<LRH>10. Distributed Data Analysis</LRH>

COMP: Please use most recent H1 head (i.e., before the end of each recto page), including section number, for each RRH throughout Chapter 10.

<RRH></RRH>

<CN>Chapter 10</CN>

<ATL>Distributed Data Analysis: Federated Computing for High-Energy Physics</ATL>

<AU>Greg Graham, Richard Cavanaugh, Peter Couvares, Alan DeSmet, and Miron Livny</AU>

<TXT>CMS (Compact Muon Solenoid) is a high-energy physics detector planned for the Large Hadron Collider (LHC) at the European Center for Nuclear Research (CERN) near Geneva, Switzerland. CMS is currently under construction and is expected to be completed in 2007, at which time it will begin to record data from the highest-energy proton–proton collisions ("events") yet produced. Data from these collisions will shed light on many fundamental scientific issues, including a definitive search for the Higgs particle and the possible origin of mass in the universe, the existence of a new fundamental symmetry of nature called supersymmetry, and even the possible discovery of new spatial dimensions. The data will contain information from potentially millions of individual elements within the detector itself, which will be used to reconstruct the actual collision. Even though these data will be filtered online before analysis, it is still expected that CMS will produce up to several petabytes of data per year.

Although the CMS detector will not begin taking data until after 2007, hundreds of physicists around the world, members of the CMS collaboration, are currently taking part in compute-intensive Monte Carlo simulation studies of the detector and its potential for uncovering new physics. Monte Carlo simulation studies integrate clean theoretical predictions of underlying physics against all of the efficiencies and electronic noise of the millions of detector elements in order to produce realistic simulation data. This simulation data can be used to help predict the impact of detector design on discovery potential. Once the CMS detector is functioning, the output of simulation studies will be compared directly against actual data. Such comparisons provide improved detector calibrations, measurements of physical processes, and indications of possible scientific discoveries.

The scientists and institutions participating in the CMS collaboration are located throughout the world. These scientists are not expected to live at CERN for the duration of its expected 15-year lifetime, but rather need to make significant contributions to the scientific process "at a distance." Therefore, even before the completion of the CMS detector, and then throughout its lifetime, there will be a need to knit scientists together worldwide and put large heterogeneous worldwide-distributed institutional compute and storage resources at their disposal in an organized way. In this regard, Grid technology has shown great promise to:

- <BL>manage effectively the addition and removal of heterogeneous institutional resources in a virtual organization that changes with time;
- expose these resources to the entire worldwide collaboration in a consistent set or protocols and APIs; and
- provide mechanisms to control and to optimize the enormous flow of data from the CMS detector to scientists working around the world.</BL>

In order to address these issues, the U.S. participants in the CMS collaboration (US-CMS) began exploring Grid technology in the autumn of 2002 to accomplish an official production request of Monte Carlo simulation data. One of these efforts is the US-CMS Grid. Participating Grid sites include the California Institute of Technology; the Fermi National Accelerator Laboratory; the University of California, San Diego; the University of Florida, and the University of Wisconsin, Madison. For a period of time, a group from CERN also joined the US-CMS Grid effort. Table 10.1 **Error! Reference source not found.**shows the resources of the US-CMS Grid (90). Sites were linked by high-bandwidth Internet connections, typically OC12 or higher, with gigabit or 100-Mbit connections to each computer. Our goal was to complete an assignment of 1 million "events" requiring roughly 200,000 CPU-h in a 60-day timeframe. This is roughly the amount of time it took for these same sites to complete an assignment of this size in the past, managing their own computations using existing non-Grid technology.

ED/AU: "Table 10.1" as meant in above para.?

<TN>Table 10.1</TN><TT> US-CMS Grid Resources</TT>

<TCH1>Site</TCH1>	**<TCH1>Number of worker CPUs</TCH1>**
<TB>Caltech	40 (0.75 GHz)
	40 (2.4 GHz)
Fermilab	80 (0.85 GHz)
University of Florida	80 (1 GHz)
UC San Diego	40 (0.75 GHz)
	40 (2.4 GHz)
CERN	72 (2.4 GHz)
UW Madison	5 (0.85 GHz)</TB>

The participating sites are typically organized as cluster farms with server nodes and worker nodes. The worker nodes were either on the public Internet or behind a NAT firewall.

