events, and reconstruction in one single job) or in two steps (generation plus simulation followed by digitization plus reconstruction in a different job and possibly with different CMSSW version). With the old processing framework each production step had to be run independently. The digitization with pile-up step was heavily dominated by I/O posing problems to the storage system at some sites when a large number of jobs were run simultaneously. In the

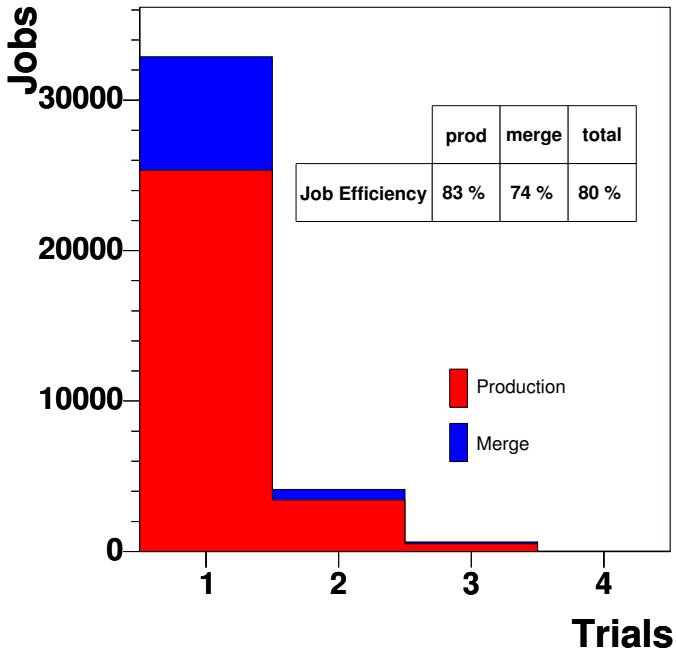


Figure 15: Distribution of trials required for a job to succeed, red for production jobs and blue for merge jobs.

new framework digitization is normally combined with reconstruction which increases the overall event processing time, reduces the rate at which events are read from storage resulting in an increased job efficiency. Pile-up samples are pre-located at the sites and digitization jobs are restricted to run only at those sites.

Efficient site support is still critical for production efficiency. Most job failures are related to site problems (temporary glitches in site Grid services, problems reading and storing data in the local storage system, local batch system and WN misbehaviour, access to experimental software). A fast reaction of site admins is required to minimize production inefficiencies. For example, a misconfigured WN where all jobs crash can behave as a 'black hole' attracting and crashing new jobs. The need to follow up problems at sites together with the limited production scale than can be reached with a single ProdAgent instance are the main reasons for the existence of several production teams. The required manpower, about 6 FTE, to run production is expected to be reduced as sites and Grid services become more reliable and the production system is upgraded with job bulk operations so that a single instance is able to handle a much larger number of jobs. With the aim of increasing site availability, the LCG central operations team periodically monitors the correct functioning of sites via the submission of grid jobs which probe general and VO-specific Grid services and functionalities [21]. It is hoped that the automatic notification of site administrators in case of problems will contribute to an increase of production efficiency and a decrease of the manpower required for production.

3.3 Further development

There is further development planned or going on in order to enhance automation, robustness, performance and scale in the production system reducing at the same time the manpower required for operation.

Block management (automatic block closure when a block reaches a size or a given number of files), migration from local to global DBS/DLS instances and injection into the transfer system have been recently automated. Multi-step processing (processing of several workflows where the input data of a workflow is the output of another) will be automated by introducing a new sensor component which polls DBS for available merged input data to trigger their processing. Workflow management is being automated by means of two new components, the ProdRequest system to automate workflow creation by physics groups and operation managers, and the ProdManager component to supply work (number of events to process, priority, etc) to the ProdAgent based on workflows entered in the ProdRequest system. At the moment workflows are manually given to ProdAgent by operators. Workflow termination will be automated (killing remaining jobs and cleaning them up).

Concerning scale issues, the number of jobs a ProdAgent instance can handle will be largely increased by in-

troducing bulk operations for job creation, submission and tracking. Available CPU slots for production will be more efficiently used by introducing a new resource monitoring component. This component will watch for available resources (querying the Grid information system or ProdAgent JobTracking) and will release jobs previously injected and queued at ProdAgent level.

In order to avoid that production jobs are killed by the local batch system when they exhaust the maximum time allocated to the job, it is planned to move from event-based jobs to time-based jobs. Jobs will not be prepared to process a given number of events but to last for certain amount of time, adjusted to the queue length at sites. Other situations of inefficient use of CPU resources have been identified. Sometimes jobs hang up at the sites, for example with access problems to input data. These jobs block batch slots for the duration of the slot not doing any processing. These kind of jobs could be easily identified and killed at the local batch system level by comparing the elapsed time with the CPU time. ProdAgent could also time out jobs which are in the running state for much longer than expected. Another reason of inefficiency are misconfigured worker nodes which immediately crash jobs thus acting as 'black holes' attracting new jobs which also crash and so on. This situation is difficult to catch and avoid in real time. Bad nodes are normally detected by ProdAgent monitoring system and sites are reported about the problem.

The global monitoring and accounting system for production is being extended by means of a new monitoring component in ProdAgent. This component of every ProdAgent instance will periodically send to a global server in a reliable way statistics of completed jobs that can be later analyzed providing a valuable information for optimization of the production system. This extension will complement the current global monitoring system based on individual job reports from the worker nodes sent to a global dashboard via a UDP-based messaging system.

