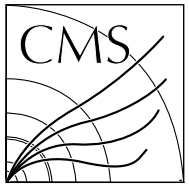
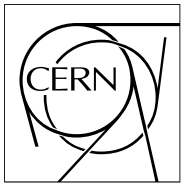


The Compact Muon Solenoid Experiment

CMS Note

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



13th September 2002

The Spring 2002 DAQ TDR Production

The CMS Production Team

Edited by Veronique Lefebure[a] ***and Tony Wildish***[b]

Abstract

In Spring 2002 the CMS Production team produced a large sample of Monte Carlo events for the CMS DAQ TDR. This note documents the process by which those events were produced, with details of the tools, the architecture of the production machinery, and the individual sites that took part.

Preliminary version

[a] CERN-HIP

[b] Princeton University

1 Introduction

In Spring 2002 the CMS production team completed a production of Monte Carlo data for the DAQ TDR. Almost 6 million events were simulated with CMSIM [1] and then digitised with ORCA [2] under a variety of conditions. About 20 regional centres (RCs) participated in this work, with about 30 people being actively involved. This note describes in depth the way this data was produced, the tools developed for this production, and the problems and features of the production.

Compared to previous years there are a number of new features:

- The use of BOSS [3] for local job-submission and job-tracking. This was successful enough that it was obligatory to use it, although it did mean that regional centres had to gain expertise with MySQL.
- The use of DAR [4] for distributing CMS software. Again, this was obligatory, but since it was easier to use it than to not use it, nobody objected!
- The use of RefDB [5] for parameter-tracking and overall coordination, greatly reducing the possibilities for operator-error. The statistical information in the BOSS and RefDB tables made possible the analysis of the production process presented in this note.
- The PRS groups typically run a production-style first pass over the data, to produce ntuples or root-trees or to filter according to trigger conditions. Many of these 'filters' were absorbed into the production machinery and run by the production team, thereby saving time for the PRS groups. This activity is ongoing, with new versions of ntuple-makers etc being released by the PRS groups as their code evolves.
- Several small samples (about 25 assignments of up to 50K events each, 0.5 million events in all) were produced using various pre-release versions of ORCA to help debugging of CMSIM 125 and ORCA 6.0.0. This also helped the production team to commission new sites and gain experience with the new tools we used.
- The production team now comprises over 20 sites, so releasing the production tools (BOSS, IMPALA [6]) needs some coordination. Production farms vary widely in nature such that it is almost impossible to release a new version and have it work everywhere first time, no matter how much local testing you do. We addressed this by introducing beta-testing of our tools at only a few chosen sites before releasing them to everyone.
- Analysis is becoming less centralised, as many regional centres now have the capacity to host analysis too. Analysis is currently being carried out at Bologna, Bristol/RAL, Caltech, CERN, Fermilab, Legnaro (INFN), Moscow, and (soon) Florida. This often involves a considerable flow of data back to the analysis centres from the production centres, so file-transfer (dataset-transfer) is becoming an important aspect of analysis.
- The amount of data imported to CERN or to Fermilab is as much or more than the data produced at each site. Clearly we are producing important volumes of data at the T2 centres.

The numbers of digitised events processed (as of mid-July) is as follows.

- no pileup ('NoPU'), 2.5 million events.
- low luminosity pileup (2×10^{33}), 4.4 million events.
- high luminosity pileup (10^{34}), 3.8 million events.
- PRS-filters, 2.9 million events.

These events occupy about 500 Objectivity 'datasets', with the datasets ranging in size from a few GB up to over 700 GB in size. These datasets comprise about 150 different physics channels digitised at different luminosities.

The data in Objectivity amounts to about 20 TB. Of this, over half (over 12TB) has travelled over the wide-area network to one regional centre or another. Most of the transfers were to CERN (7 TB) or to Fermilab (5 TB), but other site-to-site transfers occurred too, notably internally to INFN.

The sites that were involved in this round of production are: Bristol/RAL, Caltech, CERN, Fermilab, Florida, IN2P3, INFN (Bari, Bologna, Catania, Firenze, Legnaro, Padova, Perugia, Pisa, Roma, Torino), Imperial College, Moscow (ITEP, JINR, SINP MSU, IHEP), UCSD, Wisconsin.

Some of these sites are new to CMS productions, this being the first time they participate, and were commissioned during the course of the production. Other sites have over two years experience. It is worth noting that the scale of CMS productions is now such that data was produced at a constant rate throughout the production period (4 months of simulation and 2 of digitisation, see figs 4, 5 and 6). Given that many farms had significant problems at one time or another throughout this period this demonstrates that the CMS Production environment is now large enough to survive the (temporary!) loss of one or two farms. Of course, production on this scale brings its own problems, such as how to provide front-line support for this many regional centres.

By way of comparison, in Autumn 1999 CMS produced just under 1TB of Objectivity data in one month. The Spring 2002 production produced 20 TB in 2 months; a full order of magnitude higher rate, less than three years later. However, to make the comparison fair, the 1999 production was a one-man operation, the Spring 2002 production took the combined effort of about 30 people!

2 The CMS Production Process

2.1 Architecture

The CMS Production machinery architecture was presented in some detail in the CMS Grid Production-Tools workshop in July 2002 [7]. Here we recall only the outline, and refer you to the talks at that meeting for more details.

The main software components of the Spring02 Production were IMPALA [6], BOSS [3], RefDB [5] and DAR [4], plus the so-called "Tony's scripts' [12] for the transfer of the data.

The data transfer aspect is discussed in section 3.3.4.

Figure 2.1 gives a schematic overview of the production components.

RefDB is an SQL database located at CERN, where the physicists register their production requests via a web interface provided on the *cmsdoc.cern.ch* server. Via html forms, they define and record all process details: the name of the data to be produced, the application version and executable name, the list of input parameters. They also specify where and when the data should be delivered. Requests were then assigned, by the production coordinator, to a Regional Centre. The corresponding AssignmentID was communicated via email and web information, to the RC production staff who used it as input to the IMPALA production component.

IMPALA is a set of scripts that takes an AssignmentID as input argument, contacts RefDB in order to retrieve all request parameters, and creates the job scripts that need then to be submitted to a job scheduler. It uses a system of local tracking files for the discovery of the input data and for tracing the progress of the jobs submission. Each production site had at least one installation of the IMPALA software and had to configure it for their executable location, output data destination, BOSS installation (see more about BOSS below).

The physics executables were distributed beforehand to the RCs in tar files provided by the DAR tool. DAR is a tool developed at FNAL by Natalia Ratnikova, that creates a tar file containing the binaries and possibly some data files needed for processing data. The DAR files were created at FNAL by Natalia for each CMSIM and ORCA version. The DAR file was then made available both at CERN and at FNAL for distribution to the RCs. During Spring02 production, four DAR files were needed: *Pyt6158_Sim125.tar*, *Pyt6158_Sim125.1.tar*, *ORCA_6_0_2_dar*, and *ORCA_6_0_1_dar*. Coordination was needed for the preparation of the DAR file, because the name of the executables had to correspond to the designations used in RefDB and transmitted to IMPALA.

BOSS is both a job-tracking system and a generic interface to multiple schedulers. It uses an SQL database that each RC had to install locally, that keeps track of all job submissions and that was used for monitoring the job progress as well as to record the job output details (such as input/output run number, output file name, number of processed events, job status, etc). The BOSS schema and monitoring scripts were part of the IMPALA software. Part of the schema is BOSS-specific (or rather, experiment-neutral), the rest is highly dependant on the type of job, and therefore of the experiment. When all jobs of a given AssignmentID were completed, the production staff was requested to run an IMPALA summary script that would read from the BOSS tables what had been produced and send the information to RefDB for the central book-keeping. These summary scripts were also checking the consistency between the information provided by BOSS on one side, and the content of the produced META data which were stored in Objectivity database files together with the actual data (see section 2.2).

This natural division between site-local resources (the BOSS database) and CMS-wide resources (RefDB) meant that no single batch job anywhere in CMS depended on the availability of a central resource. The coupling between

site-local databases and RefDB (the only CMS-wide production database) always happened outside batch jobs.

2.2 CMS Monte Carlo Production

During the Spring02 production, the Monte Carlo production chain included all calculation steps from the generation of the primary physics interaction in the beam-pipe to the simulation of the detector response, called 'digitisation' and some Level-1 trigger filtering. Track reconstruction and particle identification was not part of the official production chain.

In more details, the data production steps are the following:

1. the Generation of the primary physics proton-proton interaction for a given physics channel.
2. the Simulation of the secondary physics interactions: i.e. how the particles produced in the primary interaction fly into the detector and interact with the material and with the magnetic field.
3. the Digitisation, i.e. what kind of electrical signal is produced by a sensitive element of the detector that is crossed by one or more particles.
4. the Level-1 trigger filtering, i.e. which of the signals generated during the Digitisation step pass the thresholds defined by the Level-1 trigger logics.

In the CMS production jargon, the Generation step is also called 'PYTHIA' [8] or 'CMKIN' step; the Simulation can be called the 'CMSIM' step.

The output data of the simulation step are CMSIM .fz files (ZEBRA files). They need to be reformatted into Objects in Objectivity/DB files in order to be used as input by the COBRA/ORCA reconstruction software that performs the Digitisation step and further analysis (Filtering and more). This data translation is called the 'ooHit Formatting' step.

To each step of the production chain and for each dataset (physics channel), corresponds one Assignment as defined in RefDB (see more details in reference [5]).

Digitisation can be performed for different beam luminosities, taking into account or not, the other 'Minimum Bias' proton-proton interactions that may occur during the same beam-crossing as the one of the signal interaction. The added minimum bias events are referred to as the 'pileup'. For this production the same pileup events were used in all regional centres and for all luminosities.

Two different luminosities were produced, corresponding to the expected initial and final luminosities of the collider. These are referred to as 'low' and 'high' luminosity, i.e. $2\times10^{33}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ or $10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$. The pile-up of the Minimum Bias events is done taking also into account the time of flight of the particles in the detector. For that reason, not only the 'in-time' interactions are included, but also typically the ones produced in the 5 previous and 3 following beam-crossings. Hence the total number of pile-up events may reach about (5+1+3)×20 = 180 events. The minimum bias events are small (0.4 MB) compared to the signal events (1 MB), but even so moving 180 minimum bias events for every signal event means transferring over 70 MB of data for every digitised event. This is large enough that farms have to be configured carefully if they are to perform high-luminosity digitisation efficiently.

If unique events were generated for the pileup sample (i.e. if pileup events were not re-used from one digitisation to the next) then something like 800 million pileup events (and 300 TB of disk to put it on) would have been needed. This is obviously not possible, so we use a smaller sample (200 K events) from which we (pseudo-)randomly[1)] select events and assign them to beam crossings to build the pileup. The sample is large enough that the random selection of events and their random assignment to bunch-crossings in the interaction is enough to avoid physics bias in the final results. However, it is possible that a minimum bias event may satisfy the trigger conditions by itself, which would add a considerable bias to the events that it was digitised with. A filter was applied to eliminate such events from the sample.

In Objectivity/DB, COBRA writes data into different files: for each production cycle (defined by an 'OwnerName' such as 'jm_Hit600_g125_CERN' or 'jm_Hit600_g125_FNAL' or 'jm_2x1033PU600_g125_CERN') there are 6

[1)] The actual algorithm for selecting pileup events involves skipping forwards a small but random number of events from the last event that was read, rather than selecting every event at random. It is expected that this will cause less disk-thrashing than a fully random algorithm and lead to better performance from the disk servers.

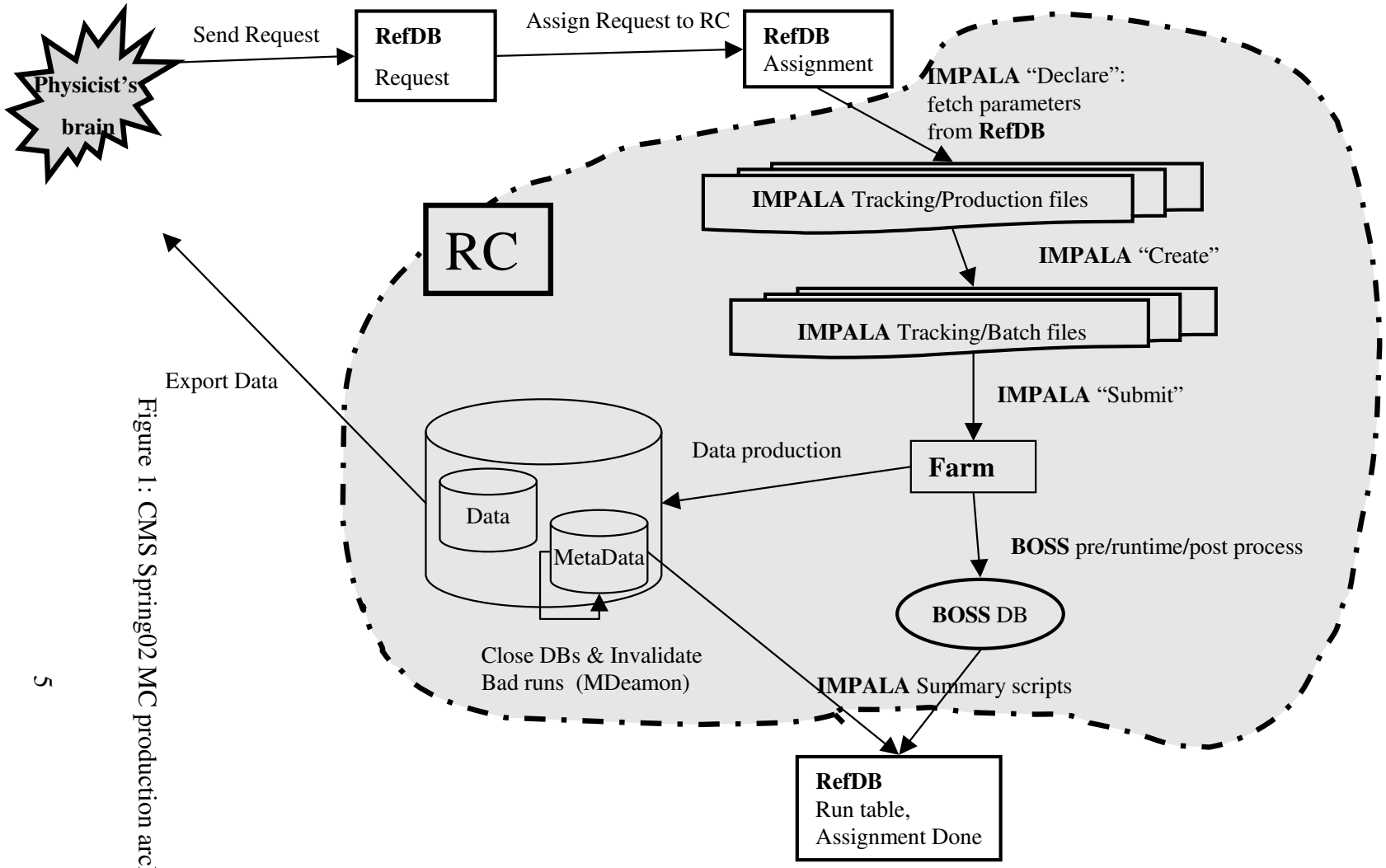


Figure 1: CMS Spring02 MC production architecture.

'META' files. Then, for each dataset with a given OwnerName, there are 'Events', 'Collections', 'MCInfo', 'Hit', 'THits' databases for the 'ooHit' step, and 'Events', 'Collections', 'Digis', 'TDigis', 'TAssoc' databases for the Digitisation phase. The META files contain the list of datasets, number of runs, number of events per run, DB pools (list of DB files containing the data of a given dataset, list of runs that have been writing into a given DB file), status of the DB files (closed or not, number of locks), status of the runs (initialised, running, closed, valid, invalid), etc...

Typically 5 jobs write at the same time in the same DB file (This number is defined by the 'DataSet:JobsPerDB' orcarc card.). When a job is writing to a DB file, a COBRA lock is set on it, to prevent the total number of jobs writing to the same DB file to go over the limit. (This lock is a COBRA flag, that has nothing to do with Objectivity/DB locks). When a job crashes, the lock is left there and has to be removed by running the COBRA application 'MDeamon':

```
MDeamon -w -1 -a -1 <ownername> <datasetname>
```

If this is not done, a long list of very small DB files may be the result of a large scale crash, since no more job will be writing to them.

When a dataset is complete, the DB files need to be 'closed' in the COBRA sense: a flag 'closed' is set to the DB file so that no more job will be allowed to write into it, and the file may be safely archived. This is also done with MDeamon:

```
MDeamon -w -1 -s 0.00001 <ownername> <datasetname>
```

where s gives the size is GB of the minimum size of a DB file to be closed.

The reason for this rather complex approach to writing data is to try to ensure that datafiles are as large as possible, so that fewer Objectivity DBIDs are used and fewer files will have to be staged in from tape to perform an analysis. Rather than have each job write its own separate files we try to fill existing files first. The practical result is that a single event may be split over many files and a single file may contain (parts of) events from many runs. The loss of a single file in a dataset may therefore render the entire dataset useless, depending on exactly what it contains.

The upper limit on filesize is 2 GB, because although Solaris supports larger files, Linux and CASTOR, do not (such support is coming). Hence COBRA has implemented a checking system for preventing the files to go over the 2GB limit. The writing ('commit') to the files is made every [DBPopulator:CommitInterval] events (typically 50). After the first commit, COBRA has an idea of the size of the data to be written at the next commit, and according to the number of jobs already writing to the DB, it can decide to close it. It may happen (it has happened) that these parameters are set too loosely, allowing files to grow larger than 2 GB. In such (rare) cases, the data were reprocessed.

