PHENIX On-Line and Off-Line Computing

S.S. Adler [a] T. Chujo [a] E.J. Desmond [a] L. Ewell [a] T.K. Ghosh [k]
J.S. Haggerty [a] T. Ichihara [f,g] B.V. Jacak [h,g] S.C. Johnson [c,h]
H-J. Kehayias [a] J. Lauret [i] C.F. Maguire [k] M. Messer [a]
S. Mioduszewski [a,j] J.T. Mitchell [a] D.P. Morrison [a] I.D. Ojha [k]
C.H. Pinkenburg [a] M. Pollack [h,j] K. Pope [j] M.L. Purschke [a]
S. Sorensen [e,j] I. Sourikova [a] T.L. Thomas [d] M. Velkovsky [h]
Y. Watanabe [f,g] C. Witzig [a] S. Yokkaichi [f] W.A. Zajc [b]

(The PHENIX Collaboration)

[a] *Brookhaven National Laboratory, Upton, NY 11973, USA*

[b] *Columbia University, New York, NY 10027 and Nevis Laboratories, Irvington, NY 10533, USA*

[c] *Lawrence Livermore National Laboratory, Livermore, CA 94550, USA*

[d] *University of New Mexico, Albuquerque, NM, USA*

[e] *Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA*

[f] *RIKEN (The Institute of Physical and Chemical Research), Wako, Saitama 351-0198, JAPAN*

[g] *RIKEN BNL Research Center, Brookhaven National Laboratory, Upton, NY 11973-5000, USA*

[h] *Department of Physics and Astronomy, State University of New York at Stony Brook, Stony Brook, NY 11794, USA*

[i] *Chemistry Department, State University of New York at Stony Brook, Stony Brook, NY 11794, USA*

[j] *University of Tennessee, Knoxville, TN 37996, USA*

[k] *Vanderbilt University, Nashville, TN 37235, USA*

Data handling in PHENIX is carried out by the On-Line Computing System (ONCS) and Off-Line Computing System (Off-Line). ONCS provides the overall control and monitoring of the front-end electronics, trigger and data acquisition system and detector ancillary systems. It configures and initializes the on-line system, monitors and controls the data flow, coordinates calibration processes, interlocks the data acquisition process with the slow control subsystems and performs a number of other functions. ONCS uses CORBA software to monitor and control the hardware. Off-Line provides all aspects of data handling not directly connected to the collection of data and monitoring, such as event simulation and reconstruction, data analysis and information management. The impact of the unprecedented data volumes on the design is presented, along with a detailed discussion of the tasks and methods of simulating, obtaining and monitoring the data.

zajc@nevis.columbia.edu

1 Introduction

The PHENIX detector [1] is one of four detectors at the Relativistic Heavy Ion Collider (RHIC) that are designed to perform a broad study of A-A, p-A and p-p collisions to investigate nuclear matter under extreme conditions. The central arms of PHENIX measure electrons, photons and hadrons with excellent resolution using tracking detectors [2], ring-imaging Cherenkov and time-of flight detectors [3] and electromagnetic calorimeters [4]. A pair of forward spectrometers [5] provide excellent resolution for muon pairs. The characteristics of the collision are measured using zero degree calorimeters, beam-beam counters and a multiplicity vertex detector [6]. Events of interest for further study are selected using level-1 and level-2 triggers [7]. All of the above systems are described in other articles in this volume.

PHENIX records data at a rate of roughly 150 events/s for Au-Au running and roughly 1000 events/s for p-p running. One Au-Au event is typically 200 kbytes and a p-p event is 60 kbytes in size. Event data in PHENIX are digitized at the front-end and transferred via data fiber to VME-based Data Collection Modules (DCM) [7], where they are zero-suppressed, packaged, and shipped to the Event Builder (EvB)[7]. The EvB assembles a full event from the individual fragments of data from the DCM's. When the event is fully assembled and passes the level-2 trigger, it is temporarily stored on a local disk. A fraction of the events are made available to processes on a farm of PC's running Linux for on-line monitoring purposes. Long-term storage is provided by a HPSS-based tape robot system [8] operated by the RHIC Computing Facility (RCF) [9]. The average rate of transfer of data to HPSS is 20 Mbytes/s but for short time intervals rates as high as 60 Mbytes/s have been obtained.