4 Summary and Conclusions

During 2005 an end-to-end system to run Monte Carlo production in the LCG distributed computing environment was implemented within the McRunjob framework. It meant a big step forward with respect to the previous LCG implementation, where only generation and simulation processing steps were supported. The new production system made possible the full production workflow, from the generation of events to the publication of data for analysis.

McRunjob production in LCG suffered from the unreliability and instability of the LCG distributed environment and from the complex workflow imposed by the CMS event data model and the event processing framework. In particular the processing limited to a single step and the requirement of metadata generation between processing steps and for final data publication caused a serious overhead in production operations. Several improvements were incorporated in McRunjob for LCG processing, like robustness in data stage-in/stage-out, dedicated central replica catalogue, etc, in order to deal with the inherent Grid unreliability. In addition, a white list strategy for production sites, based on stability, site administrator responsiveness and size, was followed. That resulted in an increased production efficiency reaching a production scale of about 1000 concurrent jobs.

In 2006 the production system was refactored to adapt itself to the new CMS event data model, processing framework and data management system. The new production system, ProdAgent, incorporated the pioneering work and valuable experience gained in implementing and operating production in LCG. The lack of automation in job handling, proper monitoring and error handling in McRunjob in LCG made production a manpower-intensive task and resulted in a limited production scale and an inneficient use of the available computing resources.

ProdAgent brought to the production system automation, scalability, robustness and efficiency. Through a set of loosely coupled components production workflows are carried out in an automated fashion. Data processing and management is optimized in ProdAgent by running multiple processing steps in a single job and by incorporating a data merging step in the production workflow. ProdAgent is fully coupled to the various data management system components. Each ProdAgent instance runs its own local data and job tracking systems. Produced data are then promoted to the global data management databases, including the data transfer system. The new system brings scalability by the possibility of running any number of ProdAgent instances in parallel. Each instance can currently handle few thousand concurrent jobs. A production scale of more than 10000 concurrent jobs has been reached by running several ProdAgent instances in parallel.

Robustness is still an important issue in the production system given the current unreliability and instability of global Grid services and sites (local storage and batch systems) or even unavailability for an extended period. Responsiveness of Grid and site administrators is crucial for an efficient production. The overall job efficiency is

currently about 80%.

Running production in LCG is still a manpower-consuming task. Further automation is being implemented in ProdAgent. New components will take care of continuous job injection based on available computing resources or available data to be processed. Workflow management will be further automated coupling ProdAgent with a production manager component which in turn gets production work to be done from a production request system. Bulk operations will allow to increase the scale of jobs a single production instance can handle. Time-based jobs will allow to use more efficiently the computing resources. An extended global monitoring and accounting system will allow to analyze production performance and usage of resources for further optimization.

Acknowledgments

Many people have contributed to the development, integration and operation of the CMS MC production system. We are specially indebted to D. Evans, A. Fanfani, T. Wildish, L. Barone, V. Sekhri, G. Graham and M.A. Afaq. We also thank the responsiveness of the CMS contacts and site administrators at the production sites.

References

[1] Runjob Project, http://projects.fnal.gov/runjob/

[2] LHC Computing Grid, http://cern.ch/LCG/

[3] P. Capiluppi et al., CMS Results of Grid-Related Activities Using the Early Deployed LCG Implementations. CMS NOTE-2004/034.

[4] J. Rehns et al., PhEDEx high-throughput data transfer management system. Proceedings of the CHEP06 Conference. Mumbai, India, February 2006.

[5] P. Garcia-Abia, J.M. Hernández, Implementation of Monte Carlo Production in LCG-2. CMS-NOTE-2005/019.

[6] ORCA, Object-oriented Reconstruction for CMS Analysis, http://cmsdoc.cern.ch/orca/

[7] COBRA, Coherent Object-oriented Base for Reconstruction, Analysis and Simulation, http://cobra.web.cern.ch/cobra/

[8] Reference Database for CMS Monte Carlo production, https://twiki.cern.ch/twiki/bin/view/CMS/RefDB

[9] BOSS, Batch Object Submission System, http://boss.bo.infn.it/

[10] M.A. Afaq, CMS Publish Service CMSGLIDE, http://lynx.fnal.gov/runjob/Setup_20Publish_20Service

[11] PubDB, CMS Database for dataset publication, http://cmsdoc.cern.ch/cms/cpt/Computing/Technical/subproj/PubDB.html.

[12] LFC, LCG File Catalogue, https://uimon.cern.ch/twiki/bin/view/LCG/LfcAdminGuide

[13] J. Caballero, P. Garcia-Abia y J.M. Hernández, CMS Monte Carlo Production in the LHC Computing Grid. Proceedings of the CHEP06 Conference. Mumbai, India, February 2006.

[14] ProdAgent, CMS production Agent https://twiki.cern.ch/twiki/bin/view/CMS/ProdAgent

[15] C. Jones et al., The new CMS Event Data Model and Framework. Proceedings of the CHEP06 Conference. Mumbai, India, February 2006.

[16] DBS, CMS Dataset Bookkeeping System, https://twiki.cern.ch/twiki/bin/view/CMS/DBS-TDR

[17] DLS, CMS Data Location Service, https://twiki.cern.ch/twiki/bin/view/CMS/DLS

[18] TFC, CMS Trivial File Catalogue, https://twiki.cern.ch/twiki/bin/view/CMS/SWIntTrivial

[19] CSA06, CMS Computing, Software and Analysis challenge 2006, CERN/LHCC 2007-010, CMS NOTE 2007/006.

[20] gLite, EGEE middleware for Grid Computing, http://glite.web.cern.ch/glite/

[21] SAM, Site Availability Monitoring, https://lcg-sam.cern.ch:8443/sam/sam.py