If a job crashes for some reason all data produced after the last commit is lost. The job can be re-run to complete the events it was supposed to process, but it should start after the last committed event and not from the beginning. COBRA uses a datacard ('CARF:ResumeRun') to track the identifier of the run that needs to be restarted. This datacard is given by the first job that initialises the run (the run identifier is not known beforehand) and tracked in the BOSS database in case it is needed. Alternatively, the entire run can be declared invalid in the metadata and the run reprocessed from scratch. For hitformatting, which is fast, we invalidated runs and re-ran jobs that failed. For digitisation with pileup, we used the CARF:ResumeRun mechanism to salvage the CPU invested in them.

2.3 Coordination

At the beginning of the production 'season' all participating RC are requested to run a few test-assignments. These tests allow the production staff to play with the production software and get accustomed to it. The results of their tests are compared to reference files produced at CERN (fz file checksum have to be identical, and ORCA internal check-tests have to be successful, summary scripts have to be run).

When ready, each RC gets assignments for production of data taking into account

- the urgency of the data (hence the size of the farm, the availability of the production staff and the efficiency of the network for exporting the data)
- the interest of the data for the local physicists

One or two production requests are assigned to an RC at a time, so that each RC is always busy with at least one assignment and always has a pending one. This is done completely by hand. Assignments are not at all automated because there are lots of time dependent parameters to take into account. When an RC becomes idle, a pending assignment from another RC is reassigned to that RC. During Spring02 production, 724 assignments were processed [2)], of which 48 were reassigned (33 of the 48 were reassigned to CERN).

The production status is available online on the web [9], both CMS-globally and at the RC level. However, RCs are requested to report on their production status and problems at the fortnightly production meetings. Also the cms-production mailing list is used a lot for problem reports and solutions. This communication channel has been very useful and efficient. A FAQ web page was also built [10].

For the assignments which were taking too long, the RC contact persons were asked by email for the reasons of the delay. If needed, the production was taken over by another RC.

When completed, the status of the assignment turns to 'transfering'. It is set to 'done' only when the data is available in a User Federation and when a few basic checks have been successful:

- all DB are closed (this check has been switched off for two RCs because they systematically didn't close the DBs properly)
- all DBs of a given data collection are attached to the federation
- the number of produced events corresponds to the number of requested events (there are no duplicated runs left over)

When this validation procedure is not satisfied, problems are reported by email to the RC contact person where the data were produced. He/she has to make the needed corrections and the META files need to be transfered again to each RC where the data is published. Each production centre is supposed to keep the master copy of their META files.

Note that the physics content of the data was not checked because the input parameters were supposed to be coming automatically from RefDB.

When an assignment is 'done', an email is automatically sent to the requestor of the data. See Figure 2 for a state-diagram of the Assignment progress.

2.3.1 Run numbers, Random numbers and DBIDs range preallocation

Generation and Simulation Run numbers and seeds for random numbers, as well as Objectivity/DB DBIDs, were pre-allocated to each Regional Centre at the beginning of the production season.

Actually, IMPALA takes care of generating the seeds from the run number itself. The production staff only has to write the first run number of the range allocated to his/her RC in a dedicated IMPALA file (the '/run-data/RunNumber.txt' file), and no more intervention is needed anymore afterwards, except if a new IMPALA version is checked out (then the file with the run number has to be taken from the previous IMPALA installation).

Having different run number ranges allocated in such a way makes it safe to split the simulation of a dataset between two RCs (the data produced at each RC are then unique because the random number generator uses the run number as its seed). So far, there has been no clear need for a more sophisticated attribution of the run numbers. However, it could become necessary if one needs to regenerate data with the same random numbers. The random numbers used during production are recorded in RefDB. They could then, if needed, also be retrieved from RefDB.

For the 2002 production, it was decided that the Objectivity data produced for each of the four physics groups would be published into four different Federations. That would give more DBIDs available for each. Since data of a given group was produced by several RCs, the DBIDs had to be pre-allocated for each of them in order to make it possible to attach all files in the same Federation later on. The pre-booking of the DBIDs was done at CERN, and Production Federation templates were made available for all RCs. This worked perfectly well, although one

[2)] During the CMS Production Tools workshop, a number of 1200 assignments was mentioned. This number was directly taken from the RefDB tables, and included about 250 assignments registered for 2001 production (just for the record, since RefDB was not used yet) and about 200 'test' assignments.

out of the 20 production sites didn't use the template but still used the correct DBID range. Not using the template implied that the name of the produced DB files did not follow the convention and the tools used for automatic publication of the data were broken.

Both for the Run numbers and the DBIDs, if further division of the ranges was needed for production sites under the RC level, this was done internally by the RC coordinator. No mechanism exists to protect against an RC overflowing its DBID range into a range allocated to another RC. So far, this has never happened.

2.4 Operations

During Spring 2002, the production operations were the following: The RC contact persons got an assignmentID via email. All production details were contained in the email for information only. The assignmentID was used as argument to the 'Declare' IMPALA script. That script gets all production parameters from RefDB, except for

- Run number and random number seeds, which are handled by IMPALA and Run number ranges that were preallocated to each RC, and documented on the web
- Application version and executable name
- Geometry file
- boot file.

These parameters had to be set by hand by the production staff and were recalled by the 'Declare' script itself. Unfortunately, among the 724 Spring02 assignments, it has happened twice that the wrong executable was used, and it has happened once that the wrong boot file was used. In version 3 of IMPALA (in use now for the Summer 02 production), all parameters will be taken from RefDB except for the Geometry file (which will be name-checked) and the boot file.

The list of assignmentIDs was browsable from the web, for each RC individually.

In the case of INFN, assignments were given to the 'INFN' RC, but for ORCA jobs, they had to be reassigned by the INFN production coordinator to the actual production sites. This was done with a web interface for a total of 89 assignments. The reason for this reassignment is that the COBRA META files cannot be shared among different production federations. Consequently, each production site had to have a different set of META files, identified by the site name. Since the site name is taken from RefDB it is necessary for the assignment to be reassigned from 'INFN' to the real RC that will process it for it to get a unique name.

For each assignment, the production staff had to:

1. run the IMPALA 'Declare' script ($<$1 min)
2. check parameters listed above ($<$ 1 min)
3. run the IMPALA 'Create' script ($<$1 min)
4. start with a single job for the creation of the META files, when starting with a new 'owner' (to avoid poor behaviour during lock contention).
5. run the IMPALA 'Run' scripts ($<$ 1 min)
6. baby-sit the progress of the runs (between 2 and 2000 jobs, typically 200 jobs, i.e. between 1 hour and 5 days)
7. in case of crashes: clean COBRA locks with MDeamon and resubmit crashed jobs. At CERN, the resubmission of crashed jobs was automated sometime after the start of production, using the BOSS post-process facility, but could otherwise take quite a bit of time, of the order of one hour for an assignment.
8. run the summary script ($<$ 2 min)
9. invalidate duplicated runs if needed, according to the summary script output ($<$ 2 min)

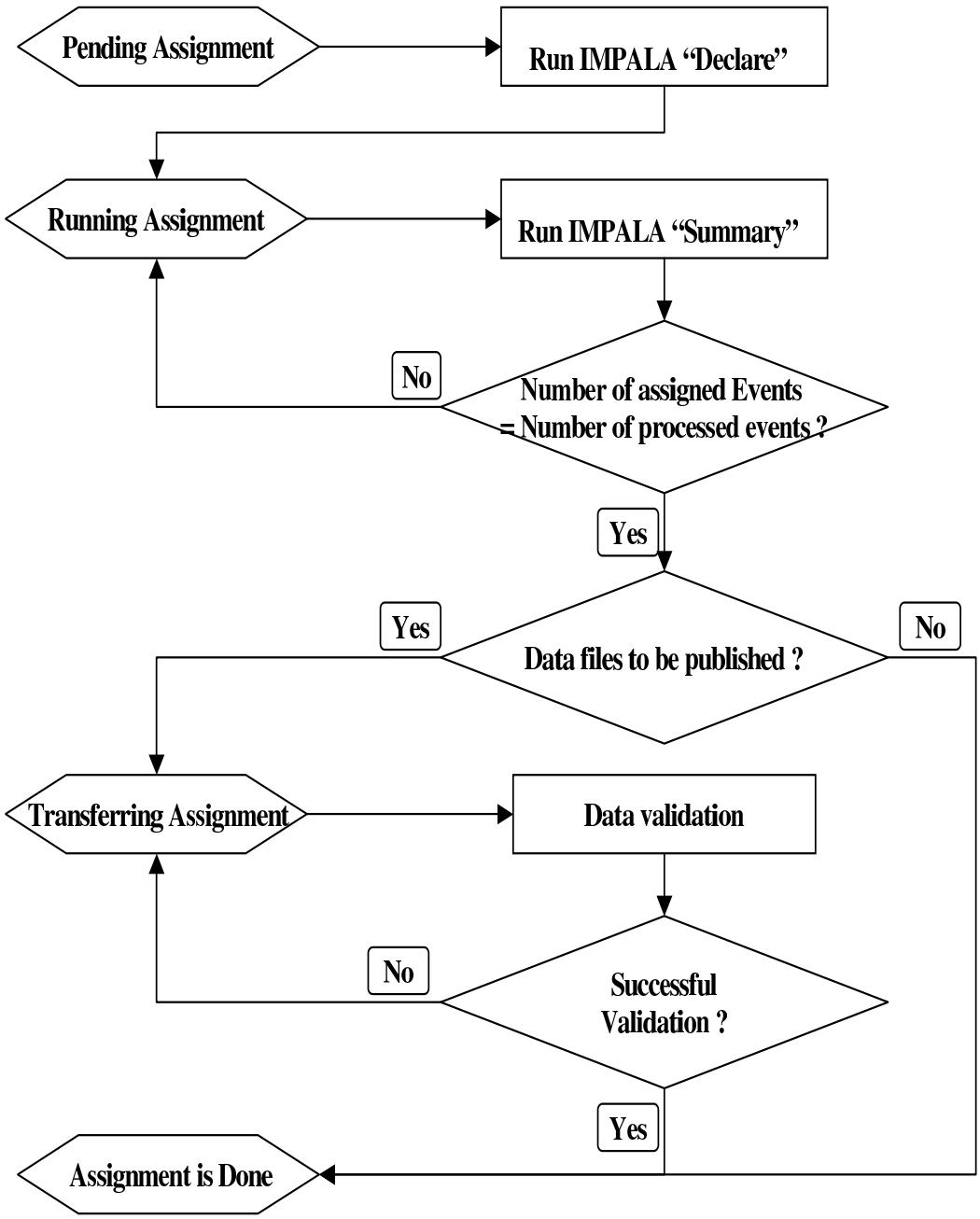


Figure 2: Assignment state-diagram.

10. run MDeamon to fix the META data after bad run invalidations ($<$ 1 min)
11. run MDeamon to close the DB files before exporting the data ($<$ 1 min)
12. publish the data for export

3 Regional Centre Participation

The CMS production system is already hierarchical in nature in that there is not a one-to-one mapping of physical sites to 'Regional-Centres' as known to RefDB. Thus Bristol/RAL is handled as a single production centre by the production coordinator despite being two physical sites. Similarly, the Moscow production centre comprises four physical farms, and the INFN distributed-T1 has ten sites (not all of which were operational for this production). There are several new production centres being planned in many places, so the ability to deploy a sensible hierarchy, at least of human effort, is rapidly becoming essential. It is simply not possible for one person to support 20 production centres directly.

3.1 Bristol/RAL

CMS production was carried out in the UK at RAL (prototype T1 centre), IC (proto T2) and Bristol (proto T3). The RAL and Bristol productions were both handled by Bristol personnel, but run separately with unique BOSS databases, run number sequences, etc. The allocation of run numbers between Bristol and RAL was handled internally, the two sites were seen as one regional centre by the production coordination staff and software at CERN.

At Bristol, production was carried out using a local linux PC farm consisting of 24 866MHz CPUs, a front-end machine hosting the IMPALA / BOSS software and PBS batch system, two IDE RAID disk servers and a dedicated Objectivity federation server / lockserver. All machines were connected by 100Mbit/s Ethernet (200Mbit/s aggregated links in the case of the disk servers), via a single switch. The production and farm were both managed by Kate Mackay. Bristol produced around 300k events, processed through the entire production chain.

No particular problems occurred at Bristol, though we were seriously inconvenienced by a lack of disk space (1TB in total). Hardware and software both worked well. Funding is currently being sought to upgrade the disk facilities; the current resources are being turned over to analysis use. All data produced at Bristol was transferred both to CERN and to RAL, in the latter case for archiving on the tape system. The Bristol $\rightarrow$ RAL link throughput was around 10Mbytes/s by the end of the production, with an upgrade to 50Mbyte/s ongoing.

At RAL, production was initially carried out using the CSF linux PC farm. This consisted of 150 CPU nodes, of a range of speeds (450MHz to 1.4GHz), and a front-end machine. The front-end machine was used to host the IMPALA / BOSS system and to host the Objectivity lockserver. CMS typically had access to 50 nodes from the farm via the batch system. We initially used a single disk server with multiple IDE RAID arrays for all data storage. Towards the end of the production period, the prototype T1 facilities came on line. These consisted of 300 CPU at 1.4GHz, and 40TB of IDE RAID disk space spread across 26 servers. CMS used around 150 CPU from this facility, and around 1.8TB of disk on two servers. The RAL farm was managed by support staff at that site; no direct root access was available to production personnel on any machine. Moreover, the front-end machine was shared between many users, and was often overloaded (not always by CMS). These factors caused major problems upon several occasions, and for future productions we will certainly make use of dedicated machines hosting the various production and database services. The production at RAL was managed by Dave Newbold and Kate Mackay.

RAL also provided access to a large tape-based MSS. All data produced at Bristol and RAL was copied to the MSS as production took place. The interface to the MSS is somewhat rudimentary, giving no functionality beyond the ability to copy a named file to a location on a tape; it was necessary to construct an indexing and cache management system to keep track of file locations on tape and to handle automatic staging of data. This was written and maintained by Owen Maroney and Dave Newbold. This system and the MSS itself were clearly experiencing scaling problems for the last part of the production, with a large number of CPUs in operation; these issues are being examined in the context of the storage management part of the UK Grid project (GridPP).

RAL produced around 250k events, processed through the full chain. The lack of throughput at RAL compared to the available facilities was due to a number of problems that hampered the production for the entire period. These chiefly consisted of unreliable hardware and overloaded shared hardware. In the latter case, it was necessary for

support staff to reboot the front-end machine several times at short notice during production, causing a corrupted BOSS database and sometimes inconsistent metadata in the production federation (use of AMS on the disk servers prevented file-level Objectivity corruption). Towards the end of the production period, these issues were largely solved, and the last three weeks or so we produced data with good efficiency. We also experienced problems with an unprecedented power cut at the RAL computer centre, and with a small number of tape failures in the MSS. Both these events necessitated reproduction of data.

It is worth noting that a major demand on time was the manual repair of the IMPALA tracking database when things went wrong. We had to go through and manually delete files from the tracking database to allow us to re-run crashed jobs, and then to check by hand that everything was consistent in the 'batch' and 'production' halves of the tracking database.[3)] We also had files wiped out at random when the disk server crashed, but this is obviously not a specific IMPALA problem.

With BOSS, we experienced frequent 'lock-ups' when lots of jobs tried to update at one time. It's not too hard to restart the MySQL server when bad things happen, but because the clients maintain an open connection to the server for the whole job duration this would interfere with running jobs. i.e. it's not possible to clean the MySQL database while any jobs are running.

The last problem was that after cleaning up from repeated problems, the job number space (for both BOSS/IMPALA) is somewhat fragmented, which makes running the 'makesummary' scripts very difficult.

These production tools worked very well while the underlying system was stable, but not so well when problems occurred. We suspect that RAL is currently unique in experiencing this level of problems, and may be the only shared farm in use for CMS production (i.e. several experiments competing for the same servers etc, not just shared batch machines). However, Grid jobs submitted 'into the wild' in the future will certainly experience overloaded and unreliable systems, so it is important that we take this into account as we extend and improve the production tools!

All data produced at RAL was copied to CERN and onto the MSS. Throughput of around 10Mbyte/s was typically achieved to CERN by the end of the production period. This was enough to hit the limits at the CERN end, namely filling the import server faster than the data could be archived to CASTOR.

In summary: Production at Bristol went smoothly, with no major problems. We were limited by the capacity of the farm and of the disk servers. Production at RAL was initially hampered by various problems, most of which were solved by the end of the production. Much larger resources (hardware and personnel) are now available at RAL, and we believe that future production will go much more smoothly.

3.2 Caltech

Caltech participation on this year's Spring 2002 CMS Monte Carlo productions has concretely laid the foundation for the future production-related activities. By getting involved in this year productions, we did not only make full use of our hardware resources but also had several important opportunities to try some emerging technologies (e.g. ReiserFS on Winchester RAID5 disks).

We were able to complete six different assignments assigned to us. All these assignments were run on tier2 cluster which is built for the purpose of CMS productions/analysis activities, high performance network measurements and grid computing developments.