Upon transfer to HPSS the data becomes the responsibility of PHENIX off-line computing. Off-line computing encompasses a broad range of activities including data archiving, information management and event reconstruction. It also includes creation and maintenance of both the detector simulation package and the overall framework to support these various endeavors. Simulation of the detector response is also necessary. While these tasks are common to the off-line computing environments of most large experiments, the nature of the PHENIX research program imposes some novel challenges, specifically the very high multiplicities of the Au-Au collisions and the large data volumes of approximately 1.5 Tbytes per day. In addition PHENIX must be able to accomodate the large event size mentioned above for central Au-Au collisions but also efficiently handle date at the approximately 500 kHz interaction rate expected for p-p collisions at design luminosity.

2 The PHENIX On-Line Computing System (ONCS)

The PHENIX On-line system[10] is a hierarchical, distributed system containing 12 varieties of custom front-end modules and 120 DCM's reporting to 26 Sub-Event Buffers (SEB). Event fragments in the SEB's are pulled through an ATM switch to a set of Assembly and Trigger Processors (ATP). This heterogeneous mixture of custom elements, crate controllers and commodity PC's running a variety of operating systems and communicating via several network protocols must be configured and controlled by the PHENIX Online Computing System (ONCS).

In order to function properly, a large number of individual components have to work together and need to be configured properly. These components, which are described in more detail elsewhere in this issue[7], are controlled by a number of individual computers and other processors. A schematic view of the DAQ front-end is shown in Fig. 1.

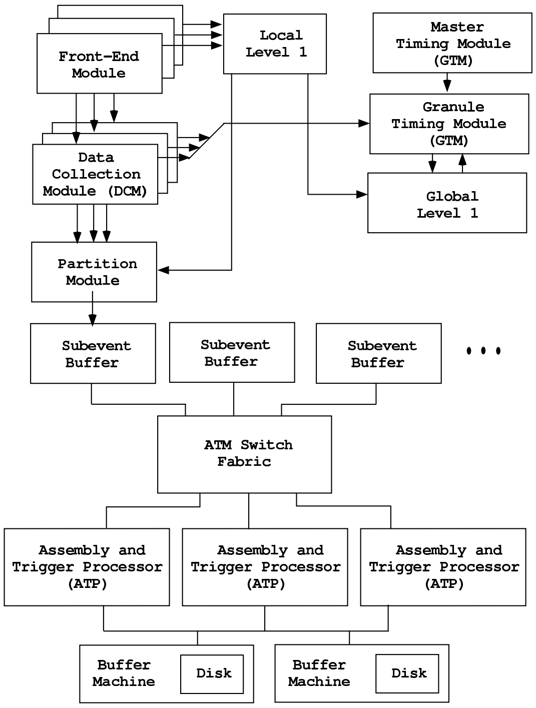

Fig. 1. Schematic view of the DAQ front-end.

2.1 Control of the System

PHENIX has developed a software package which controls the data taking function of the detector. This software package is called Run Control (RC) and provides a graphical user interface which allows one to operate one of the configured partitions of the DAQ system to enable it to collect data to a file. It also provides a server which places all the detector DAQ components in the appropriate state to start and stop the collection and storing of data onto disk. The development of this client-GUI front-end and server back-end architecture was necessary due to the distributed nature of the PHENIX DAQ system

The 5 main components of the DAQ system are the timing system, the front end modules (FEM), the DCM's, the EvB and the data logger. RC orchestrates the control commands to each of these systems in order for data to flow properly. This process begins when an accept signal from the level 1 (LVL1) trigger is sent to the front-end electronics (FEE) through the timing system. Next RC tells the FEM's to send data up to the DCM's, then through the EvB's and onto the data loggers where the data is recorded to disk files. Flexibility has been designed into the system such that one can run the full detector with all granules configured into one partition or at the other extreme can be run as a single granule configured into a single partition.