<H1>10.1 Implementation</H1>

We chose to base the US-CMS Grid upon the basic functionality provided in early versions of the GriPhyN Virtual Data Toolkit, which is in turn based upon the Globus Toolkit (276) (Chapter 4) and the Condor High-Throughput Computing System (446) (Chapter 19), including the Condor-G (292) job submission interface to the Globus Toolkit. In addition, we employed rudimentary software to manage the US-CMS Virtual Organization. This approach toward basic middleware functionality allowed rapid middleware deployment and facilitated a relatively high level of fault tolerance by reducing the variety of possible failure modes.

<H2>10.1.1 The Virtual Data Toolkit</H2>

The Virtual Data Toolkit (VDT) is produced by GriPhyN (93), and includes the core Grid middleware necessary to deploy and operate a computational Grid. In addition, the VDT employs a packaging manager known as Pacman for automated installation of Grid middleware to the various US-CMS Grid sites. Once installed, the Grid site administrator is able to manually configure the VDT to fit the appropriate local compute cluster architecture.

The US-CMS Grid evolved over several VDT releases, each time providing useful scalability testing information back to the middleware developers. We relied on the VDT for all Grid components, which allowed for simple and consistent management of the middleware across the entire Grid.

This is not to say that Grid middleware deployment was never problematic. There were problems from time to time with low operating system resource defaults for file handles and inodes, for example. There were also problems with unreliable file transfers. However, none of these problems proved to be showstoppers, and more importantly all provided important feedback to the middleware developers themselves.

<TN>Table 10.2</TN><TT> Software from the VDT 1.1.3, Currently Installed on the US-CMS Grid</TT>

<TCH1>Virtual data toolkit components</TCH1>	<TCH1>Version</TCH1>	<TCH1>Comments</TCH1>
<TB>Server		
Globus Tookit	2.0	Modified GASS cache/jobmanager
Condor	6.4.3	Includes DAGMan
Fault Tolerant Shell	0.99	Provided fault tolerant data transfers
Client		
Globus Clients	2.0	GSI, GridFTP
Condor-G	6.4.3	—</TB>

The particular version deployed on the US-CMS Grid for the production run described here was VDT 1.1.3 and 1.1.4, which included core client and server components from the Globus Toolkit and Condor (see Table 10.2**Error! Reference source not found.**).

<H2>10.1.2 CMS Specific Software</H2>

The physics simulation software used by CMS is complicated and has evolved over years—in some cases decades—to embody a great deal of accumulated knowledge and problem-solving experience. Furthermore, it has taken time for scientists to trust the core software to perform correctly. For these reasons, it was important to adapt existing simulation software to the Grid as much as possible, rather than rewrite it from the ground up as a Grid application.

This approach presented challenges, however. Specifically, past practice had been to run the software in much smaller and more controlled environments than the Grid. For example, shared-file systems and common user databases were assumed to exist between submission and execution machines and the necessary software was assumed to be installed locally beforehand. Also, the standard methodology for running the CMS software had evolved over time from systems managed by hand on individual computers by a few researchers or small clusters of loosely managed computers, to large batch systems utilizing large clusters of largely homogenous resources.

Several layers of management software had therefore been written to help automate the process of running the multiple computations in order on multiple computers and organizing the results. In CMS, this included a legacy, Bash script based, job-tracking system (IMPALA), which provided a relatively robust system for declaring, creating, executing, and tracking large numbers of individual jobs through a variety of locally resident batch systems. The more recent MCRunjob (319) package provides a metadata-based approach for specifying more complex workflow patterns, translating them into a set of submittable jobs in a variety of environments, including virtual data language, DAGMan directed acyclic graphs, and the legacy IMPALA environment.