It consists of 20 dual 800 MHz Pentium III worker nodes (40 CPU), a dual 1 GHz Pentium III master node and 2.5TB RAID5 Winchester storage systems. The master node (Dell PowerEdge 4400) has its gigabit connections to cluster-wide private network and to Internet through CACR's Cisco switch. During the course of productions various hardware and software related upgrades were performed. The hardware part included upgrade of RAID storage from 1 TB to 2.5 TB, networking upgrade from 100 Mbps to 1 Gbps and addition of more internal hard drives for users home directories. The software upgrade included system kernel upgrade in master node, time server (ntpd) upgrade to all the nodes, periodic upgrade of IMPALA, CMSIM, ORCA, BOSS and PBS batch queueing system. The final versions of these software used were IMPALA_2_0_8e, CMSIM125, CMSIM125-2, ORCA_6_0_2, BOSS-v2_1 and PBS 2.3.8.

[3)] After analysing this problem at CERN it looks like this manual labour could have been avoided by expert use of the information in the BOSS database. Though may we lack specific tools to support this particular case, it is not a problem of IMPALA itself.

A cluster monitoring tool [11] developed by Jan Lindheim and Suresh Singh was used to check the health of cluster nodes. This helped us starting new job or assignment when one is completed or restart any particular job whenever it is failed.

We used 'Tonys scripts' [12] to transport data between Caltech and CERN. Likewise we used GDMP to transport data between Caltech and Fermilab. Julia helped us by running one 10^{34} assignment (933) at CERN cluster due to time constraints. Similarly, we ran two Fermilab's assignments (1106 and 1107) and sent the data back there. This reflects whole production activities as a single entity.

We also had been through difficult times to make productions work. We spent a significant amount of time to understand the working principle of above mentioned software tools. Keeping software up-to-date was also a challenging task, as it was still evolving rapidly at the beginning of the production period. We did in fact get tremendous support from Greg for configuring IMPALA. Quick responses and accurate solutions of multitude of problems related to Objectivity, RefDB, BOSS, transport etc. from Tony and Veronique were accelerating factors during the entire production cycle. Vladimir's expertise on Objectivity helped us quickly analyse database inconsistency problems (caused, for example, by interference between user-analysis and the production activities, e.g. running the AMS as the wrong user).

We would certainly like to capitalize the experience we gained on last productions and have expressed our willingness to participate in the forth-coming summer productions.

We produced 214K of CMSIM events, 146K of 2×10^{33} events and total of 0.5TB of Objectivity databases. We produced about 5.6 GB of CMSIM events for Fermilab and shipped them there using GDMP.

3.3 CERN T0/T1

CERN has a dual role in CMS productions, being both a T1 centre capable of running large-scale productions and as the T0, hosting the task of coordinating the overall CMS production effort. CERN has no batch farm dedicated for CMS productions, using a combination of the LXBATCH (CERN generic batch) and LXSHARE (LHC testbed) facilities. These are presented transparently as a single LSF batch service. Typically the facilities available are somewhat dynamic, both in terms of batch facilities and server facilities, since CERN does not have T1-capacity for all the LHC experiments.

The CERN CMS production team wishes to express its thanks to the many people in CERNs IT division who helped us throughout this production.

3.3.1 The CERN farm

The CERN farm is a shared facility, in the sense that CMS at CERN has no dedicated batch hardware. We have dedicated servers plus servers on loan from the LHC testbed, plus two 'federation servers' (See [13] for an explanation), one dedicated for analysis the other for production. Judicious use of the AMS Request Redirection Protocol (RRP [15]) allows us to dynamically move the loaned and dedicated bulk-data-servers between production and analysis to match the changing needs as the production progresses. We keep the two activities decoupled as far as possible, although it is not always possible to separate them totally.

Some 420 batch nodes were in use at one time or another, although not all at the same time[4)]. The CPUs speeds vary from 450 MHz to 1 GHz, with the majority (295 nodes) being 800 MHz or above. All but 15 nodes have 256 MB or more of RAM. We ran up to 120 high-luminosity digitisation jobs at a time, but mostly we ran a mixture of all the production cycles for different datasets in order to spread the load on the servers.

The federation servers (*nobj03* and *nobj05*) are two Sun Netra machines, single-CPU 500 MHz with 512 MB RAM, dual power supplies and 15 GB mirrored disks. They host the federations, the journal files, and the lockservers, and are somewhat underpowered for this task. The bulk data servers are of five different kinds:

- *cmsdsrv06 and cmsdsrv07*: Linux servers with 150 GB RAID-0 SCSI disks, dual 600 MHz CPUs, 256 MB RAM, given to CMS as part of our COCOtime allocation for 2001. They were used for serving 'Event' and 'Collection' files (see section2.2). These files are small and frequently accessed, rather like the metadata, so it is useful to keep them on disk. Towards the end of the production *cmsdsrv06* was used purely as a redirection server instead of serving any data.

[4)] The CERN farm consists of some 800 nodes, but because of our use of dedicated queues we only saw 420 of them. 95% of the jobs actually ran on only 120 nodes.

- *cms001d - 08d, tbed009d - 014d*: Linux servers with 3Ware[5)] controllers and between 500 and 750 GB of disk. The *cms** machines are dedicated servers for CMS, the *tbed** machines were loaned to us for the production period and have since been returned to IT for other users. The machines are otherwise nominally identical, dual 800 MHz with 512 MB RAM. The dedicated servers have a total capacity of 5.1 TB, the loaned servers added another 3 TB.

- *suncmsa*: A Solaris 5.8 machine, with 400 GB of local disk, 1.1 GB RAM, and two 450 MHz CPUs. This machine also belongs to CMS.

- *idsrv1-30 and idsrv2-30*: These are two Linux machines lent to CMS by the University of Mississippi [14]. Both have single Athlon CPUs (1.1 GHz and 1.6 GHz) with 512 MB RAM and 1 TB of EIDE disk configured as software RAID-5. Both run RedHat 7.2, and have hardware that is not supported under RedHat 6.x (e.g. 160 GB disks), so neither of them could run AMS servers because of the licensing issues. Instead, they were attached to *suncmsa* via a private gigabit ethernet network, and *suncmsa* NFS-mounted their disks. *suncmsa* was then able to serve their data as if it was local. This is obviously less efficient than serving the data directly, but is clearly a viable technique.[6)]

- *shift20*: A Solaris 5.6 machine used as the output disk server. It has four 300 MHz CPUs, 1 GB of RAM, and three large disks, with a total space of 900 GB. This provided enough buffer space for 2-3 days production of high-luminosity data when we ran flat out.

All of the servers have gigabit ethernet connections. Most of them (*cms00*d*, *cmsdsrv** and *tbed*d*) were configured as CASTOR pools [16] with *cms001d* being the sole stager for all of them.

3.3.2 CERN operations/problems

The CERN production team spent some of its time on testing and debugging of the new versions of the production tools (IMPALA, BOSS, DAR distributions) trying to find out and fix the potential problems in further mass production.

At any given time we ran different production steps for different federations using different executables. That is why normally we have several IMPALA installations configured for a set of production jobs. We have also several BOSS installations pointing to the different databases. Though it sounds a bit complicated it allows us to avoid errors such as running the wrong executable or writing to the wrong federation. We have several BOSS installations for historical reasons (testing of every new BOSS version) but it also helps to reduce the load on the MySQL server caused when we run many jobs of the same type updating the same table in the BOSS database.

CERN uses the LSF batch system. This worked fairly well most of the time, but there were still occasional problems with jobs that would not start or were lost by LSF. We submit jobs from AFS and the logfiles go back to AFS. The number of jobs that we are running is such that we were obliged to run a script every half hour to sweep the AFS volume for logfiles and archive them on a machine with sufficient local disk. Eventually the logfiles are archived to CD. Once or twice when we had problems (e.g. 'acron' not starting the archiver for some reason) we actually filled our quota and lost a few logfiles.

The number of Objectivity jobs running in parallel is limited (by design) by a system of auto-submission to LSF. This was necessary to maintain the right balance of jobs for different datasets to make efficient use of the farm, so that for example 20 hitformatting jobs for one dataset would be running in parallel with 40 digitisation jobs for each of two different datasets. This system allows us to use our allocated CPUs efficiently during night hours and to reduce the time spent on monitoring the queues and manual submission. At regular intervals a crontab job checks how many jobs of the specified datasets are in the queue and submits more if necessary according to parameters defined in a per-dataset configuration file.

We found that manual re-submission of the crashed jobs is rather time consuming. To avoid this we added to BOSS postprocess the action which allows to auto-resubmit the job if it has exited with non-zero status. The BOSS

[5)] The problems seen last year with the combination of 3Ware controllers and IBM DeskStar disks has certainly been solved. We had no significant problems with any of these machines this year.

[6)] The fact that we had these two machines on loan from the University of Mississippi made a big difference to the analysis phase. The most crucial datasets were put onto these machines, rather than having to fight in CASTOR pools for staging space. This allowed the most important analyses to proceed and also reduced the load on the CASTOR staging pools.

postprocess script checks the job status and creates the corresponding entry in the IMPALA tracking directory so that the auto-submission crontab would run the failed job again. We consider the auto-(re)submitting system rather helpful and will be happy to share our experience with the other centres.

Most of the time, production at CERN ran smoothly. In particular we had none of the severe hardware problems that handicapped us so much last year. We frequently managed to digitise samples of 100K events at high luminosity in less than 24 hours.

We experienced the same sort of problems caused by the fact that BOSS uses MySQL that other regional centres have reported.

When the jobs were running for a while without updating MySQL the connection to the MySQL server timed out and the jobs crashed. This problem was fixed in BOSS version 2.1 (in case of the connection timeout it gets re-established). MySQL was reconfigured with the increased **wait_timeout** parameter(5 days instead of default 8 hours) and the increased number of maximum connections (500 instead of default 100).

Another problem showed up when BOSS MySQL got overloaded with too many updates. The job executable exited but the jobs stayed in the queue waiting for the update of BOSS MySQL DB. To overcome this trouble the following steps were implemented:

- BOSS runtimeprocess was changed so that the number of updates would be decreased
- In BOSS version 3.2 runtime updates are done at the fixed time intervals. At the time being the interval is 5 seconds. This version has not yet been deployed for production.
- We are planning to test BOSS with a more recent MySQL version which supports the use of InnoDB instead of MyISAM.

A web-based interface to BOSS, 'BODE' was developed by Alexey Filine (IHEP) and was found to be very useful. BODE is still under development, and is not yet packaged in a form suitable for release to other regional centres.

A similar problem, where the job executable got stuck in an infinite loop while exiting, caused jobs to stay around forever. This appears to be caused by problems in C++ destructors. We solved this by writing a script that examined the logfile of the running job and killed the executable if it was in this state. The jobs then exited cleanly, the data was intact, only some efficiency was lost. The muon group ROOT-tree maker is particularly problematic in that it sometimes goes into an infinite loop in the middle of an event. Because of problems of this nature we still need to log in to the batch nodes regularly to check the progress of jobs, looking for defunct processes etc. Although we have tools to automate this to some extent it is still somewhat painful.

CMS at CERN has no dedicated tape hardware, we rely on standard CASTOR facilities to transfer files to and from tape. Production data is written to a single machine (*shift20.cern.ch*) and from there it is explicitly written to CASTOR one dataset at a time as each dataset is finished. Declaring the dataset to be finished means running *MDeamon* to finalise the metadata, we do this via a wrapper script that then prepares the data for migration to CASTOR. The data is written synchronously and sequentially to tape (using standard CASTOR commands) so that files from the same dataset are guaranteed to be clustered close together on tape.

This strategy makes efficient use of tape mounts, and also minimises the potential loss of data from the loss of a single tape because relatively few datasets occupy a single tape. Although tapes are rarely lost permanently (not one single file was permanently lost during this production) they can be out of action for several hours or a few days when something goes wrong. On the other hand, if the data were spread over several tapes, you could hope to achieve better performance staging it back to disk by mounting many tapes at the same time. We prefer the conservative approach that minimises the risk from the loss of any single tape.

No effort was made to cluster groups of datasets together on tape, e.g. all the digitisations of a given sample. This would further reduce the impact of inaccessible tapes by reducing the number of affected users, and would also reduce the cost of regenerating the data from scratch.

The production federation catalogue is changed to point to the data in CASTOR once it is confirmed that the data has been written, and only then are the files deleted from the output disk. This ensures that the production catalogue is always consistent in the event that anyone is running on a dataset that is being migrated. We do not normally allow user-access to the production federation, for obvious reasons, but in the most urgent cases we sometimes allowed it. Updates to the production federation are based entirely on the catalogue contents and the existence of the files in CASTOR, there is no attempt to update the federation dataset-wise. This allows us to reclaim disk

space on the output disk in the middle of writing data to tape, which is important since we often run many datasets at once and some of them are very large. Data which has been migrated to CASTOR is declared read-only in the federation catalogue.

The user federations are updated by comparing the production and user federation catalogues. Any file which is in CASTOR in the production federation but which does not exist in the user federation is attached to the user federation. If the production federation was updated while a dataset was being migrated to tape this can mean that the user federation contains only that part of the dataset which was safely migrated at the time of the update. The data is not published to the users until it is complete in the user federation and the metadata is correct. We verify this with an explicit (automated) check on the metadata and the catalogue, but not on the data itself. The person who requested the data is notified automatically by email when the assignment is complete, and the assignment details are published on the CERN Production pages (this is true for all assignments, wether completed at CERN or not). The data remains on tape and is only staged in to disk when a user starts to analyse it. All data and metadata are also declared read-only in the user federation.

The steps in this chain are all automated from the moment the production operator has declared the dataset closed. The steps are executed asynchronously and independently, which improves robustness at the expense of increasing latency. For example, if (as has happened) we run out of space in our tape pool we do not want to wait until more tape is available before we purge everything that has already been migrated. If we wait we run the risk of filling the output disk and crashing the production in a particularly nasty manner. Typically the time from a dataset being closed to the time it is published to the users is about a day.

Metadata in the user federation is updated by simply copying the metadata files from the production federation and overwriting the file in the user federation if the version in the production federation is more recent. If users have the metadata open for analysis (quite likely!) then it can be some time (up to an hour or so) before the previous version of the metadata is flushed from the AMS cache and the new version is read in. It would have been useful at times to be able to explicitly flush the AMS cache.

3.3.3 The use of the AMS RRP at CERN

CERN makes heavy use of the AMS RRP [15] for managing the data on our servers. It allows us to add or remove servers from the farm without halting either production or analysis, and allows us to optimise the pileup-serving by load-balancing among several servers. The configuration changed several times during production. Here we describe only the most interesting configuration, in use when production and analysis were both running heavily (late May).

For the analysis, the configuration was relatively simple. *nobj03* was configured to have all files in the federation catalogues registered as being served directly by itself. It served the files if they were on its disk (e.g. the metadata), redirecting all other requests. The 'Collections' and 'Events' were redirected to *cmsdsrv07*, which served them directly. All other file requests were redirected to *suncmsa*.

suncmsa served files itself if the file was on local disk (i.e. its own disk or the NFS-mounted disks of *idsrv1-30* and *idsrv2-30*). All other requests were passed on to *cms001d*. *cms001d* redirected all requests in a round-robin manner to itself and the other bulk servers except for *tbed009d* (i.e. to *cms001d - cms008d*, *tbed010d - tbed014d*). *tbed009d* was not used in the analysis phase, it was reserved exclusively for production. *cms001d* was configured to remember where it redirected the first request for any given file, and to redirect subsequent requests for the same file to the same host (the *sticky* RRP algorithm).

The production environment was a little more complex. *nobj05* served local files directly and redirected requests for files that were not on disk, exactly like *nobj03*. Requests for the pileup events were redirected to *cmsdsrv06*, all other requests went to *tbed009d*.

cmsdsrv06 redirected all requests to three of the tbed servers (*tbed009d - tbed011d*). It did not serve files itself. It did not remember where it served files from, and always chose the next server in turn. This is the *roundrobin* algorithm, that leads to load-balancing. All three target servers had a copy of the pileup metadata and the effective serving bandwidth is thereby tripled.

tbed009d used the *partialmatching* directive of the RRP to serve the pileup itself, but for all other requests it redirected them to *tbed009d - tbed011d* using the *sticky* algorithm. In this way the same three servers are used for serving both the pileup and the signal for the production, but the pileup is load-balanced while the signal is not. This makes the most effective use of a limited number of servers.

The fact that *tbed010d* and *tbed011d* are serving data for both analysis and production is unfortunate, but at this point the production was beginning to wind down. Having three servers dedicated entirely to production would have reduced the pool available to analysis, this was a suitable compromise. Because they were shared between activities, it is possible that the pileup data would be purged off of the disks of *tbed010d* and *tbed011d* if production halted for a couple of days. If that happened then when digitisation with pileup started again the pileup files would have to be re-staged to these two servers. Because *tbed009d* did not serve analysis it would have retained its copies of the pileup files, and CASTOR is intelligent enough to do a pool-to-pool[7)] copy rather than restage the files.

3.3.4 Import/export to/from CERN

File transfer between CERN and the regional centres was based on '*Tony's scripts*' [12]. The reasons for using these scripts instead of GDMP were that the GDMP team were heavily occupied with their own milestones at the time production started, and had no time to support the CMS production effort. Instabilities in underlying software (Globus) meant that it was not possible to deploy a version of GDMP that would work well between all sites involved in the production, so we reverted to our fallback solution. The disadvantage of using the fallback solution is obviously that a greater burden falls on us to support our own tools instead of attempting to leverage grid products.

The scripts rely on the use of SSH for authentication and data transfer. Since SCP (the SSH-enabled file-copy command) is not particularly performant, the scripts were extended to be able to use BBCP [17] for the bulk transfer. This improved performance by a factor of 6 or more on transatlantic links, with sustainable rates of over 100 GB per day from the US to CERN and vice-versa.