The underlying communication system used by the RC system is the Common Object Request Broker Architecture (CORBA) [11] which controls the many components that must work together to successfully accumulate data. CORBA, an industry standard, provides the ability to transparently access objects on remote computers of heterogenous types throughout the network. A CORBA server is set up on each one of the detector control systems. For example, the timing system and DCM's are operated by a VME crate controller which has a CORBA server running under VxWorks. The EvB is made up of over 60 Intel based PC's running Windows NT. In each of these PC's, a CORBA server is running. Additional components that need to be configured are the Local LVL1 trigger and lookup tables, the Global LVL1 trigger and the LVL2 trigger. The main job of the RC server is to reference the objects served by the CORBA servers and execute commands which set the DAQ components into the appropriate data taking mode.

The overall configuration of the DAQ system is read from the Global LVL1 trigger system by RC when it is first executed. RC then collects all the references to the timing system, DCM boards, EvB nodes and other elements which are operated through CORBA objects. When a user request is downloaded to initialize a certain partition, RC then goes through a list of objects in the correct order and executes the initializes the function for each object which in turn executes the initialization software for each component. A similar operation is done when the user requests the detector to start or stop collecting data.

2.2 Data Transfer

The data acquisition system can acquire data at a peak data rate of about 60 Mbytes/s. The duty factor of the RHIC accelerator and the PHENIX experiment combined is expected to be of the order of 30%. This duty factor, combined with the instantaneous data rate and substantial local buffering capability , is well-matched to the average bandwidth of 20 Mbytes/s allocated to PHENIX for storage into HPSS. This data rate is driven by available mass storage technology, and by the amount of data that can be handled and analyzed by RCF. With an event size of about 200 kbytes for Au-Au minimum bias collisions, this translates into a sustained rate of roughly 100 full events per second to tape.

The data are funneled through the "Event Transfer" (ET)[12] system, which was developed at Jefferson Laboratory and adapted for use in PHENIX. The ET system allows one to receive data from an arbitrary number of sources into an event pool, and to distribute the event data to an arbitrary number of local and remote client processes. The data logger is one special example of such a client. It has special privileges which guarantee the delivery of *all* events. The data logger is the only client allowed to throttle the data flow by raising the data acquisition busy signal if it cannot keep up with the data rate. All other clients receive events on a best-effort basis only, thus they must have the capability to "skip" events. These clients are typically on-line monitoring processes which analyze the data and verify the proper state and proper functioning of the detectors.

The data from the EvB are temporarily stored on two large disk arrays of about 850 Gbytes, each of which holds a maximum of 12 hours of data taken at the design rate. By buffering the incoming data on the local disks, the data can be transferred to the HPSS system at a steady rate and excess data can be transferred during down times of the accelerator or the data acquisition. In addition, the buffering allows one to continue data taking for some time in case of a network or HPSS system outage. A given data set will stay on the disk for several hours before making room for new data, which makes it possible to access the data for various monitoring purposes.

The two disk arrays are attached to two powerful dual-CPU computers running the Linux operating system. The machines are equipped with Gigabit network interfaces which allow raw data accumulation rates of 1000 Mbit/s. One of the systems will receive data from the EvB, while the other one is sending data to HPSS. When the receiving disk array reaches a fill status of about 70%, and the files of the sending systems have been transfered, those files are deleted and the systems switch their roles. In this way, the disks on one system are being written only, while the disks on the other system are being read only, which greatly increases the throughput of the disk systems and the respective PCI and SCSI data busses. We have achieved long-term sustained data rates of 40 Mbytes/s into the HPSS system, which leaves sufficient contingency to make up for lost transfer time, for example in the event of a network outage.

2.3 Online Monitoring and Calibrations

During data taking, the event data are continously monitored to ensure the quality of the data. In addition to a number of desktop systems, the PHENIX experiment has a cluster of 32 dual-CPU computers using Linux, which run the monitoring and calibration algorithms. On-line monitoring processes typically use data from the ET pool, with a latency of a few seconds after the data have been recorded. It is possible to distribute, in addition to the data logging, about 20% of the events in this way without an impact on the total data rate through the system. For this kind of monitoring, priority is given to so-called *calibration events*. These are special events reading data generated by various calibration signals, which is detector specific. For example, detectors with a photomultiplier readout system have a laser system which distributes short light pulses of known intensity to the detector cells. The data from those special events allow one to monitor the photomultiplier gain, and correct gain changes that exceed predetermined thresholds. Other systems with charge-sensitive ADC's allow one to inject a charge into the ADC and thus monitor the ADC gain. Those events, which typically produce a signal in *all* cells of the detector, also allow one to recognize nonfunctional areas, for example in the event of a high voltage failure. Some data from actual collisions is also distributed through the ET system, but the main use is the distribution of these calibration events.