In order to get early buy-in from CMS, we utilized as much of the existing scientific and production-management software as possible, while enabling it to run on the Grid. This approach enabled direct comparisons between Grid and non-Grid methods. To produce these results as quickly as possible, we chose to insert an adapter layer of software into the existing system, called MOP, and reengineer the existing layers as little as possible. See Table 10.3**Error! Reference source not found.**.

<TN>Table 10.3 </TN><TT> Post-Grid Software Layers</TT>

<TB>Condor/FBS	Local site batch system
Globus Toolkit	Security, I/O, GRAM resource allocation protocol and services, GridFTP
Condor-G	Grid job management
DAGMan	Job dependency management
MOP	Grid "wrapper" generation for non-Grid jobs
IMPALA/MCRunJob	Job creation layer
CMSIM	Physics simulation code</TB>

Actual Monte Carlo production depends most critically on the size of each "event" at the CMKIN stage (which simulates the "event"): the more by-product particles produced after the initial proton–proton collision translate into higher processing times for later stages of computation. The CMSIM stage simulates the CMS detector's response to the particles produced in the CMKIN stage and is the most CPU intensive of all stages. CMS Monte Carlo production consists of pipelining several stages together where the output of one stage serves as the input to the next (434). The longest stages are typically CPU-bound, but some are I/O-bound, and some vary depending on the data. Table 10.4**Error! Reference source not found.** summarizes the typical characteristics of the stages used in CMS Monte Carlo production.

<TN>Table 10.4 </TN><TT> CMS Computation Stages and Their Typical Characteristics (Approximate)</TT>

<TCH1>Step</TCH1>	<TCH1>CPU time (s/event)</TCH1>	<TCH1>Output size (Mb/event)</TCH1>	<TCH1>Bound</TCH1>
<TB>Stage 1 (CMKIN)	0.05	0.05	CPU
Stage 2 (CMSIM)	350	2.0	CPU
Stage 3 (writeHits)	0.05	1.0	I/O
Stage 4a (writeDigis No-PU)	2.0	0.3	CPU
Stage 4b (writeDigis 10^{34} PU)	10.0	3.0	CPU and I/O
Stage 5 (ntuple)	< 1	0.05	CPU and I/O</TB>

<TFTN>*Note*: The overall results can be highly variable depending on the physics process being simulated.</TFTN>

Quality assurance considerations require that all productions run uniformly and utilize specific versions of the CMS software. In order to create a "sandbox" environment for the CMS binary executables, a distribution after release (DAR) packaging and deployment mechanism was developed for CMS software. DAR bundles all shared object libraries (including any necessary gcc libraries) along with scripts for setting up the necessary environment variables for job execution. The DAR release version corresponding to the particular production run described in Section 10.2 was then uniformly preinstalled across all US-CMS Grid sites.

CMS Monte Carlo production normally proceeds by breaking up production requests into 250-event collections and processing each collection serially through all stages. For the US-CMS Grid production during autumn 2002, there were two requests for events. The first request was for 1 million events processed through all steps. The second request was for 500,000 events processed only through the CMSIM stage.

<H2>10.1.3 Integration Software—MOP</H2>

MOP (short for Monte Carlo Production) is a "Grid adapter" developed for CMS that sits between the job creation step and the Grid middleware in the Virtual Data Toolkit, and adds

necessary subtasks to each job to enable it to run on the Grid without modification. As such, MOP provided a Grid interface very much similar to that of a traditional batch system.

ED/AU: Change in above para. OK? Or is "added" an adjective, as in "the [necessarily] added subtasks"? If the latter, please clarify the sentence.

The jobs, as produced for the US-CMS Grid production run, were not themselves particularly "Grid aware." MOP represented each generated job as directed acyclic graphs (DAGs), using four DAG node types: stage-in nodes to transport the execution environment to the worker node, run nodes to run the executables on remote resources using the Globus GRAM interface, stage-out nodes to transport results back to the submit site, and cleanup nodes to remove any leftover job state from the worker nodes. From the standpoint of the CMS software, the jobs are still "local jobs" and MOP takes care of the Grid issues of staging, data transfer, and cleanup.