Unfortunately, SSH has become somewhat fragmented. There are two protocol versions and several different variants (like Kerberos-enabled SSH). Configuring one site to talk to another was largely a matter of trial and error, and took a lot of effort. Given that these scripts are only a fallback solution it was not deemed worthwhile to spend time automating the installation and configuration too much. We hope that new tools will make these scripts redundant.

The scripts support load-management (server-side), checksumming, migration to tape, recall from tape and attaching Objectivity DBs to federations automatically. The data was verified (checksummed) once it had arrived and was then copied to a CASTOR pool for migration to tape. The data was not deleted from the import disk until it was safely migrated to CASTOR. Once it was in CASTOR the file names were published on a web-page that the RCs could use to determine that it was safe to delete the data from their disks (often necessary in order to free space for further production). In practice most RCs waited until an entire dataset had been imported to CERN before deleting it from their local disks, so the problem for us became one of RC $\rightarrow$ CASTOR throughput per dataset rather than just RC $\rightarrow$ disk per file. This is largely a consequence of the way our data is organised in files, if there are any files missing in the dataset the data is essentially useless.The transfer of FZ files doesn't suffer from the same problem.

We have only about 150 GB of disk on the import machine (*cmsdsrv08.cern.ch*), and could fill it from scratch in only a few hours. We had no dedicated tape hardware or infrastructure, relying purely on CASTOR to optimise tape-mounts for us. Most of the time this worked fine and we could keep up with the incoming dataflow, but when there were problems (either hardware or with bottlenecks because of the CERN-wide load on CASTOR) we were unable to use the network flat-out.

No real attempt was made to cluster the imported data on tape in a way that allows CASTOR to retrieve it optimally for analysis. Files were copied to a CASTOR pool several at a time, so CASTOR would naturally cluster them when it chose to migrate them, but no optimisation was attempted beyond that. Given the volume of data imported to CERN (and to Fermilab too) during this production this is something that deserves more consideration in future.

File-transfer itself was relatively painless, but dataset-transfer was not. Often, when running checks on the imported data (e.g. checking for the right number of events in the metadata, checking that all files known to the metadata were imported etc) we found discrepancies. Typically the metadata manipulations at the production site were not complete and the metadata had to be reimported. Many of these checks were implemented downstream of the importer, at the point where the data was served to the users in the user federation, because they were intended to check our own data as well as imported data. This caused longer delays than necessary in discovering problems and also meant a lot more communication between people who were trying to resolve the problems. The CMS-specific

[7)] In CASTOR terminology a pool is the set of filesystems used by CASTOR on a single server. So a pool-to-pool copy is a disk-to-disk copy over the LAN.

dataset-verification tools need to become flexible enough that they can be plugged in to any part of the system, from the producing site through to the WAN-transport machinery and the user analysis facility.

Data-export initially used *cmsdsrv08.cern.ch* for the export of the minbias (pileup) sample. When *cmsdsrv08.cern.ch* became busy with importing data, we switched to using *cmsftp01.cern.ch* for data-export. This is an old Solaris machine, but it is still just adequate for the scale of export we are running. It uses the MSS support in the transfer scripts to allow access to all the data in CASTOR, the files being staged in on demand. So far there has been a significant volume of data (approx 3-5 TB) exported back to RCs on both sides of the atlantic for local analysis.

The entire data import chain for CERN is rather complex:

- When data is declared 'closed' at the production site its state in RefDB goes from 'Running' to 'Transferring'. If the data is to be exported to CERN this transfer is then initiated (by hand), and when the data arrives it is attached to a separate import federation for each production site.
- The imported data is attached to the correct production federation according to the physics channel by comparing the import federation catalogues with the production federation catalogues.
- Then, when the user federation is updated from the production federation, the imported data is attached along with the data we produced directly at CERN.

The reason for the three-step process is that in many cases we imported hitformatted datasets from external sites and then digitised them at CERN (an improvement over the past, where we would import FZ files and hitformat them locally). Importing the data into separate import federations allows us to make sure we have the data in a federation where we are safe from DBID clashes in the event that a production site uses the wrong DBID range by mistake. We can run basic tools over the data in the import federation without affecting our own production, and digitise it when it is complete.

3.3.5 Analysis at CERN

Production, chaotic though it often seems, is simple compared to the feeding-frenzy of the first pass of analysis. The first time a dataset is read it must be staged in from CASTOR, which inevitably means delays while waiting for tapes. The way our data is streamed to different database files means that an analysis job can end up waiting for staging at almost any time, even in the middle of an event. The efficiency of staging can be improved by submitting many jobs for different runs, so that all files are requested as soon as possible. CASTOR optimises the use of tapes by staging all the files that are requested from a given tape once the tape is mounted, and from the file-staging viewpoint CASTOR behaved very well.

During May (the peak production month at CERN) we triggered 30 K tape mounts, this being about one quarter of the total number of CASTOR mounts for that month. We transferred 38 TB of data to or from CASTOR tapes (some imported data, some written, some staged in for further digitisation or for analysis etc). The major part of the analysis started in June, and it was not so busy. There were only 18 K tape mounts and 20 TB of data-movement between disk and tape (most of which happened in the first week or two). It would seem then that the analysis was probably not thrashing the CASTOR pool, as was originally thought. The user federations were serving about 80 datasets daily in early June (with a daily turnover of about 20 datasets). This comes to about 5-6 TB of data. We had 2 TB of non-staging disk plus another 6 TB of CASTOR staging pool (at the start of analysis), so we should have been able to cope with this.

The egamma group had been quite happy with the performance of CASTOR when they were the only group analysing data (late May) but in early June when the other groups started to get data too the analysis throughput became very low. We do not yet understand the reasons for this dramatic drop in performance. For the largest datasets it became effectively impossible to run through the complete dataset in less than several days. Eventually we had to resort to pre-staging datasets to allow certain critical analyses to proceed, and several of the more important datasets were staged by hand onto the 2 TB of non-staging disks where they were not subject to being purged by CASTOR.

At least part of the problem stems from our own poor understanding of how to use CASTOR efficiently and a lack of a clear knowledge of what features we need CASTOR to implement. For example, the standard CASTOR purging algorithm is based on the usual sort of last-used or least-used file methods. It may be better to base it on the dataset rather than just the file, so that fewer users wait for data to be restaged. Those who do wait will get

better restaging because all the files that CMS are asking for will be clustered on fewer tapes, so the overall tape queues will be shorter.

Another problem we encountered was due to a mismatch between the use of the RRP and one of the newer features of CASTOR, internal pool-to-pool copying of data:

For files that are declared read-only (all of them, in our case) CASTOR can recognise that a file already exists in one of its pools if, for some reason, it is asked to stage it into another pool. Then it makes an internal pool-to-pool copy rather than restaging the file from tape.

The RRP library supports on-the-fly updates if its configuration, but when the configuration is updated the in-memory table of where files are served from is lost. This is deliberate, since changing the RRP configuration often means adding or removing hosts. Reinitialising the table completely allows the servers to rebalance themselves and take up more load or reduce their load accordingly. This means that a file which is being served from one server may, a few seconds later, be served from another server instead. Providing the files are read-only there is no problem with this, and the new server will fetch the file by an internal CASTOR pool-to-pool copy. The original server will eventually purge its copy when it needs the space.

The CASTOR pool-to-pool copy is driven by the staging host (*cms001d*) and is a two-step process, the file being copied first from the source to the staging host and then from the staging host to the target machine. If there are only two or three servers involved then this two-step process is probably acceptable (it is certainly easier to implement!). When the number of servers becomes large (over 10 in our case), and the users are very active, this is not so good. First, with many users, many files are requested in a short space of time. If there are a lot of them on disk already then there is a massive migration of data from the previous server to the new server, all going through the bottleneck of *cms001d*. It might actually be faster in this case to restage the files from tape because you can get paralellism from mounting many tapes at a time instead of serialising through one machine! Secondly, while a file is being copied, it occupies twice as much space on the pool. There appears to be no flow-control on the pool-to-pool copying, so many dozens of transfers can happen at the same time, which only makes the congestion worse. We saw load-averages of up to 40 on the stager and continuous network traffic of 20 MB/sec for many hours through the stager but very little traffic from the rest of the servers to the users batch jobs.

Another noticeable feature was that the time it takes to mount a tape varies considerably, for both reading or writing. Over half of all stage-in requests were satisfied in less than 10 minutes, 95% were satisfied in less than one hour, 1% took over 2 hours, but the tail goes out to many tens of hours. Writing to tape was better, not surprisingly, because data which is not yet written to tape is vulnerable to loss and therefore takes priority over reading data back from tape, but here too the tails of the distribution are long. 95% of the tape-mount requests for writing were satisfied in less than 55 minutes. 1% of the tape-mounts took over three hours. The longest took time taken to mount a tape for writing was over 4 hours 40 minutes.

With such a distribution it is not surprising that users become stressed when they see their jobs waiting for tapes to be mounted, or that the production operators are worried about being able to flush data from the disks fast enough.

Finally, the entire system of serving data via the AMS, RRP, and CASTOR is so complex that it is very difficult to diagnose problems. How long should users wait before deciding that the system isn't working and it is reasonable to ask for help?[8)] When users report problems we first had to track down exactly what file they were accessing and which server was serving it, then hand over the problem to the CASTOR or Objectivity support team (or both!). Simply identifying which domain of expertise is relevant to each problem is non-trivial! This is very expert-intensive to say the least.

We clearly need a better understanding of how to use CASTOR before we run our data-challenges, and will need the help of the CASTOR team to get that understanding. This should not be taken to infer that CASTOR itself is part of the problem, rather that we are a long way from understanding how to use CASTOR as part of our solution.

3.4 Fermilab Tier-1 Center

During the Spring 2002 production season, Fermilab served as the Tier-1 center for USCMS. Unlike INFN, US Tier-2 centers received their assignments directly from CERN and not indirectly through Fermilab, so we will only concentrate on Fermilab in the following.

[8)] longer...

3.4.1 Fermilab Facilities

The main facilities used for production at Fermilab include a 40 node farm of dual PIII 700 MHz CPUs with 40 GB of local disk and 512 MB RAM each (*popcrn01-40*), two quad PIII 700 MHz CPU file serving machines with 250 GB of RAID attached and 1024 MB RAM each (*velveeta* and *gallo*), and a Storage Tek tape silo currently containing about 300 tapes at 60 GB per tape. The tape system is managed by Enstore [18] and lately with a dCache [19] interface as well. Other facilities included several dual PIII machines for file transfer or for federation serving and were allocated on an as needed basis, though *snickers* with 1 TB of attached RAID became the usual file transfer, and *burrito* with 500 GB of attached RAID became the User Federation server machine. A backup User Federation was run on *cmsun1*, one of only two Solaris machines at Fermilab CMS. All of the servers have gigabit ethernet connections, and the connection to/from the Enstore system is also gigabit. Each farm node is connected through a fast ethernet connection, however.

The local batch queue system is provided by FBSNG [20] which is a very configurable batch system that allows for dynamic reallocation of farm nodes to specific production related purposes and running conditions. Our typical allocation used 9 nodes for serving pileup, 2 test nodes, and 29 generic nodes. The pileup-serving nodes each contained approximately 1/9 of the minimum bias files needed to do Spring 2002 production. They also contained a copy of the pileup from Fall 2000 production in order to serve old requests; however this only happened twice during the early stages of Spring 2002 production.

The 250 GB RAID partition of one of the file serving nodes (*gallo.fnal.gov*) was NFS mounted to each of the farm nodes. The IMPALA production scripts were installed there and visible to the farm nodes. DAR (or DAR-like) distributions of the CMS production runtime environment were also located on this partition. Use of AFS was scrupulously avoided for any part of the Spring 2002 production to avoid performance problems seen during Fall 2000 production.

Objectivity AMS servers and Lock servers were allocated in the following way. Each pileup server plus the two fileservers each ran an Objectivity AMS server. The journalling was handled by one of the production farm nodes (*popcrn03*) to avoid hitting the big fileserving partitions with lots of concurrent tiny transaction requests. The Objectivity Lock server was also run on this node. The User Federation machines also ran AMS servers and Lock servers. The AMS servers on the UF machines were special in that they ran using an AMS/Enstore interface which automatically staged needed Objectivity files from tape to local disk. Later, with the addition of *ketchup* and *mustard* (two new fileservers each with 1 TB RAID attached) a dCache pool was formed and an AMS/dCache interface was run instead. dCache allows for a performance improving buffer between tape and disk.

3.4.2 Fermilab Operations

At the beginning of Spring 2002 production, a set of roles was designed for the production team. The roles included:

- Coordinator: responsible for contacts with the outside world and coordinating with both CERN and the US Tier-2 centers.
- Coach: responsible for overseeing day to day operations and making sure that the operations staff was in good communications with the support staff at FNAL.
- Data Access Administrator: responsible for maintaining both the User Federations and the production federations, including the attachment of pileup datasets and datasets shipped from the outside. Also responsible for maintaining a catalog of available datasets.
- Database Administrator: responsible for maintaining and taking backups of support databases, such as the IMPALA/MySQL database and the BOSS/MySQL database.
- Mass Storage Manager: responsible for maintaining excellent contact with the Mass Storage support teams at Fermilab and chasing down problems (usually our own fault,) and for organizing our data on tape. Responsible for AMS/Enstore interface issues.
- Software Manager: responsible for making sure that the production software (IMPALA/BOSS) is installed and up to date, responsible for making sure that the needed CMS DAR distributions are installed.
- Hardware Manager: responsible for working with support staff to certify that the production hardware is alive and functioning well, responsible for requesting extra hardware for extra busy times.

- Operations Staff: responsible for running day to day production and updating the RefDB at the end of assignments.
- File Transfer Manager: responsible for satisfying all file transfer requests between FNAL and CERN or FNAL and US Tier-2 sites.
- Grid Software Guru: a future role is envisioned for a Grid software troubleshooter.

Each role had two persons assigned, a primary and a secondary, in such a way that the secondary could take over from the primary seamlessly in case of vacations or absence. Initially the 'Coach' role was a rotating assignment, but it was felt that this led to a lack of continuity in the group so it became a permanent assignment. There was also some feeling that the roles were too 'finely tuned', but the analysis is still useful as it points towards a set of use cases for Grid software; the overall load of FTEs should eventually decline as the roles become automated. The assignments for Spring 2002 production are shown in table 1.

Role	Primary	Secondary	Estimated FTE on Primary
Coordinator	Greg Graham	Shafqat Aziz/Hans Wenzel	15%
Coach	Greg Graham	Shafqat Aziz	20%
Data Access Admin	Shafqat Aziz	Shazaad Muzaffar	15%
Mass Storage Manager	Moacyr Souza	Enstore Team Member	5%
Software Manager	Greg Graham	Natalia Ratnikova	10%
Hardware Manager	Greg Graham	Hans Wenzel/Joe Kaiser	10%
Database Admin	Yujun Wu	Greg Graham	5%
File Transfer Manager	Shafqat Aziz	Moacyr Souza	25%
Grid Software Guru	Shafqat Aziz	Joe Kaiser	0% (Spring 2002)
Operations Staff	Greg Graham	Shafqat Aziz/Yujun Wu	20% (each)

Table 1: Production role-assignments at Fermilab during Spring02, with estimated FTE contributions. Moacyr's roles were taken over by Shafqat Aziz upon Moacyr's departure.

The chart estimates about 1.7 FTEs during the Spring 2002 production for FNAL Production Operations. This chart does not include support of IMPALA software during the Spring nor support given to the USCMS Tier-2 sites for operations (nor their support of us in some cases.) This accounts for an additional 25% FTE from Greg Graham and 50% from Michael Ernst respectively for support.

Backups of Fermilab metadata were periodically taken by the Mass Storage Manager, usually after the completion of a dataset or a significant portion of a dataset. More frequent backups were taken by the operations staff whenever the farm was drained and a new dataset was about to begin. This more frequent backup was usually kept on disk and was intended to 'rollback' from a production that was just started and yet failed almost immediately because of configuration errors or because someone forgot to check if all of the Objectivity servers were working correctly; if a pileup server was not responding for example, digitization could slowly continue until all processes were waiting on the affected pileup server. In that case, it was expedient to restore to the point just before that dataset began and start over.

Fermilab did not learn how to deploy RRP until late May of 2002; until that time we became adept at using OO_DB_PATH and OO_DB_HOST environment variables in the IMPALA control_env file to control where data was being written. This was automated by a script which watched available disk sizes and throttled the queue using declining FBSNG quotas while changing OO_DB_PATH and OO_DB_HOST. It was forbidden for other FNAL staff to write huge amounts of data to the production machines while production was running. In one case however, we lost an 'Owner name' because the Coach input a bad disk size limit for which to watch on the *velveeta* partition.

MDaemon was not used at Fermilab until May of 2002. Once we started using MDaemon, we noticed the following problem: Before MDaemon, problem runs were identified using dsDump. Crashed runs would leave the runs in a state of 'Running' or 'Initialized' and were thus easily identifiable for the Invalidation step. However, MDaemon is able to close and Validate some of these runs, so it became important to remember to do the Invalidation step before the MDaemon step. Due to the possibility of ending up with multiple runs, it is important for the production tool to have a way to trace the input data to all instances of output data to guarantee uniqueness. IMPALA keeps this kind of data, but it is hard to compile.[9)]

We didn't usually use the CARF:Resume flag on low lumi runs because the BOSS runtimeprocess script was broken and we had to put it in by hand. We never used it for hits runs for the same reason. Both of these run types finished relatively quickly. Correspondingly, this is where we noticed the MDaemon problem; we did not to see it for high lumi runs since we always used the CARF:Resume flag for these runs. Since the datasets were fixed before running the summary scripts, we did not have the opportunity to use the summary scripts as an alternate way to detect duplicate runs.