When reading from the ET pool, clients request data according to an *event profile*. At the time when an event is added to the pool, it is supplemented with an event profile, which provides a high-level summary of its characteristics. In this way, a monitoring process can request only those types of events which are of interest (for example, the abovementioned laser events) and will not be burdened with other events it would otherwise need to discard.

The calibration events are used to monitor gain variations over time and alert the shift crew to a problem as quickly as possible. In order to establish the calibration constants, and to determine the efficiency and verify the proper working of the Level-2 trigger, it is necessary to analyze a large amount of data from actual collisions. This is done by reading the files which are being transferred to the HPSS system, which is possible because the files stay on the disks for several hours. The throughput permits data transfer at a rate of 20 Mbytes/s, while providing simultaneous read access for the various analysis processes at an aggregate rate of roughly 40 Mbytes/s. The goal is to have a preliminary set of physics calibration constants with an accuracy of better than 10% for a given dataset by the time the data are stored in the HPSS system.

3 PHENIX Off-line Computing

PHENIX off-line computing encompasses a broad range of activites including archiving to and retrieving data from mass storage, event reconstruction and simulation, support for physics analyses and the information management underlying all these efforts. PHENIX faces many novel challenges in addressing these tasks.

One challenge stems merely from the volume of data which will need to be handled. In a nominal year PHENIX is expected to record to tape in excess of 200 terabytes (Tbytes) of raw data, 100 Tbytes of reconstructed data, 100 Tbytes of physics analysis data and tens of Tbytes of simulation data. This is one to two orders of magnitude more data than that collected by previous generations of nuclear physics experiments. Many of the techniques for working with data which are familiar to researchers with a nuclear physics background simply do not scale to data volumes of this size.

The computing demands of PHENIX off-line also pose a challenge. Event reconstruction, physics analysis and simulation all require computing power on a scale which is very large when compared to the needs of previous generations of nuclear or high energy physics experiments. The average reconstruction time for a minimum bias sample of $\sqrt{s} = 200$ GeV Au+Au events is about 300 SPECINT95-seconds per event (a typical CPU today rates around 30 SPECINT95). This problem has a fairly conventional solution in the form of loosely coupled compute farms. Event reconstruction is an example of a "trivially parallel" problem and is well suited to this sort of computing architecture. Raw data files are kept to less than 2 Gbytes in size, they contain nearly 10,000 events and take nearly 12 hours to reconstruct. Each of the farm nodes has a 100 Mbit/sec network interface and the transfer rates of data to and from the reconstruction farm nodes have been measured at approximately 6 Mbytes/sec. The total data transfer time of a reconstruction job is a small fraction of the total reconstruction time so that the reconstruction time per event would have to fall by a huge factor before a more tightly coupled computing architecture would be advantageous.

The compute farm is divided into a reconstruction farm and an analysis farm, each of which is roughly comparable in size. All of the nodes are identical which allows us to reassign nodes from one part of the farm to the other as needed. The Load Sharing Facility (LSF) [13] system was adopted to manage the batch system which runs the analysis portion of the farm. We have defined a single LSF queue for jobs on the analysis farm A Control Reconstruction Server (CRS) was developed inhouse to manage the reconstruction farm. This was necessary since the compute farm consists of hundreds of nodes and the expense of purchasing LSF licenses for the full farm would be substantial.

Some of the machines of the analysis farm allow interactive logins while some are only accessible via a batch system. The interactive machines were assigned to different analysis and detector groups in PHENIX. On their respective machines, each group is allowed to decide how their machine is to be used. They are not guaranteed exclusive use of the machine since the batch system continues to run jobs at low priority on all analysis machines, but they are allowed to use the local storage and run jobs interactively on the node as they see fit.

The development of off-line software is organized around packages, each with a named person as a primary contact. CVS (concurrent version system)[14] is used to maintain and track software revisions. The main software repository is kept in AFS, and individual CVS clients act upon that repository as though it were a local filesystem. The use of an AFS resident software repository predates the wide use of CVS servers. If the repository were being set up today, the latter approach would be the preferred one. As appropriate, we use CVS to define branches in the development line, which allows us to segregate bug fixes on a development branch headed for production use from more vigorous development taking place on the main trunk of the development tree.