During production Grid runs, MOP was invoked to create DAG representations of each job at submit time. Once a DAG was produced, MOP submitted the DAG to the DAGMan package of Condor, which ran the DAG nodes using the Condor-G gateway, allowing DAGMan to run DAG nodes on remote compute sites running Globus job-managers. In turn, these Globus job-managers are able to run the jobs using local batch queues.

<H2>10.1.4 Virtual Organization</H2>

Much like local networks of machines, worldwide networks of Grid resources require some kind of centralized user database management. The Globus Toolkit provides a local mechanism for each system to map Grid user certificates to local users (280) (Chapter 21), but provides no way to synchronize or automatically distribute this information between multiple systems.

In order to automate the process of adding and removing users from the US-CMS Grid, we used the Caltech Virtual Organization Group Manager, which stores the user information in a central LDAP database and allows an administrator to create groups and populate users.

<H1>10.2 The Production Run</H1>

In large collaborative environments, such as that associated with the CMS experiment, the stress of running large-scale Monte Carlo productions can approach stresses not encountered anywhere outside of the running of the actual experiment. The consequences of failure in large-scale Monte Carlo production do not approach those associated with the loss of actual data. However, they include missing important deadlines set by funding agencies, failure to validate fundamental computing models, and in the era of data taking they include the possibility of falling behind competitors in the race for scientific discoveries. In this modern age of high-energy particle physics, computing is seen more and more as a critical extension to detectors themselves.

The emerging US-CMS Grid entered this highly charged environment during the spring 2002 CMS Monte Carlo production, in support of the technical design report of the data acquisition system, and quickly fell to its knees! Close inspection of the middleware revealed that although the underlying Grid computing model was sound, several key components were lacking in implementation. After a six-month period of reengineering, the US-CMS Grid reemerged in the autumn of 2002. After breezing through an initial 50,000-event test run in September, the US-CMS Grid was ready to participate in a 10 million-event study of the backgrounds in the CMS detector. The part assigned to the US-CMS Grid consisted of a 1 million-event request processed through all steps **Error! Reference source not found.**(excepting pileup) plus a 500,000-event request to be processed through the CMSIM stage only.

CMS Monte Carlo production is highly organized. Conveners of special purpose groups (organized around specific physics topics) enter production requests into a reference database at CERN. Production staff at CERN review new requests and break them up into smaller parts. Each part is assigned to a participating CMS regional center. Regional centers receive requests for production by e-mail, with each request including a key into the assigned part of the production in the reference database.

The US-CMS Grid was set up as a virtual regional center in order to participate in the CMS production. After receipt of an e-mail request, the CMS job creation tools were invoked with the given key. The tools then contacted the reference database at CERN and downloaded all necessary parameters via HTTP. Each created job consisted of executable scripts with parameters to generate 250 events (using the CMKIN stage) and process them through the CMSIM, writeHits, writeDigis(NoPU), and ntuple making stages for the 1 million-event request; MCRunJob also create a different, simpler script performing just the CMKIN and CMSIM steps for the 500,000-event request.

During the running of the US-CMS Grid, the operator would typically generate a few hundred jobs at a time and assign them to different Grid sites by hand. This approach did not attempt to use a scheduler or resource broker because we felt that there were still lessons to be learned lurking in the middleware itself. Job submission involved the invocation of MOP, which took the job scripts and wrapped them into DAG nodes as described previously. DAGMan then took these DAGs and, using Condor-G as a backend, was able to run the DAG nodes on remote Globus job-managers. In the US-CMS Grid, these job-managers were configured to use either regular Condor or the Fermilab Farm Batch System as queue managers on local clusters. Information and job output were sent back to the submit site by the "stage-out" DAG node.