It took quite a long time in the beginning for Fermilab to run the digitization step. This was due to incorrectly transferred pileup files from CERN. We were plagued by failures to catch CRC checksum errors; however this was mostly due to operator error as the tools used for this file transfer support checking the checksums automatically.

Before starting Objectivity servers, we do increase the max open files with ulimit to 4096. We do run into robustness problems at a lower level of operation than CERN, however; and have therefore developed different strategies to manage these problems.

The rest of the discussion of Fermilab production operations will be broken down by production step.

3.4.2.1 Generation/Simulation

We found as during the Fall 2000 production that the steps of Pythia generation and GEANT3 simulation were very smooth. The automatic mechanism of retrieving parameters from the RefDB made life particularly easy compared to Fall 2000 production. We have 826 recorded instances of CMSIM running successfully to completion and no instances of CMSIM failing for Spring 2002 production. Finished CMSIM output files were written directly to Enstore from the farm nodes. Since Enstore queues tape writes, this did not induce a bottleneck obviating the need for a fileserver node for this purpose.

3.4.2.2 Hit Formatting

We found that per federation, we could run safely no more than 15 instances of writeHits without inducing AMS server failures on the file serving nodes. This situation was improved by moving the journalling and transaction support off of the fileservers as described above. During the interstitial period between Fall 2000 production and Spring 2002 production, we were able to run 60 simultaneous instances of writeHits by hosting four production federations on two fileservers with four different Lock server and journalling server nodes; but for the Spring 2002 production it was considered enough to utilize only one federation for this purpose. During Spring 2002 production, we have 1267 instances of writeHits running successfully to completion and 14 instances where writeHits exited with status 8. In those 14 cases, writeHits apparently lost contact with the AMS server on the file serving host. In these cases, IMPALA caught the error code they were redone by re-issuing the CreateHitsJobs command and generating new scripts to reprocess the problem runs.

IMPALA handles this: all you have to do is re-run the Create step. Any executable that exits with non-zero exit code ends up in 'problems' so IMPALA will know that the 'declared' input data product corresponding to that problem is no longer running nor is it processed correctly. Create will then generate another script to process it.

3.4.2.3 Digitization - No Pileup or Low Luminosity

We found that per federation, we could safely run about 45 or 60 instances of writeDigis when the pileup conditions were 2x1033 or NoPU respectively. There were 707 instances of writeDigis successfully running to completion, 239 instances of failure with exit code 139, 9 instances of failure with exit code 8, and 2 other errors with non-zero exit. The failure modes was most often the loss of the AMS server on a pileup serving node and sometimes the loss of a Lock server or an AMS server on the fileserver node. It was observed that troublesome pileup AMS servers often entered a state in which they responded correctly to an 'oocheckams' command, but were nonetheless dead with respect to writeDigis processes. The only way to observe this behavior outside of actual production was to issue an 'oostopams' command at which time the zombie AMS servers would fail to die. It was not considered a good way to detect the problem, of course. Rather, pileup AMS servers were stopped and started prophylactically every few days. In these cases, IMPALA caught the error code they were redone by re-issuing the CreateDigisJobs command and generating new scripts to reprocess the problem runs.

3.4.2.4 Digitization - High Luminosity

We found that per federation, we could safely run about 25 instances of writeDigis when the pileup conditions were '1034.' There were 1022 instances of writeDigis successfully running to completion, 327 instances of failure

[9] The summary scripts would have detected this duplication from the information in BOSS, but Fermilab staff were in the habit of only running summary scripts when the dataset was complete.

with exit code 139, 44 instances of failure with exit code 8, and 1 other error with non-zero exit. The failure mode was most often the loss of the AMS server on a pileup serving node and sometimes the loss of a Lock server or an AMS server on the fileserver node. It was observed that troublesome pileup AMS servers often entered zombie states here too. In these cases, IMPALA caught the error code they were redone by re-issuing the CreateDigisJobs command and generating new scripts to reprocess the problem runs. For the high luminosity problem cases, we used the CARF resume run flag to recover partial runs since the processing time was so high. Towards the end of Spring production this became automatic using the runtimeprocess scripts of BOSS, but before that it was done by hand.

3.4.2.5 Ntuple Production

We found that per federation, we could safely run about 35 instances of *JetMet*, The JetMet-group ntuple-maker, without crashing the AMS server when the production federations were used or when the User Federation was used. There was a significant improvement in response time when the production occured per-dataset and a decision was made to pre-stage the data to disk from tape in advance of production. Until May 2002, Ntuple production at FNAL was run exclusively by Yujun Wu for the JetMet group on the FNAL user machines. This accounted for 80% of all ntuple production at Fermilab. Starting in May 2002, the Ntuple generating step was incorporated into the official IMPALA production scripts and run on the regular production nodes.

3.4.2.6 Statistics on Dataset Sizes in Enstore

data-type	PRS-group	data-volume (TB)
CMSIM	all	0.5
ORCA	Jetmet	1.0
ORCA	Btau	1.7

Table 2: Data Produced at Fermilab during Spring 2002 production

data-type	PRS-group	data-volume (TB)
CMSIM	all	0.8
ORCA	Jetmet	1.7
ORCA	Btau	0.5
ORCA	Muon	3.0

Table 3: Data transferred to Fermilab during Spring 2002 production

Tables 2 and 3 show the breakdown of the data produced by and imported to Fermilab. The total data from the Spring 2002 production on Enstore is 9.2 TB.

At Fermilab this data is available in three objectivity federations corresponding to their groups. These federations are

- Jetmet User Federation:
  snickers.fnal.gov::/home/Federation/UserFederation_Spring02/jet0102/jet0102.boot
- Btau User Federation:
  snickers.fnal.gov::/home/Federation/btau_production/btau0102.boot
- Muon User Federation:
  velveeta.fnal.gov::/data/Federations/muon_federation/muon0102.boot

There are only a few users who are using these federations for their analysis jobs for the time being. These users are not yet causing any significant load on our user federations.

3.4.3 File Transfer to/from Fermilab

3.4.3.1 General Considerations

Within the scope of the production activities a variety of data transfer tools were evaluated, including

- FTP
- SCP
- BBCP
- GDMP
- Globus-url-copy

Even though Internet backbone speeds have increased considerably in the last few years due to projects like Internet 2 in the US and GEANT in Europe, we found that our tools rarely take full advantage of these high-capacity networks. In fact, recent data for Internet 2 show that 90% of large TCP transfers use less than 5 Mbps, and that 99% use less than 20 Mbps out of the possible 622 Mbps.

By design, TCP was built with robustness as the primary goal, and hence hides most problems. When something goes wrong, TCP silently compensates at the expense of reduced performance. Some of these issues are related to TCP configuration problems (e.g. small buffer space or features such as SACK being improperly negotiated). Other problems are due to the application (mainly small messages or pauses in the data flow) and in the network as well.

Finally, it has been proven that end systems can achieve good performance over high-capacity WANs through a number of techniques, including proper manual tuning.

TCP is a window based protocol: new data is transmitted into the network when old data has been received as indicated by acknowledgements flowing from the receiver to the sender. The data rate is determined by the total amount of outstanding unacknowledged data in the network, referred to as 'the window size'. The window size (or data rate) is limited by the application, the buffer space at the sender or the receiver and by the congestion window (using 'slow-start', 'congestion avoidance', and related algorithms) to find an appropriate share of the network capacity on the path between the source and the destination. TCP repairs missing or corrupted data segments by retransmitting data from the retransmit queue within the sender's buffer to the reassembly queue in the receiver's buffers. These buffers generally have default sizes, which can be changed by the application by using a system library call. In addition, the operating system in general has limits that restrict the maximum buffer size that the application can request.

Ideally, TCP throughput should be only limited by some intrinsic bottleneck such as disk data rate or network path capacity. However, in practice, throughput is often limited by the send or receive buffer size or by an artificially small congestion window induced by some problem in the network.

As was indicated above, buffer sizes must be adjusted for both the send and receive ends of the TCP socket. To achieve maximum throughput it is critical to use optimal send and receive socket buffer size for the path. If the buffers are too small, the TCP window size cannot fully open. Making the buffers too large is wasting memory (though memory is cheap these days). The optimal TCP window size is usually twice the available bandwidth-delay product for the path, where delay is the one-way latency of the path.

As network capacity has increased in recent years, operating systems have gradually changed the default buffer size from common values of 8 kB to as much as 64 kB. However, this is still far too small for today's high speed networks. For example, most of the nodes devoted to data transfer are connected with GigE interfaces to an OC-3 (155 Mbps) wide area network (e.g. ESnet, MREN) with an upgrade to at least OC-12 (622 Mbps) coming shortly. With typical round trip times (RTT) between 50 and 120 ms the TCP buffer should be roughly (150 Mbps / 8 bits/byte * 0.05 sec) = 1 MB (example shown for RTT=50 ms) and for OC-12 networks 4 MB respectively. Using a default TCP buffer size of 64 kB, the maximum utilization of the network path will only be less than 10% under ideal conditions.

To work around problems such as buffers that are too small or loss-rates that are too high, some data transfer tools (e.g. bbcp, GridFTP) are utilizing parallel data streams. Even with properly tuned TCP buffers, parallel streams can improve performance. Let's take for example the connection between UCSD and Fermilab. The RTT was measured with ping to be 60 ms and the WAN link runs at 155 Mbps. With different tuning parameters, actual measured transfer speeds spanned more than an order of magnitude. Tuned TCP buffers alone provided more than a 10x performance increase, and parallel sockets alone yielded about a 15x performance improvement, but it requierd a very large number of streams. Using just 5 parallel streams with tuned TCP buffers, we were able to outperform 20 untuned streams.

Packet loss is interpreted by TCP as an indication of network congestion between a sender and a receiver, and may be due to intermittent hardware faults or physical layer distortions. In fact, measurements have shown that about 1 in 7500 datagrams pass the CRC checksum but fail the TCP or UDP checksum, indicating that the data was damaged in transit. TCP treats segments lost or damaged in transit as loss and goes into one of its congestion avoidance algorithms. Therefore, the use of parallel streams provides an increase over optimally tuned single stream TCP because even if one or more of the streams experience random loss and slow down, the other streams keep sending and help to keep the pipe full, allowing the application to utilize a greater fraction of the network.

Several data transfer tools now include the ability to use parallel streams, like GridFTP, gsiftp (used with globus-url-copy) and bbcp.

3.4.3.2 Data Transfer Results

We used the following tools to trasfer data during Spring2002 productions.

- SCP and FTP
- BBCP
- GDMP
- Globus-url-copy

Each of the above methods provides different throughputs as each of the above tool has certain limitations. We used all of these tools during Spring 2002 production and our observations are:

SCP and FTP: Both of these methods use the default system's TCP buffer size and therefore are very slow. Their throughput is only few 100 KB/Sec. Multiple streams can be used to speed up data transfer.

BBCP: Actually Tony's scripts use this method for data transfer. This method is much faster than simple SCP and FTP and has been used a lot during Spring 2002 production. Using this method with proper adjustment of parallel streams, one can get throughput up to 5 MB/Sec.

GDMP: This method of data transfer is developed on globus. It requires grid-proxy certificate for authentication. This method also used the system's default TCP buffer size and does not offer flexibility to change TCP buffer size. GDMP offers upto 1.5 MB/Sec throughput.

globus-url-copy: This method is based on gsiftp and requires grid-proxy certificate for authentication just like GDMP. But this method provides flexibility to tune TCP buffer to proper size. Also we can adjust number of parallel streams to get maximum throughput. The throughput of this tool is more than 12 MB/Sec.

The basic purpose of this research was to investigate about how to get maximum throughput using the existing file transfer tools and indentify the best suited tool for this purpose. Based on this research and results we got using existing data transfer tools we found out that the globus-url-copy is the best suited data transfer tool to get maximum throughput at the moment. We just need a little TCP buffer tuning and adjustment of parallel streams to get maximum throughput. At Fermilab we are using a script which is based on globus-url-copy. So far we have successfully transferred more than 6 TB of data from different regional centers to Fermilab using this script.

The data transfer we used which is based on globus-url-copy enables simple MSS to MSS file transfer and retries failed transfers and partially does server side load balancing. It does not do checksum as we do it by hand. Of course this is not a good way.

Actually we can easily incorporate globus-url-copy mechanism into Tony's scripts and we can get all the other benefits for site to site data transfer including checksum.

For MSS to MSS data transfer, at Fermilab a gridFTP compliant server has already been implemented for dCache. There are some utilities available that we can incorporate into our scripts to perform automatic checksum on data streams while moving data. For example adler32.c mechanism can be used for this purpose.

3.5 Florida

The proto T2 center at the University of Florida has built a computing farm for CMS production and analysis activities, developments in grid computing and data transfer activities. The CMS production facility consists of the following:

- one head node both on public/private network, connected to a 1 Gbit/s switch and with 100 Mbit/s connections to the worker nodes
- 60 farm nodes on private network
- all nodes are dual P3 Linux boxes (rack mounted) with 1 GHz CPUs, 512 MB RAM and 75/80 GB IDE hard drives (IBM Deskstar/Maxtor)
- disk server: two RAID5 arrays - 0.5 and 1 TB (T3 Storedge - fiber-channel/ Flash Winchester - SCSI HD - fiber-channel), connected to a SUN server
- PBS batch system.

The experience with the farm and the disk storage is quite positive so far. The most important hardware problem was caused by the IBM Deskstar disks, and fixed after installing new firmware.

The production is managed by Dimitri Bourilkov and the farm by Jorge Rodriguez.

The CMS production software suite for spring 2002 comprised of:

- IMPALA versions v2_0_4a, v2_0_6d, v2_0_8e
- BOSS 2_1 / MySQL 3.23
- CMSIM 125 and CMSIM 125.1
- ORCA 6.0.1 and ORCA 6.0.2
- Objectivity-scripts-2.0.2 / BBCP

was installed, adapted and tested on the GRINUX farm at the University of Florida. We proceeded step by step: first install MySQL stand-alone, test - run - understand it, then BOSS in the same way, IMPALA etc. In a complex system like the CMS production environment good understanding of the components is important when things do not run as expected.

We have used the RefDB both to make requests and to produce them - in all cases it worked fine.

The experience with BBCP is mixed - it is difficult to configure (looks for ssh just in a fixed place), but file transfers are fast. In stand-alone tests from the farm to local desktops we reached rates $\sim$ 12 MB/s (close to the disk writing speed!).

The jobs write their output (except when writing to Objectivity/DB) on the local disks of the farm nodes where we have enough space. At the end of the job the output is moved to a centralized location on the RAID systems.

Five assignments for CMSIM production were completed. Total 540K events, 739 gigabytes of produced FZ files, 34100 CPU hours, corresponding to approximately 1/6 of the total volume (size of FZ files, which is a good measure for the CPU time) of the worldwide CMS production for spring 2002. The variation of CPU times per event for the CMSIM step is large: 63–64 s for the muon assignments, and 220–238 s for the electron-photon assignments.

For the digitization phase 10 farm nodes were used as AMS servers besides running jobs. The pileup files were stored on the local disks of these nodes (one single copy of the pileup distributed across all the pileup servers). Up to 25 jobs could be run in parallel without saturating the network both at low (2×10^{33}) and high (10^{34}) luminosity. This configuration is similar to the one at UCSD, but is not using the RRP.

The size of the files produced and stored in Objectivity database in this stage: hits 20 GB, digitizations at low luminosity 50 GB, digitizations at high luminosity 150 GB, total 220 GB.

The produced files were transferred to CERN with Tony's scripts and to FNAL with GDMP for use by the PRS groups for the DAQ Technical Design Report.

In autumn 2001 a special large sample of minimum bias events was requested to study the calibration of the CMS electromagnetic calorimeter. In addition, this sample was recognized as very important for unbiased studies by the muon and jets/missing energy physics groups of CMS. The production of such large sample requires a special 'vertical' mode of running, where the intermediate data products are discarded after producing the ntuples for the

three physics groups. The corresponding software based on IMPALA version v1_7_2, was installed, adapted and tested on the GRINUX farm at the University of Florida, which became one of the places outside CERN to operate in this special high productivity mode. [10)]

This assignment was completed successfully in January 2002: a total of 4.1 million events, corresponding to a produced data volume of 7.3 terabytes, and 3075 CPU days. Besides completing the production, this run was very useful for putting our farm under full stress for more than a month and gaining experience with operating on up to 120 CPUs simultaneously. Scalability, software and hardware problems were discovered and fixed. This experience proved to be invaluable in operating smoothly for the spring 2002 CMS production.

The vertical running mode has several advantages:

- efficiency
- easy error handling and restarting jobs
- good candidate for gridifying and study of virtual data concepts in a production setting
- similarity with analysis tasks where the user wants fast output for developing new ideas and is not interested in keeping the intermediate data products.

In summary the Florida proto T2 center is now fully commissioned for all types of running.

Some problems:

- MySQL configuration:
  The MySQL defaults do not match the needs of BOSS e.g. when sending 400 jobs at once. The **connect_timeout** was changed from 5 to 60 seconds, the **max_connections** from 100 to 500, and the **wait_timeout** from 28800 to 432000 seconds (from 8 hours to 5 days - enough for the longest jobs to complete).
  There is also some ambiguity about the MySQL passwords: only the MySQL server should be installed as root, then the user who will use BOSS should set the password from his/her account (using the option '-u root' in the mysqladmin command - here root refers NOT to Linux, but to MySQL).
- local IMPALA installation still requires lots of manual work, and this for each new version
- BOSS/pbs interface was developed from scratch (thanks to UCSD for input on this)
- frequent ORCA crashes (status 139) while running digitization, affecting 15 to 30% of the high luminosity jobs. This necessitates resubmission by hand of the crashed jobs. The automatic handling of data cards at this stage was very helpful.
- the inherent instability of running a large farm; monitoring and fixing problems often requires human intervention:
  One example is the 'queue drain' - when N jobs are in the queue, and one node starts to fail, all jobs go there just to crash, the queue is soon empty and other nodes will become idle (especially at night or during the weekend). This makes manual restarts unavoidable and can lead to loss of efficiency. For the 'vertical' running mode this problem was solved by finding the points where jobs fail and putting them to 'sleep' for one year in place of letting them crash and exit - in this way the faulty nodes 'self-block' themselves preventing further queue drain.
  Another problem is that when jobs start to fail, on the surface it looks like e.g. a pbs (or any other scheduler!) problem, but after we analyze the farm and run short test jobs, we always find a node which has developed problems (hardware - disk, or network, or even software) which are at the root of the failure. This is not easy to trace/monitor - we have to do it by hand and often to stop/start pbs which requires root privileges. So even having full access to our farm it is not easy - on the grid it will be much harder to find, understand and fix errors.

[10)] Imperial College and CERN also took part in this 'vertical' production. Both Florida and Imperial College were comissioned this way.

3.6 Imperial College

Farm Configuration:

Hardware resources:

- 4 dual processor master nodes 1GHz P3 2GB memory 0.3 TB disk each.
- 30 dual processor worker nodes 1GHz P3 managed by pbs.
- 1 storage node 3 TB disk.

Software configuration Objectivity:

- *gm00*, *gm02*, *gm03* master nodes serving pileup
- *gm01* masternode as journal and lockserver
- RRP–enabled AMS used on *gm01*, *gm00* to serve pileup roundrobin style (this is the same as the CERN load-balancing configuration) and control where output files are created. This was important because we filled almost all space on the master nodes with Objectivity DB files.

Other:

BOSS MySQL server on *gm01*, pbs master *gm02*

Problems encountered:

- Digitisation jobs crashing. Mostly solved using the *NewDBServer* perl script to accelerate the creation of new database files, and corrected minbias META data.
- Only SSH protocol 2 allowed, which caused problems with file exporter scripts. This meant that Tony had to do more work at the CERN end.
- Problems with hardware. The farm was only a few months old and this was the biggest use of it yet. This caused the master nodes to crash occasionally. Our new system administrator made some kernel upgrades, configuration changes etc to solve these problems.

3.7 IN2P3

We processed 200K events CMSIM. Each event was rather long , more than 200 seconds on 1 GHz CPU. This was due to the low Pt cut on forward tracks. For this type of events it may be easier to use 250 event runs instead of 500.

We had about 40 CPU for CMS, from the 200 double processors 750 and 1000 MHz available for all the experiments running at IN2P3. All were running on RedHat 6.1 In the future, priority will be increased so to satisfy LHC data challenges. Better is when production are planned in advance so that we can optimize the farm usage.

The transfer to CERN was done with one export server using 477 GB disk space. The transfer was done in about 4 days rather automatically with Tony's scripts. Transfer of egamma datasets from CERN to CCIN2P3 is foreseen in September. We will try usage of bbftp for direct transfer from MSS to MSS

Tests were done for hits and digis, the transfer server was used as the Objectivity server with the same disk space. No problem to run simultaneously 5 hits jobs. Still, having only one server is the bottleneck.

During this Spring production we were clearly not using our full hardware capacity and we produced a small amount of data. This was due to missing manpower due to personal problems, combined with the fact that we had to install several versions of CMSIM. We note also that we did 2 M events for the BigMB production in winter, all ooHits being recorded in hpss. This delayed a little bit the next production. We plan to contribute more during summer 2002.

3.8 INFN

The INFN Regional Centre is a distributed Centre: CPU and disks are scattered over several sites, namely: Bari, Bologna, Catania, Firenze, Legnaro, Padova, Perugia, Pisa, Roma and Torino. The distribution of resources is such that the Legnaro site hosts about 41% of the CPUs and 35% of the total storage. Many of the other sites are new to CMS productions, this being the first time they take part.

Despite the scattering of resources, INFN is working to make itself appear as a unique Centre. During the Spring 2002 Production INFN succeeded to appear as a unique resource for the traditional part of the production, i.e. generation and simulation. The newer sites were also able to commission themselves without requiring support from the production coordinator or project manager at the CERN T0, which they greatly appreciated!

Due to limitations introduced by the computing model and by the production tools, it was not possible to completely hide the distributed nature of the INFN site. However, INFN people are working to accomplish this task for the next production cycles.

The limitations mainly come from the organisation of the data, or more specifically the metadata databases. These store production-wide information such as run numbers, dataset names, and so on. They must be unique CMS-wide, and require write-access during production, so it is not possible to split the running of one dataset across multiple sites, for example.

In order to distribute the production, Objectivity DBIDs were preassigned in such a way that the resultimg files can later be attached to a unique federation. For each site a set of metadata databases have to be preassigned too. In order to make them unique, they must have a unique name too (the so called 'owner' name). This is achieved by embedding the RC name (given by RefDB) in the owner name.

File transfer tools too showed a few problems, due to some, even minor, hardware/software/configuration differences between the farms. A more hierarchical model would imply that file transfer would be active between CERN and a reference site for INFN (The INFN T1 site): local sites should then download and upload files to and from it. This model is convenient both for efficiency and for bookkeeping, especially since file–transfer (or even dataset–transfer) is one of the largest problems currently faced by the CMS Production team.

Actually INFN people is working to implement such a model for the next production season.

In this respect, it is important that, in a near future, the access to the RefDB will be provided through command-line tools too (at the moment only a web based access is provided), in order to make it possible to develop Regional Centre tools that automatically reassign jobs to other sites. It is of course possible to use tools like 'wget' with a hand-crafted URL, but this is clumsy, and a more direct solution is needed.

3.8.1 Architectures

The INFN Regional Centre, being distributed, employs several different architectures for the farms. The architectures were chosen taking into account local expertise and the actual size and complexity of the farm. One of the purposes of having different architectures was not only to make use of local experts who would be able to fix problems in a timely manner, but also to try different configurations in order to evaluate their relative pros and cons.

There are two main categories of farm architectures: the hidden farm and the distributed farm.

The first has been used in those sites where the number of PCs is small enough. In this architecture we usually have a PC, called the gateway, which acts as a software repository and as a gateway to computing nodes. The gateway has two network cards: one connected to the public network and the other to a private, non–routable network to which computing nodes (clients) are attached.

Clients are configured in such a way that they mount, via nfs, both the home directories and the software repository directory from the gateway. The same set of users have been defined for both the gateway and the clients, namely: cmsprod, cmsadmin and httpd. The cmsprod user is the one that runs the production, while cmsadmin is the one that manages software. The production is then led by the gateway, which acts as a batch server too. Condor, LSF and PBS were used as batch systems. We are quite happy with all of them.

Usually farms host a disk server too with large RAID disks attached to it (either interally or externally) and clients

write results on them using NFS at the end of computation. During the run all I/O is done on local disks, then results are copied to the disk server, but for ORCA steps, who produced databases directly on the disk server using NFS or AMS. Sometimes the disk server coincides with the gateway.

The disk server acts as import/export server too.

The distributed configuration, instead, consists of a farm in which clients host large disks, so that it is worth storing data on them rather than using external disk servers. When the farm becomes large, in fact, the network traffic may become too heavy and for I/O intensive jobs it may represent a bottleneck. We observed that up to 15 clients per farm is still a sustainable number, but above this figure problems may arise expecially during hitformatting or digitization with pileup at high luminosity.

In distributed farms results are stored locally up to the simulation step so no network bandwidth is used to read/write data, although it is used to read pileup during digitization. In order to further reduce the network traffic, in the distributed farm there are more than 1 database server. The databases are distributed over those servers, so the probability that all the clients read or write from the same server is drastically reduced.

The Legnaro farm is a distributed farm, for which problems appeared during production, related to the complexity of the production management. The IMPALA and BOSS tools, in fact, were developed having in mind a centralized management and storage, so they were found not to be useful in distributed farms. We have then modified the tools to be able to exploit the batch-host-local storage, and the modifications have now been included in the new version of the production tools.

The main problem appears passing from one step to the following: as an example, having stored pythia ntuple on a given disk, the corresponding CMSIM job must run on the CPU hosting that disk, so that when generating the job scripts we have to take into account the physical location of the data. The same happens for the hitFormatting step. For the digitization step the association between physical and logical storage has been managed by Objectivity, so it was less problematic from this point of view, but we had to take into account the location of the pileup servers, since many of them were used, each of which had the same data replicated on their local disks, for better performances. In principle this kind of application should have been taken care of by the FTO option of Objectivity by means of the concept of 'autonomous partition', but, unfortunately, it was found not to be reliable enough.

During the next production steps, we should get rid of Objectivity, nevertheless the problems will be the same: namely the proper associations between jobs, physical storage and logical storage. We then hope to be able to use new tools helping us to manage the production in a simpler way. Ultimately, the tools should be able to interface to a 'local replica manager' to fulfill this functionality.

Another problem that arose during the production on large farms (i.e. Legnaro) was related to BOSS. We experienced severe bottlenecks during low luminosity digitization. With more than 100 hundred jobs running the access to SQL tables was very slow; for every job the time spent waiting for sql access was larger than the processing time. The suggested workaround to change configuration in order to update the BOSS DB every 10 or 20 events instead of every one would have caused the loss of the entire digi table in the BOSS DB (because of a now-removed operational limitation of BOSS), so we proceeded without changes till the end.

Moreover BOSS was not able to specify different batch queues for different types of jobs. Recently, however, new versions of both IMPALA and BOSS have been released, fixing the above-mentioned problems.

A third class of farms is the one used in Firenze, where they do not have complete control of it, since it is shared with other groups. In this case, we experienced no or very few problems in installing tools and managing the production: the architecture was flexible enough not to interfere with the software installed by others. The main problem in this case was related to the fact that, being shared, the farm cannot be tailored to the needs of everyone in terms of the operating system. Actually RedHat 7 was installed on the farm and for this version of the OS, no Objectivity license has been bought. Because of this, no ORCA steps can legally be performed on it, even if they may be technically possible.

3.8.2 Operational Experiences

The production ran almost smoothly for the most part. Problems arose at the beginning, when a new production tool (IMPALA) had to be installed in all the sites. Most of the problems were due to misunderstandings and poor documentation. Only in one case a quite deep modification of the production software was needed, due to the peculiar architecture of the Legnaro farm. The main reason, as already explained, was that IMPALA was tailored to a farm with a centralized data storage, while in Legnaro Objectivity DBs are stored on many different hosts;

moreover, all the computing nodes provide a local storage for CMKIN ntuples and fz files, thus CMSIM and ooHit jobs need to be executed on the same host where the previous step has run before.

At the beginning of digitization phase, we experienced some blocks of the GigaEthernet interface card on the pileup servers (Intel Pro 1000/T), which eventually caused the failure of all the running jobs. This problem produced quite a waste of time because it happened a couple of times during unattended runs overnight and at weekends. The operating system version used (2.2.19-6), in fact, did not contain full support for these network cards. The corrected driver was taken from later versions and recompiled on our systems.

In general more flexibility was requested for BOSS commands in order to make it possible to run batch system specific tools. Such a functionality is now available with the new version of that tool.

A further problem consisted in the access to the repository of minimum bias events and into the transfer of data to and from CERN. The availability of the data was not always guaranteed and the bandwidth was not efficiently used. Transferring the data to a central INFN site from CERN, and then redistributing them to local sites through it, has been identified as a possible solution. During the Spring 2002 production part of the data have been exchanged between local sites quite efficiently, in particular between Bologna and Legnaro, through a GARR-B connection. In this case the transfer rate was 26-28 Mbps (see fig. 3) for a total of 620 GB within 3-4 days over a weekend. We estimate an average rate of 220 GB/day sustained.

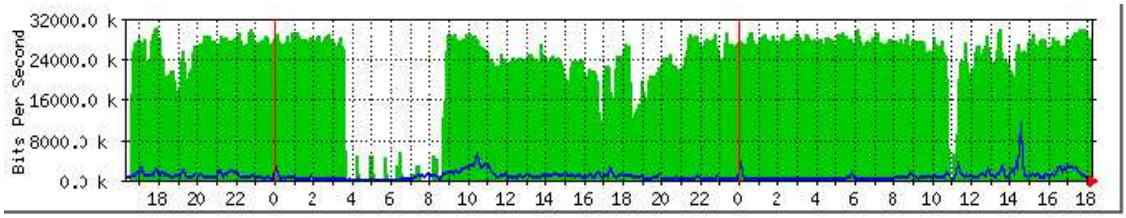


Figure 3: Transfer rate from Bologna to Legnaro (GARR-B daily graph).

Sometimes resubmission of crashed jobs failed, due to a bug in the parsing of cards in *.orcarc*, the ORCA configuration file. Of course, the bug has now been fixed, but this meant a new release of COBRA was needed.

AMS overloading caused several jobs to crash. Apparently that appears when approaching the file-handle limit within Objectivity. This problem especially happens when both production and analysis runs on the same federation. It shows that analysis must be performed on dedicated servers. When the servers are dedicated for production they can easily support 100 clients.

Moreover, about 20 % of the jobs crashed with status 139 appeared during digitization with pileup. We were not able to identify the source of this error. Resubmission of the jobs was made difficult by the lack of automatic tools, that in fact appeared when the production was approaching its end.

Another problem encountered was a bug in the Muon endcap digitisation in ORCA_6_0_1. This caused about 20% of the jobs to enter an infinite loop. A patched ORCA library was provided and tested, leading to a new release of ORCA. This example illustrates the importance of productions as a large-scale test of new versions of ORCA.

3.8.3 Summary

In short, INFN has the equivalent of 220 1 GHz CPUs and 12 TB disk. Quite a large fraction of this space is currently used to store the last production data for local usage. The whole muon database is located at Legnaro and is available to anyone who wants to do analysis on it.

The average ratio between pileup servers to clients is 1:7. Every site is capable to run every step in the production, but one, where RedHat 7.2 were installed, so no Objectivity steps can be accomplished. Local efficiencies may vary depending on the network bandwidth, available storage, etc.

A total of 1.5 M events were generated and simulated at INFN. Most of the events that will be or are being analysed in Italy were fully reconstructed at INFN too. A small fraction were sent to CERN for further processing. The whole muon database was also copied to CERN for analysis purposes.

3.9 Moscow

4 farms (see table 3.9) participated in the Spring 2002 production and generated 430 K events (CMKIN-CMSIM chain), of which 90 K events were hitformatted and digitized without pile-up.

farm	CPU	disk
SINP MSU	32 CPU	1.3 TB
ITEP	50 CPU	1.3 TB
JINR	16 CPU	
IHEP	10 CPU	

Table 4: Moscow farms facilities duting Spring 2002

Most of these events were generated in ITEP and SINP MSU, which had installed new farms and servers. In principal the same size farms were received by IHEP and JINR. But JINR had not installed the new farm yet (there are some problems with RAID) and so only 12 CPU from 16 CPU from the old farm were available for production. The same situation was in IHEP (an old farm with 10 CPU participated). All sites use the PBS sceduler.

Two large datasets (200 K events) were urgently transferred to Fermilab. The transfer was performed in two streams. One stream connected directly Moscow and FNAL and took 200 GB. Firstly we decided to make dedicated channel through DESY which connected *lhc.sinp.msu.ru* and *chimichanga.fnal.gov* directly. However this line failed after 1 day. The solution which was found thanks to Shafqat Aziz was to send data on two different computers to FNAL. The heaviest occupancy of this channel was 10 MBit/s when 60 lines worked at once.

The second stream was going to CERN from ITEP and then to Fermilab, and this had some troubles because the processes sending the data were often killed because of the total CPU time they consumed. This situation was not easily diagnosed with present tools (although it was detected by them because the file size or checksum was not correct!). It looked like the file was almost transferred and suddenly interrupted. It would be useful to include in the transferring scripts the description of this interruption. Something like: 'killed because of time limit'. Unfortunately there is probably no general solution to this, the parent of a dead process cannot portably discover who killed it or why.

Hitformatting and digitisation (without pileup) was carried out in SINP MSU for part of generated events (90 K events). All these data were replicated to CERN. In our turn we took min bias events from CERN and also some datasets digitized with low luminosity pile-up. All together we took from CERN around 90 GB in addition to the 80 GB of pileup.

The data sent to Fermilab and received from CERN was gzipped for better use of network bandwidth. Moscow now has the full sample of pileup events and so could perform digitisation with pileup, the main problem being lack of disk space.

The main speciality of our center is that we have Objectivity database distributed between two sites connected with 1Gbit/s Ethernet (SINP+ITEP). It has common catalogue located in SINP and databases are widely spread. It works properly. However some GRID tools as GDMP does not treat this really distributed database normally.

The main problem we met was connected with combined usage of IMPALA and DAR. We had a mixture of kinematics executables. Specifically:

- Executables (kine_make_ntpl.exe have the same name (or almost the same name) for all datasets and this name sometimes is not connected with dataset.
- One should create by hand the links in proper places if some special executable is needed.