We encountered many problems during the run and fixed many of them, including integration issues arising from the integration of legacy CMS software tools with Grid tools, bottlenecks arising from operating system limitations, and bugs in both the Grid middleware and application software.

Every component of the software contributed to the overall "problem count" in some way. However, we found that with the current level of functionality, we were able to operate the US-CMS Grid with 1.0 FTE effort during quiescent times over and above normal system administration and up to 2.5 FTE during crises. This compares favorably with the official spring 2002 Monte Carlo production of CMS, but concrete comparisons are hard to draw because of the looser organization of the spring 2002 effort, as discussed in the next section. The following are examples of the problems encountered:

- <BL>(Pre-Grid) During spring 2002, the Globus 2.0 GASS Cache (117) was found to not support the required level of performance for CMS production. The software was reengineered in consultation with Condor developers and Globus developers over the summer of 2002, and released in Globus 2.2.

- It was found that many simultaneous globus-url-copy operations originating from the MOP master site when submitting many jobs would cause some globus-url-copy operations to hang. Globus-url-copy operations were wrapped in Fault Tolerant Shell (FTSH) scripts. FTSH contains semantics to time out and retry shell commands; we found that it could be applied to many other places to add fault tolerance to existing applications (see Chapter 19).

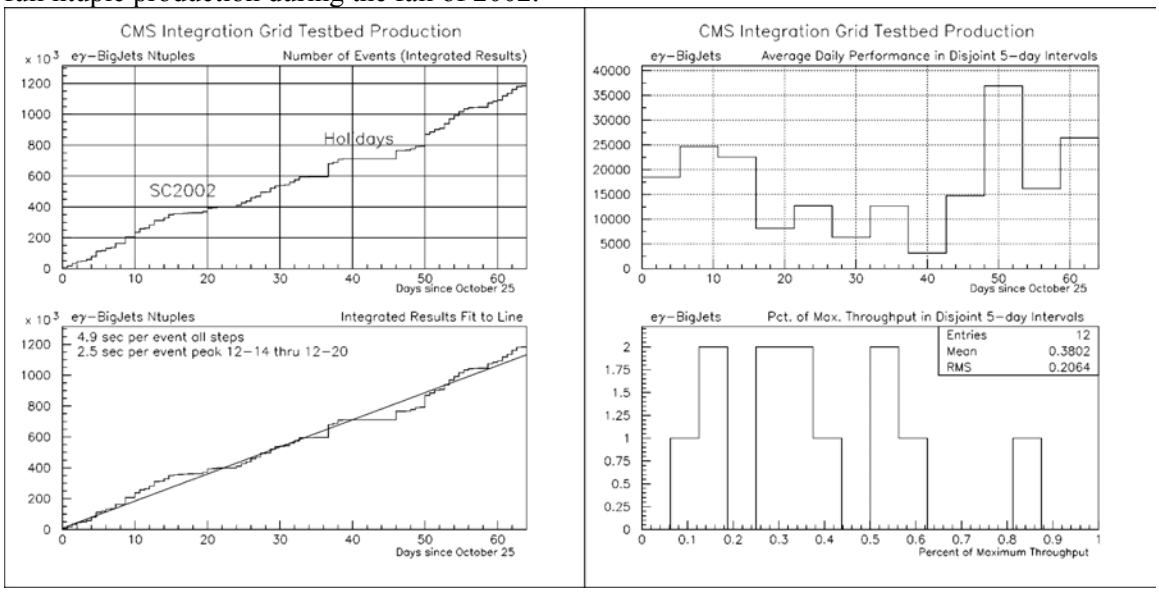
- Condor was configured to resubmit failed jobs in some instances. We did not have sophisticated problem-tracking tools during this run, and therefore there was often an inability to realize that something was wrong when problems were occurring.

- Jobs sometimes failed due to application code problems. During one episode in November, middleware was suspected of causing disk cache overruns. A "War Room" of middleware developers was organized to create a problem tree and explore all of its branches. Eventually, after three days, the bug was traced to incorrect but innocuous-looking program input from the job creation step and given to a developer of the job creation software, after which the problem was diagnosed and fixed within 90 minutes. More sophisticated error analysis is needed to sort bugs correctly to the right people.