At the same time there is a path to executables instantiated inside IMPALA. As soon as a misprint in localDefault.rcs appears or a link to the special executable appears in the wrong directory, the job finds the wrong executable from the default place because of the equivalent name.

So, from one side the production system pretends to be automatic and independent of the RC administrator; from the other side it demands some hand manipulation. This is dangerous because a lot of information is hidden and administrator really has no full understanding what is happening.

The issue could be to finalize automatization of procedure: all description of dataset, including name of executables (which should be different for different datasets), paths to executables and so on have to be also distributed through RefDB. Then the only thing that the administrator has to do is to type the number of the request. In the latest version of IMPALA (version 3.x) Greg has already implemented this.

3.10 UCSD

UCSD performed CMS 2002 spring production using 18 dual CPU computational nodes. Each node had two Intel 800MHz P3 CPU's and 512MB of PC133 SDRAM. The nodes were connected with fast ethernet and all systems performed reconstruction and served pile-up simultaneously. The data was output to two disk servers for a total of 2.5TB of storage: 0.5TB of high performance SCSI RAID and 2TB of 3ware 7850 based IDE RAID. The job scheduling between the nodes was handled by the PBS queueing system.

The UCSD Spring 2002 Production experience was relatively good. We were able to install and configure the CMS production software without exceeding our manpower budget. The validation samples were completed without incident and we were able to start official production quickly. Our new IDE based RAID5 data server had good performance and reliability. We utilized two techniques that improved our efficiency: the request redirection protocol (RRP) and the use of computational nodes to serve pile-up.

We found the RRP very useful in preventing the output disks from filling up. As the primary output disk filled, the database files could be copied to an auxiliary server and the RRP text file could be updated. Database files could be easily moved across the farm between high and low performance data servers with very little effort as long as a common directory structure was maintained across the servers. The RRP was also useful in balancing the load on the pile-up servers by sharing one copy of the pileup among all the servers.

The second technique was the use of computational nodes to serve pile-up. During the Fall 2000 production we prototyped this technique because we saw the CPU load was low on the dedicated pile-up servers, while the computational nodes were running slowly because they were waiting for pile-up data. The computational nodes at UCSD do not have high performance disks, but we found that by striping two low performance IDE disks together as a RAID0 partition we could saturate the network links. This allowed us to use all of our CPU's for digitization at a very good efficiency.

We had a few small problems during the production. We experienced the same digitization job failures that were reported elsewhere. The high luminosity events failed with greater frequency than low luminosity. The improvement to IMPALA of adding the resume run parameter automatically to the job file made the resubmission process fairly painless. Job resubmission was not automated because we liked to be able to look at the number of jobs that failed and the reasons why. We looked at the 'problem' area of IMPALA production tracking and copy them to the 'created' area and the corresponding entries in the 'finished' batch area.

We did not experience any hit formatting job failures during this production, though they were prevalent during the fall 2000 production.

We found that the RAM in our computational nodes was only barely sufficient for writing tracker digitization. The dual nodes have a total of 512MB of RAM, which was enough to prevent excessive swapping with two running executables, but it was clear that if the CMS software was to grow we would be required to upgrade the computational nodes. An alternative would be to run the digitisation in two steps, calorimeter first then tracker in a separate pass. This is obviously much more unpleasant from an operational viewpoint![11)]

[11)] This two-pass digitisation was the default mode of production in 2000, because the tracker digitisation code was not ready at that time. It was also used for a few Spring02 samples where the e-gamma group filtered the data based on calorimeter digits alone to get a small fraction of the events. The events that passed the filter were cloned, then tracker digits were added. The filtering and cloning makes for a more compact dataset, and not digitising the tracker for the full sample saved a lot of

The final two production problems were related to data transfer and required manpower. We found transferring data to Fermilab for archiving to be slow and painful before Michael Ernst of FNAL optimized the TCP window size for transfer. Michael deployed globus-url-copy ,which allowed us to nearly saturate our slowest network link at 80Mbit/s sustained transfer rated between UCSD and FNAL. We were able to transfer 1TB per day, which allows a remote location to serve as an efficient archiving facility. However, throughput alone is not enough, and a few files were corrupted along the way. This was not detected until the data was mirrored from Fermilab to CERN, where a different technique of attaching it to the import federations detected the corruption.

The manpower required to keep farm running efficiently was larger than we had anticipated. Keeping the farm running, making sure jobs were resubmitted, making sure the disks don't fill, and making sure the data is transferred for archiving all require someone diligent to check the system continuously. The tools make this easier, but it still requires that the farm not be left unchecked for very long.

3.11 Wisconsin

The University of Wisconsin (UW) facilities are not all configured exclusively to run CMS production. The facility is split into two divisions. They are administered independently but are linked together by Condor job management software[21].

- The bulk of the CPU resources are available in a large farm of machines at the UW Department of Computer Science (UW-CS). These machines include rack-mount systems, student laboratory workstations and some desktops. There are about 700 machines in this diffuse cluster, whereas we were promised the use of about 200 machines at any time. The jobs submitted to this farm controlled by Condor software can and do get interrupted using software signals. See [22] for details about various types of jobs that Condor can run. The most useful Condor environment is the 'Standard Universe'. In the Standard Universe, software signals cause the jobs to migrate from one machine to another and continue merrily. However, those jobs need to be specially compiled and linked using Condor supplied system libraries which are somewhat restrictive. Therefore, the Standard ORCA jobs cannot run on these machines. However, we did condor_compile the CMSIM executable so that we can take advantage of this bulk CPU. There is only a limited amount of disk available on this cluster for CMS use. However, Condor software automatically redirects the output to the job submitting hosts.
- The storage resources (approx. 3TB) for UW CMS production are located in the High Energy Physics laboratory (UW-HEP). In addition to disk servers there are a small number (approx. 10) workstations available for running user programs using Condor job queue. The UW-HEP machines are all configured so that any jobs submitted on them are automatically queued to UW-CS farm. It is also possible to restrict the jobs to non-interruptable UW-HEP machines alone.

The strategy used for CMS production is to host all of CMS software, BOSS database, etc., on the UW-HEP machines. CMSIM software was rebuilt using condor_compile so that the jobs can run on the UW-CS machines without a large penalty due to interrupted jobs. The IMPALA scripts were modified to run in this environment. The output from CMSIM was automatically redirected to UW-HEP disk servers, irrespective of where they ran, as the jobs were submitted from those machines. The ORCA hits and digis jobs were run on local CPUs only, so that the job interruptions on UW-CS do not cause excessive loss of data, and more importantly do not leave stale locks on the Objectivity federation. Since there were substantial number of changes to be made to the IMPALA/BOSS scripts coupled with the ORCA production learning curve for our new staff we were quite delayed in getting started.

Problems Faced:

1. One of the initial problems faced was having BOSS run in our environment.

   The top-working-dir for BOSS could not be on AFS, as it creates problems while creating pipes used to monitor the jobs. Since, we do not have NFS at UW-HEP we had to find a work around.

CPU from being wasted.

2. We encountered problems while using the data-transfer-tool. Using bbcp to transfer the minbias files to Wisconsin killed our machines for some unknown reasons. The same problem was seen receiving data via multiple 'scp' streams. We had to rely on sequential scp to recieve data from Fermilab and gdmp while tranferring data from hep to Fermilab. Exporting data to CERN worked rather better, bbcp was used with no observed problems and no data corruption. Our use of slightly different versions of linux tools/libraries compared to CERN/FNAL installations is perhaps responsible for this hitherto unexplained crashes of our machines. Unfortunately, we were not able to duplicate these crashes. We are going to be vigilant the next time we import large datasets to identify and fix this problem.

3. Lack of experience in trouble shooting problems with ORCA cost us some time.

4. With the initial IMPALA-BOSS structure, it was not possible to run the CMKIN or CMSIM Jobs on the Standard Universe of Condor, which was a big disadvantage as Vanilla Universe doesn't support Check-Pointing. Refer to [22] for details about various Condor universes. The reason for not being able to run on Standard Universe being "jobexecutor", which performs system calls, cannot be Condor compiled. We had to find a way to run the jobs on Standard Universe to make the best use of resources. This resulted in hacking many scripts.

5. One of the big challenges we still face is having ORCA-Digis run on Vanilla Universe Non-Dedicated machines, where a job could be evicted at any time. We need to find an elegant solution where in on resubmission(preferably without human intervention) the job could continue from the point at the time of eviction. Using the *CARF:ResumeRun* flag was suggested but it does not seem to work. Being unable to use our vast resource of Non-Dedicated condor machines for Digis we are restricted to being able to run just 15 Digis jobs at a time on our dedicated machines.

4 Statistics

Most of the numbers and tables shown here are drawn from RefDB or from BOSS tables. The entire set of BOSS tables is available on the web [23], should anyone wish to perform a more detailed analysis. Only numbers up to June 1^{st} are listed, although significant production has occurred since then[12)]. The fact that we have this detailed information is a major step forwards in our bookkeeping.

The Spring02 production resulted in about 19K Objectivity database files of various sizes. 70% of the files were over 1 GB in size, only 4% were below 10 MB. All were below 2 GB by design, apart from an accidental few at CERN (CERN is the only site to use a Solaris output server, which supports files larger than 2 GB). Datasets consisted of anything up to 500 files, but 90% of the datasets had less than 100 files. Table 7 shows the distribution of dataset sizes. Over 80% are below 50 GB, 90% are below 100 GB, with a tail up to 700 GB.[13)]

The overall production progress can be seen from figures 4, 5 and 6. Simulation started before digitisation (because ORCA was not ready then) and ran at a constant rate of 1.2 seconds per event CMS-wide for 4 months. Digitisation, when it started, was sustained at similar rates for both luminosities for two months. These plots are drawn directly from RefDB, and rely on the regional centres running their summary scripts on a regular basis. The same information was used to calculate the '*days to completion*' metric.

The start of digitisation did not slow down simulation because we were not using all the available CPUs for simulation. Both CERN and Fermilab have a significant role in testing new versions of the software (our own, plus ORCA and CMSIM) and in helping T2s to set up this software on their farms. This left them with little time for real production activity at the beginning. Our own tools evolved quite rapidly during the early part of production, making this support role that much harder.

An interesting statistic is the fraction of the work done by CERN. The CERN COCOtime comittee traditionally insists that experiments should attempt to perform one third of their Monte Carlo at CERN and two-thirds outside. If we add the number of events processed through CMSIM to those digitised at both (non-zero) luminosities and take that as the measure, we find that CERN contributed almost 40% of the data. This does not take into account the spread of CPU-times involved, so is not really the best measure of the ratio.

[12)] About another 500 assignments have been made since June 1^{st}. These are mostly ntuple-makers and ROOT-tree makers, though there are some significant samples of normal data too.

[13)] The smallest dataset consists of zero events, nothing made it through the pythia-level filter!

An alternative metric is the amount of time that batch jobs were active on a given farm, regardless of how much CPU they used, how fast the CPU was, or how much data the job produced - i.e. the wall-clock duration of all batch jobs. This measures the allocation of resources without being biased by our (in-)efficiencies in different farms or production channels, and takes into account the effort needed to re-run jobs that failed for whatever reason. By this metric, CERN actually contributed only one quarter of the total production.

The integrated batch-job wall-clock time for this production is 44 years.

Tables 5 and 6 show the numbers of events produced in different regional centres for different job-types. Duplicated runs are excluded, as are data that was never finalised (e.g. because an error was found in the requested data cards or simulation/reconstruction program after it had been made). This would make for only a small correction if it were applied.

Table 8 shows the status-code of the job for different job-types. Not all non-zero exits are serious, and many of the jobs that 'crashed' were run successfully after resubmission. For example, exit-status 8 means that COBRA intercepted and handled a particular exception condition, while most exit-status 139 digitisation jobs ran correctly the second time through, using the *CARF:ResumeRun* mechanism to avoid having to start from scratch. Nonetheless it is instructive to note that the production ran over 100K batch jobs and about 10% of them gave a nominally unsuccessful return-code !

Table 9 shows the running time (wall-clock) of successful jobs, where success is defined as having exit-status=0. Clearly simulation is the heaviest step, taking up to a few days depending on the channel. All physics channels are included here, from single-particles up to high p_t jets. Some digitisation jobs also took a very long time but mostly this will have been due to things like stale Objectivity locks on metadata causing the jobs to block (when the lock is cleaned the job continues successfully), or to delays while waiting for input datasets to be staged from tape.

Table 10 shows the number of jobs of different types that were run each week of the production, regardless of their success. For most of the production period we were running all steps in the chain in parallel. We were operating at the scale of 1000 jobs per day CMS-wide, which gives a nice target for prototype grid schedulers to aim at.

The AMS logfiles at CERN show that over 200K file-open requests per day were being made to the AMS for the production at its peak, with over 400K files on the busiest day (not counting metadata and journal files). Analysis activity was much lower by a simple count of file-opens (50K - 60K per day with a few higher peaks), but the access patterns were very different.

RC	Simulation	Hit	NoPU	2x1033PU	1034PU	Filter	NassID
Bristol/RAL	0.546	0.325	0.040	0.060	0.020	0.000	20
Caltech	0.213	0.149	0.000	0.146	0.000	0.000	6
CERN	0.887	2.196	1.396	2.656	2.266	2.890	300
Fermilab	0.345	0.405	0.000	0.251	0.332	0.000	70
Imperial College	0.878	0.588	0.495	0.147	0.121	0.000	84
IN2P3	0.200	0.000	0.000	0.000	0.000	0.000	1
INFN	1.545	1.183	0.400	0.719	0.709	0.000	99
Moscow	0.425	0.135	0.135	0.000	0.000	0.000	41
UCSD (San Diego)	0.338	0.296	0.000	0.286	0.294	0.000	80
UFL (Florida)	0.540	0.040	0.000	0.040	0.040	0.000	11
USMOP	0.000	0.000	0.000	0.000	0.000	0.000	1
Wisconsin	0.068	0.082	0.000	0.056	0.000	0.000	12
Nb of Events (M)	5.99	5.40	2.47	4.36	3.78	2.89	
Nb of Assignments	167	171	89	138	139	21	725

Table 5: For each RC and for each Production Step, for requests sent between January 1^{st} and June 1^{st}: number of events produced in millions; for each Production Step: total number of produced events in million; total number of assignments.

RC	Simulation	Hit	NoPU	2x1033PU	1034PU	Filter	NassID
Bologna	0.000	0.420	0.300	0.126	0.126	0.000	21
Legnaro	0.000	0.593	0.000	0.593	0.584	0.000	39
Pisa	0.000	0.070	0.000	0.000	0.000	0.000	6
Roma	0.000	0.100	0.100	0.000	0.000	0.000	4
Nb of Events (M)	0.000	1.183	0.400	0.719	0.710	0.000	

Table 6: For each INFN site and for each Production Step, for requests sent between January 1^{st} and June 1^{st}: number of events produced in millions; for each Production Step: total number of produced events in million. Breakdown for INFN (Simulation was done under 'INFN' name).

Size (GB)	# of datasets
0 - 50	412
50 - 100	44
100 - 150	14
150 - 200	11
200 - 250	4
250 - 300	4
300 - 350	4
350 - 400	1
600 - 650	2
650 - 700	3

Table 7: Distribution of dataset sizes for the Spring02 production. Hitformatted samples count as one dataset, as do each of their digitisations, so each physics channel contributes several datasets.

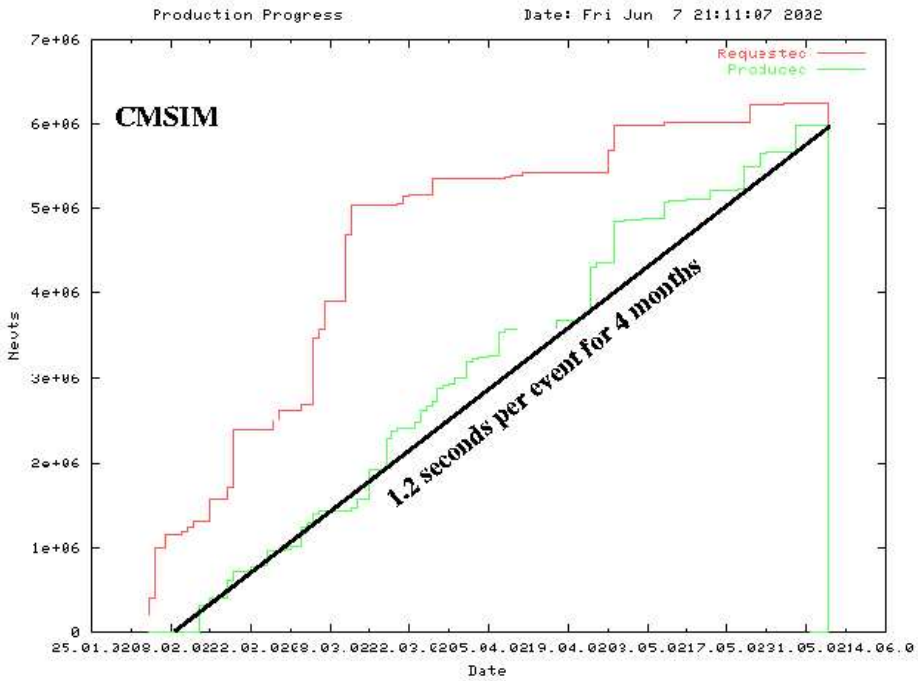


Figure 4: CMSIM progress from February 1^{st} to June 1^{st}. The upper line is the number of requested events, the lower line is the number of produced events. The line was fitted by the Weller method.