- Condor-G running on the MOP Master site uses a "gahp_server" to handle its communication with processes running under Globus on remote worker sites, one thread per tracked process. With over 400 CPUs available to the US-CMS Grid at later stages of production, running two assignments to produce 1.5 million events, we had to divide production over two physically separate MOP master machines, to avoid the scaling limit of the number of gahp_server threads.</BL>

<H1>10.3 Conclusions</H1>

The US-CMS Grid was a success in that it produced all of the required events and provided many useful insights into operating a Grid in production mode (86). Also, many problems were uncovered with the software at all levels. Figure 10.1 shows the progress of the US-CMS Grid full ntuple production during the fall of 2002.

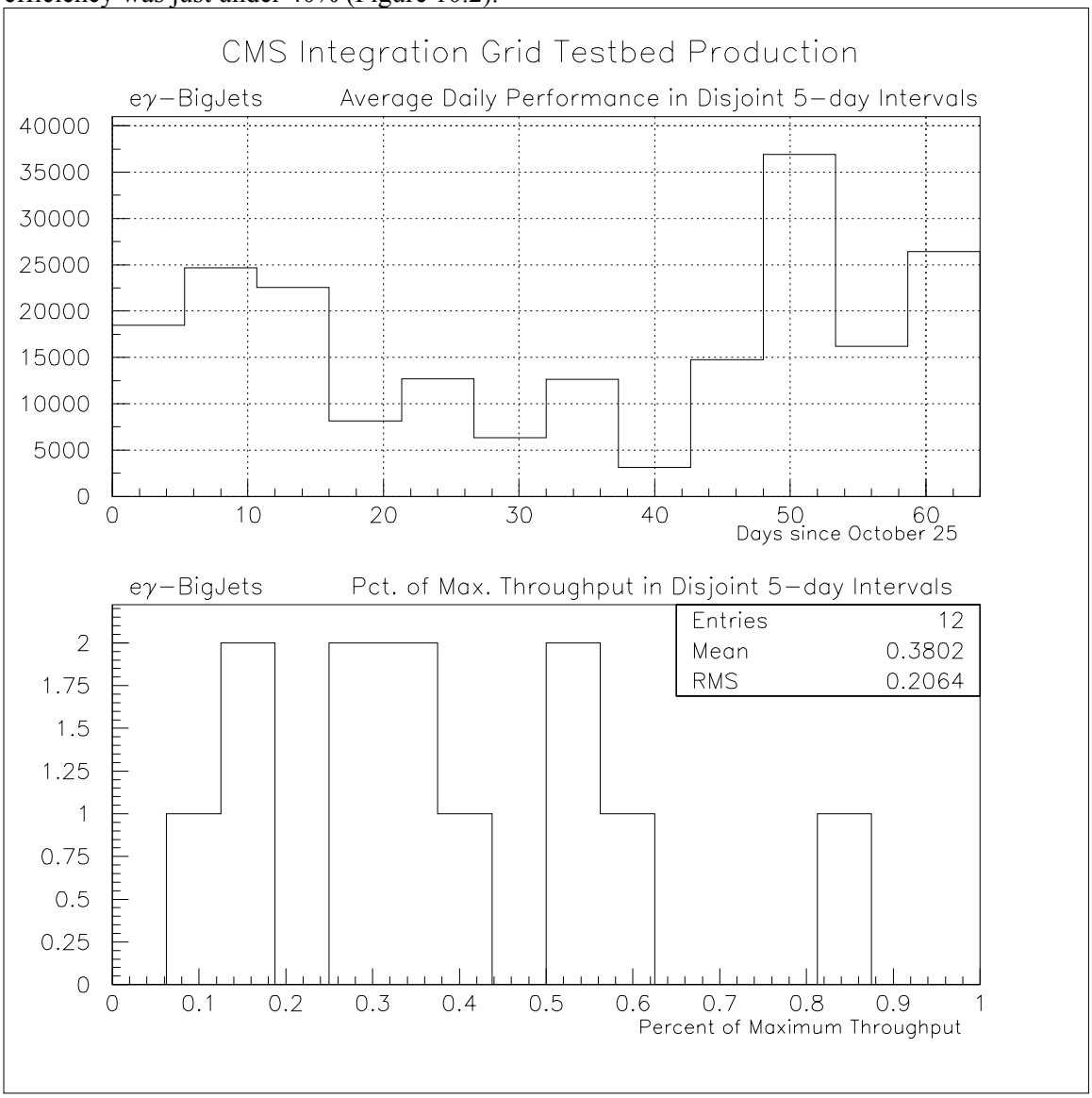


<FIG>Figure 10.1 The two left plots show the US-CMS Grid throughput as a function of time. The two right plots show the maximum throughput estimated by measuring performance on a single CPU at 750 MHz. Each completely simulated event required about 430 CPU-s. Adjusting for different CPUs, a theoretical maximum of 45,000 events/day for the US-CMS Grid was calculated.</FIG>

Despite the problems, the production was remarkably smooth and sustained for over two months. The two notable flat spots occur during the SC2002 conference and during the winter holidays, which reflect loss of manpower to submit new jobs during those periods.

In order to better quantify efficiency, the US-CMS Grid run period was divided into 12 periods of about five days each. The average daily production rate in each interval was compared

to the theoretical maximum daily rate of 45K events per day US-CMS Grid-wide. The average efficiency was just under 40% (Figure 10.2).



<FIG>Figure 10.2 Performance achieved on the US-CMS Grid, measured in average events per day, during each of 12 roughly five-day periods, shown as a function of time (above) and in histogram format (below).</FIG>

This performance is comparable to the conventional CMS spring 2002 production. The spring 2002 production was more complicated in that it involved more events with pileup and many more file transfers. Also, it is hard to calculate efficiency of the spring 2002 production because it is hard to determine when a site was unavailable due to problems or just idle for lack of a request. Nonetheless, similar estimates of efficiency range from 30 to 50%.

Scheduling functionality was not implemented in the MOP system during the fall 2002 US-CMS Grid run. Rather, jobs were distributed by direct operator specification at job submission time. MOP was logically divided into the MOP master site and the MOP worker sites. Jobs were created and submitted from the MOP master site, all input files were staged in from the MOP master site, and all output was returned to the MOP master site. No replica catalogs (Chapter 22) were used during the production process itself, but resulting data products were

registered in GDMP (612) at the end of processing. These issues are being studied in anticipation of a MOP upgrade. During fall 2002, Fermilab hosted the US-CMS Grid MOP master site.</TXT>

<H1>Acknowledgments</H1>

<ACK>We acknowledge the CMS Core Computing and Software group and the US-CMS Software and Computing projects for supporting this effort. We especially thank Veronique Lefebure and Tony Wildish of the CMS Production Team for their support and helpful discussions. Also thanks to the US-CMS Grid Team: Erik Aslakson, Julian Bunn, Saima Iqbal, Harvey Newman, Suresh Singh, and Conrad Steenberg of California Institute of Technology; M. Anzar Afaq, Shafqat Aziz, L. A. T. Bauerdick, Michael Ernst, Joseph Kaiser, Natalia Ratnikova, Hans Wenzel, and Yujun Wu of Fermi National Accelerator Laboratory; James Branson, Ian Fisk, and James Letts of University of California, San Diego; Adam Arbree, Paul Avery, Dimitri Bourilkov, Jorge Rodriguez, and Suchindra Kategari at University of Florida; Jaime Frey, Alain Roy, and Todd Tannenbaum at University of Wisconsin, Madison.</ACK>

<H1>Further Reading</H1>

<FR>For more information on the topics covered in this chapter, see http://www.mkp.com/grids.</FR>