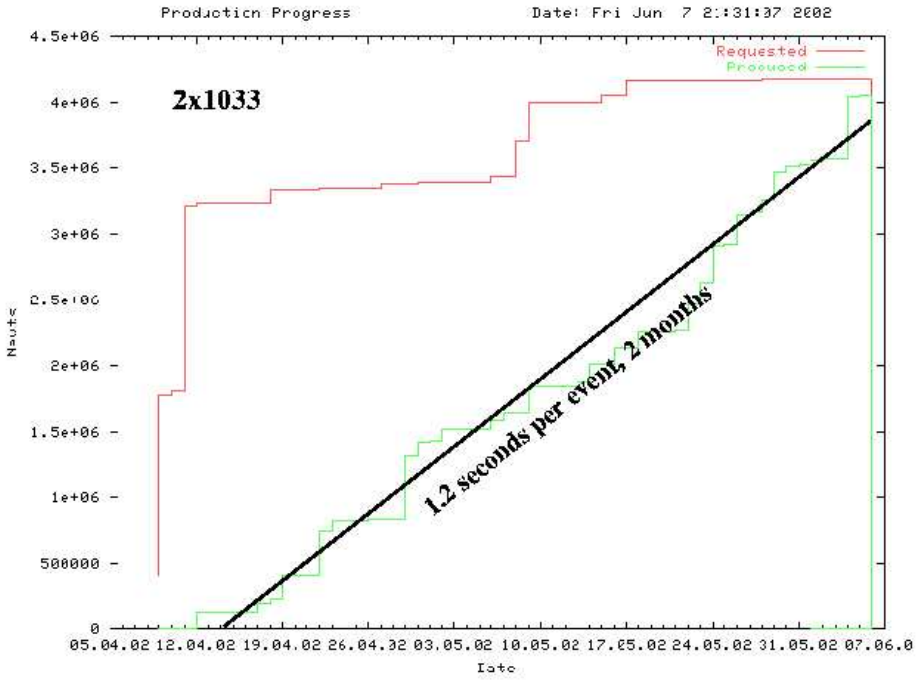


Figure 5: Low-luminosity digitisation progress from April 1^{st} to June 1^{st}.

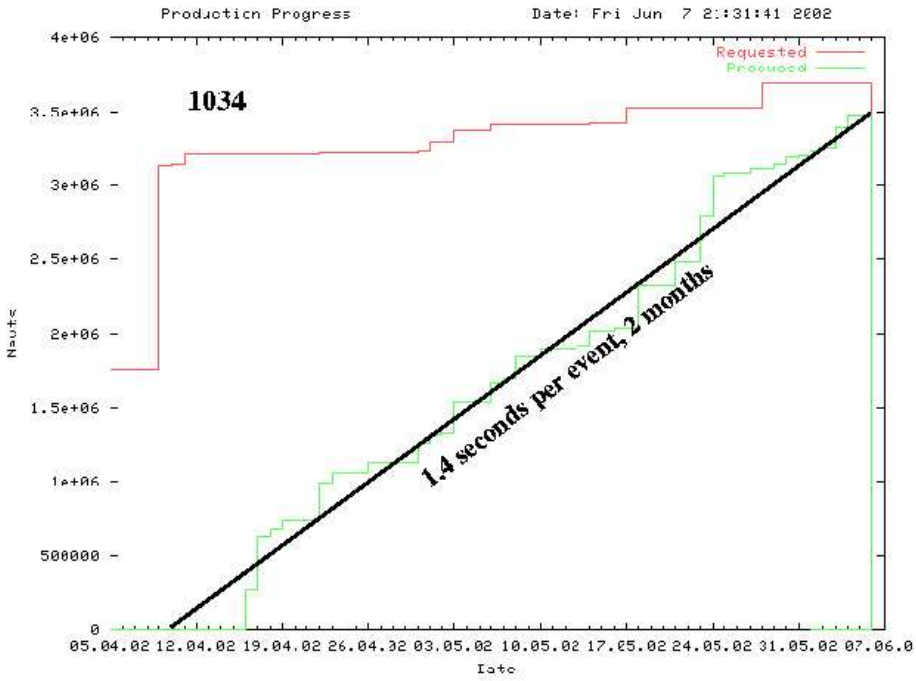


Figure 6: High-luminosity digitisation progress from April 1^{st} to June 1^{st}.

status	Total	kin	sim	oohit	oodigi
0	89577	17418	17640	18629	35890
1	4016	892	257	520	2347
2	214	-	-	10	204
7	2	-	2	-	-
8	3948	-	-	937	3011
126	11	10	-	-	1
127	1825	1318	481	3	23
129	2	-	1	1	-
132	21	2	5	10	4
134	60	-	2	-	58
136	9	-	-	3	6
137	31	-	6	-	25
138	2	1	-	1	-
139	4285	38	-	73	4174
143	35	-	-	10	25

Table 8: Job exit-status for different job-types

bin number	Total	kin	sim	oohit	oodigi
0	68195	16236	3974	18123	29862
1	8122	950	3744	337	3091
2	3695	6	2043	35	1611
3	2619	86	1592	4	937
4	2282	129	1753	130	270
5	584	11	525	-	48
6	580	-	571	-	9
7	895	-	894	-	1
8	681	-	681	-	-
9	376	-	329	-	47
10	559	-	557	-	2
11	363	-	361	-	2
12	314	-	312	-	2
13	132	-	132	-	-
14	42	-	34	-	8
15	45	-	45	-	-
16	32	-	32	-	-
17	10	-	10	-	-
18	10	-	10	-	-
19	18	-	18	-	-
20	8	-	8	-	-
21	4	-	4	-	-
23	4	-	4	-	-
24	6	-	6	-	-
25	1	-	1	-	-

Table 9: Job duration for different job-types, bins of 14400 seconds, successful jobs only

week number	Total	kin	sim	oohit	oodigi
0	82	50	32	-	-
1	556	372	184	-	-
2	5116	2169	2490	446	11
3	5059	1395	2175	1105	384
4	6141	3656	1222	318	945
5	5654	2765	980	845	1064
6	3055	926	1336	155	638
7	5041	2025	1126	1048	842
8	4667	1416	833	1589	829
9	8612	1389	216	3439	3568
10	6597	651	1652	1991	2303
11	9304	155	2130	2849	4170
12	12183	2337	1668	1886	6292
13	9921	6	1166	1670	7079
14	6792	307	226	320	5939
15	5304	-	892	1627	2785
16	9195	60	66	856	8213
17	759	-	-	53	706

Table 10: Jobs per week for different job-types

5 Summary and conclusions

The Spring 2002 production was a remarkable success compared to previous productions. Nearly all of the data was delivered on time, despite considerable delays with respect to the original schedule. The sustained production of simulated data (over 4 months) and digitised data (over 2 months) shows that we are clearly far more mature than we were in 2001. Had ORCA 6.0 been ready earlier, we would probably have delivered the data that much earlier. This must not be taken as a criticism of ORCA. Without that delay for improving the quality of CMSIM and ORCA the data we would have produced would have been useless. Also, some of the problems were only found by production-scale tests of pre-release ORCA versions, a feature that was never exploited before.

The fact that production ran at such a constant rate despite many problems at the regional centres shows that we have reached the scale where the (hopefully temporary) loss of one regional centre does not affect the global rate of production.

The amount of data produced at Fermilab and CERN was surprisingly low compared to the amount produced elsewhere. This is because other regional centres are maturing rapidly as production sites, and also reflects the fact that Fermilab and CERN are taking stronger supporting roles for the new RCs. The ability to rely on T2s to deliver data allowed CERN and Fermilab to devote more time to commissioning other centres than would have been the case if they were critical to the production itself.

Nonetheless, there are clearly significant problems with the existing production environment:

- The manpower required for this production compared to previous productions is huge. There is a long way to go in terms of automating our environment, but that will only take us so far. We must reduce the workload of people with central support and coordination roles, as well as the effort required to manage CPU farms, if we expect to reach larger scales of production.
- Installing the production machinery from scratch on a new farm currently requires a fair amount of system-level knowledge to configure MySQL databases, filehandle limits on AMS and Lockservers, RRP tables, pileup-sharing, import-export tools etc. The variety of hardware and system software configurations makes it difficult to commission new RCs. This 'proximity' to the system suggests it may be difficult to share farms with other users in a grid-enabled environment, or that the entry-point for successful sharing of resources may take a lot more effort to reach than currently thought. We cannot simply 'submit a job to the grid' yet.
- The manipulations we perform on our data and metadata are very complex. There is a lot of state information that is relevant to each dataset and to each file in the dataset, and controlling it well requires care. The 'state-machine' of a dataset is such that it is essentially impossible to deal with a fraction of a dataset at a time, or to distribute a dataset across several production centres. With dataset sizes approaching 1 TB this is a real problem, many RCs experienced delays continuing production because they were waiting for previous data to be shipped elsewhere.
- Large datasets are one end of the problem, but small datasets are problematic too. It is easy to automate all the steps in the production chain if they work well. It is almost impossible to deal with them automatically if they fail, simply because of the complex failure modes (and thus the difficulty of automating remedial action, if any such action exists). Small datasets can consume a great deal of manpower.
- File-transfer is still a serious problem. We are clearly nowhere near having a CMS-wide solution that allows us to transfer files as efficiently as possible between all participating sites. With significant production taking place at RCs with relatively poor network capability and with more RCs coming online everywhere this needs a lot of work. The biggest problem with existing candidate solutions seems to be the universality of the authentication mechanism (or lack thereof) which make them difficult to deploy.
- File-transfer is one aspect of the problem of dataset-transfer. Until an entire dataset is available it is useless to CMS (partly because of our data-storage model, admittedly). There are people working on efficient MSS-to-MSS WAN-based file-transfers, but not with the added complication of ensuring that the entire dataset is read from tape efficienlty at the source and written to tape in a manner which is optimal for analysis at the destination.
- Job-resubmission has emerged as an important issue. During simulation, with a classic one-file-in, one-file-out scenario, it doesn't matter if a job dies; you resubmit the job until it works or you produce new jobs to replace the ones that fail. During digitisation job-failure is more serious, because the events are already in an object-database and you have to know which ones were done and which ones were not. Clearly different

centres have found different solutions that worked for them, it is not obvious which is best under the current circumstances.

Note that the idea of 'resubmitting a job' is tied to the idea that one job will process one input run. If a job fails for some reason then the resubmitted job can be told to process from the point just after the last job on the same run committed its data[14]. This artificial constraint of one run per job may be clouding our view of the way to build better tools. Job-decomposition based on expected running time or number of available CPUs might be a better approach, but this has deeper implications for both RefDB and Impala at least.

- Many of the issues relating to the difference in scale between file-level and dataset-level manipulation stem from the fact that our data is written, during production, in a way that we believe to be optimal for serving, during analysis. This is perhaps too rigid, and a more flexible and scaleable approach would be to allow production to write monolithic output, isolated from the knowledge of other production batch jobs, and then 'upload' that data into the event store in a reformatting step. Put simply, production output need not directly resemble analysis input if another step is allowed in the chain.

- The use of tracking databases has been clearly demonstrated to be of great value. The Spring02 production used one (BOSS) database per regional centre plus one central (RefDB) database. This has worked very well, and without it the statistics presented in this paper would not have been available. Not only would production have been harder, we would not have been able to measure how much harder it would have been. Nonetheless there are serious problems of scalability and reliability that need to be solved. We prefer that the tracking database be updated in real-time for online monitoring purposes, but we cannot rely on the database being updated at all in a hostile or overloaded network environment.

- Individual batch jobs are not well-enough behaved that a farm can be left running unattended for several days. Even if we improve our software it is unlikely we can prevent problems like this at source, because problems may be caused on the batch nodes themselves (because of sharing with other experiments) or in the farm management (e.g. because of batch systems losing track of jobs).

- Although many problems have been exacerbated by our current data-storage model it is not reasonable to assume that they will disappear with the demise of Objectivity. It is our use of Objectivity that leads to this behaviour, and its replacement will presumably be flexible enough to support the same set of problems. It is in our interest to avoid this, and that means a change in our data-storage model.

Many of these problems are already being addressed by the production team and the Production Tools Review [24], and will help drive the next iteration of tool development between now and DC04. Together with help from the grid projects we are re-examining the component breakdown of our tools to see where it can be improved, and to see where we can hope to make maximum use of suitably mature grid products in the future.

The timescale for improvement is short. The scale of DC04 currently requires a pre-DC04 production for 5 months continuously at about twice the rate of the Spring 2002 production, to be performed in the second half of 2003. This is not feasible without a significant reduction in the manpower needed to support and run the production CMS-wide, which in turn means we need much more robust and automated systems which are well-tuned to our specific needs.

6 Acknowledgements

The Spring02 production would not have been successful without the effort of a large team of people running and supporting the production in various ways. We wish to thank in particular the following people:

Nicola Amapane, Julia Andreeva, Shafqat Aziz, Massimo Biasotto, Daniele Bonacorsi, Dimitri Bourilkov, Claude Charlot, Pamela Chumney, Dave Colling, Salavatore Costa, Maria Damato, Sridhara Dasu, Michael Ernst, Alessandra Fanfani, Livio Fano, Alexei Filine, Ian Fisk, Simone Gennai, Radhakrishna Gowrishankara, Greg Graham, Claudio Grandi, Vincenzo Innocente, Werner Jank, Olga Kodolova, Victor Kolosov, Nikolai Kruglov, Alexander Kryukov, Stefano Lacaprara, James Letts, Philip Lewis, Vladimir Litvine, Catherine Mackay, Nicolo Magini, Paolo Meridiani, Philippe Mine, Shahzad Muzaffar, Dave Newbold, Giovanni Organtini, Asif Osman, Nathalia Ratnikova, Jorge Rodriguez, Leonello Servoli, Nick Sinanis, Suresh Singh, Wesley Smith, Elena Tikhonenko, Yujun Wu.

[14] Is it really 'the same job' if it starts from a different place in the input dataset?

With a list of people this long it is highly likely that someone has been omitted by mistake, for which we apologise in advance.

References

[1] **CMSIM** , http://cmsdoc.cern.ch/cmsim/cmsim.html,
Veikko Karimäki
"CMSIM: CMS Simulation and Reconstruction Package ".

[2] **ORCA/COBRA** , http://cmsdoc.cern.ch/orca/
"CMS OO Reconstruction".

[3] **BOSS** , http://www.bo.infn.it/cms/computing/BOSS/,
Claudio Grandi
"BOSS (Batch Object Submission System) ".

[4] **DAR** , http://computing.fnal.gov/cms/software/,
Natalia Ratnikova
"Software Distribution for CMS MC Production".

[5] **RefDB**, CMS IN-2002/044 "RefDB",
Veronique Lefébure, Julia Andreeva.

[6] **IMPALA**, http://computing.fnal.gov/cms/Monitor/docs/cms_documents/ProductionSoftware/CMS-PSOFT-002/cms-psoft-002.html,
Greg Graham
"CMS-PSOFT-002: IMPALA Architecture ".

[7] **The CMS Grid Production-tools workshop** http://documents.cern.ch/AGE/current/fullAgenda.php?ida=a02826.

[8] **PYTHIA** , http://www.thep.lu.se/ torbjorn/Pythia.html,
Torbjörn Sjöstrand
"PYTHIA (and JETSET) webpage".

[9] **The CMS Production Project home page**, http://cmsdoc.cern.ch/cms/production/www/html/general/index.html
Veronique Lefébure

[10] **IMPALA and BOSS Production FAQ**, http://cmsdoc.cern.ch/swdev/fom/prod/cache/1.html

[11] **CALTECH Farm monitor**, http://www.cacr.caltech.edu/projects/tier2-support/
Jan Lindheim and Suresh Singh.

[12] **Tony's Scripts**, http://wildish.home.cern.ch/wildish/Objectivity/scripts.html
Tony Wildish
Scritps for transferring bulk data.

[13] http://wildish.home.cern.ch/wildish/BoughtTheFarm/BoughtTheFarm.html,
Tony Wildish
Configuring farms for ORCA productions.

[14] hep-ex/0112003, D. A. Sanders et al. *Redundant Arrays of IDE Drives*

[15] http://wildish.home.cern.ch/wildish/BoughtTheFarm/SoftwareConfigurations.html#RRP,
Tony Wildish
Configuring the AMS Request Redirection Protocol.

[16] **CASTOR**, http://it-div-ds.web.cern.ch/it-div-ds/HSM/CASTOR/,
CERN Advanced STORage Manager.

[17] **BBCP**, http://www.slac.stanford.edu/ abh/bbcp/,
Andrew Hanushevsky
The BaBar Copy Program

[18] **ENSTORE**, http://hppc.fnal.gov/enstore/design.html,
Enstore Technical Design Document

[19] **dCache**, http://www-dcache.desy.de/summaryIndex.html,

[20] **FBSNG**, Computer Physics Communications, Vol 140/1-2, pp. 253-265

[21] **CONDOR**, http://www.cs.wisc.edu/condor

[22] **CONDOR job types**, http://www.cs.wisc.edu/condor/manual/v6.4/2_4Road_map_running.html

[23] **BOSS table archive**, http://wildish.home.cern.ch/BossTables/BossTables.tar,
archive of BOSS tables from the Spring02 production

[24] Production Tools Review,
Not yet published, but will be linked from the CERN Production Page.
A.Afaq, M.Biasotto, R.Cavanaugh, D.Colling, P.Couvares, C.Grandi