PHENIX reconstruction, analysis and simulation code consists of about 500,000 lines of code. A code base of that scale requires an extensive set of tools to keep it functioning. We rely heavily on regular rebuilds of the code to maintain its quality. Usually a rebuild of all the code is performed once per day. These rebuilds start completely from scratch, checking out code directly from the CVS repository and compiling everything that constitutes the PHENIX off-line computing environment. There are often a variety of rebuilds underway at any time. Examples are building code for different platforms or with the code instrumented checking for memory leaks. We have found that when the code is working properly, or is very nearly working properly, the amount of effort needed to keep it that way is manageable. If the code is allowed to diverge far from a working state, the effort needed to restore it can be enormous. The daily rebuilds really help keep the code near to a state in which it works as it should.

4 Simulation

Simulation of the PHENIX detector is carried out using the simulation package called PISA (PHENIX Integrated Simulation Application). PISA is implemented in GEANT 3.21. The package has been organized in a highly modular fashion such that individual detector subsystems could be developed and tested in a stand-alone mode while still preserving the goal of a highly integrated simulation analysis.

When the detector subsystem designs reached maturity and went into actual construction, the simulation goals shifted to furnishing large, realistic simulated event sets to assist the event reconstruction software developers in unraveling potential physics signatures from the future real data. Three formal Mock Data Challenges (MDC) were undertaken before the actual start of real data acquisition in June 2000. Cumulatively, these MDCs generated close to 1 Tbyte of simulated data files, as compared with about 5 Tbytes of real data generated in the first run of RHIC. These MDC exercises, which were also conducted by the other RHIC experiments, were crucial to the verification of the hardware and software infrastructure of the RHIC Computing Facility. The progressively larger scale MDCs were also valuable in developing the simulation event generation capabilites of the PHENIX collaboration, both on and off-site, such that we are able to generate hundreds of thousands of minimum bias events per week for comparison with the real data.

The successful implementation of Level-2 software triggers for run-2 data taking relied heavily on the availability of simulated data for the development and testing of the various algorithms. The Level-2 software development uses simulated data files as part of the process to allocate the trigger bandwidth. This is confirmed by comparisons with analysis of real data. Similarly, in the outlying years there are plans to introduce whole new detector subsystems into PHENIX. The code remains non-platform-specific and will be used to optimize by simulation analysis these new systems.

Simulations using PISA is supported on a multitude of off-site operations mostly using Linux systems. PISA simulations are also carried out off-site using a Compaq/Alpha farm and on-site at the SunOS complex at the RHIC Computing Facility. The simulation software is the last major dependency on the CERN libraries in PHENIX computing. That depencency does not extend to the output of the PISA component since we have recently incorporated ROOT[15] software for our output system. Individual subsystem tracking information formerly written to ZEBRA files is now placed into detector specific C++ classes which are then written into a standard ROOT Tree file. In the future, when GEANT3 is replaced by GEANT4, these classes can be migrated to the new simulation system without impacting the existing reconstruction software for simulated events.

5 Reconstruction and Analysis

The reconstruction code is described in detail elsewhere[16]. The general approach to reconstruction was to organize it as a set of modules, each of which shares a common interface. Each module has an "event" method that takes a reference to a node in an hierarchical tree of data. Each module navigates the node tree, finds the data structures it needs and creates new nodes to hold the output of the module.

We use ROOT, an object-oriented analysis framework developed at CERN, to provide the I/O interface to our reconstruction and analysis framework. ROOT is also used as a late-stage physics analysis tool for filling and visualizing histrograms. ROOT is able to handle a wide variety of built-in and user-defined objects, and it provides a C++ interface for manipulating those objects. C++ is a strongly typed language, and this is somewhat at odds with the need to handle objects generically. In a typical application using ROOT, this results in a dangerous reliance on type-casting in user code. We wanted to reduce, or at least centralize, the need for this in PHENIX off-line code. This is accomplished with an interface layer which uses C++ templates to generate specialized code for handling user-defined types. The result is a general, but still type-safe, interface to physics data.

5.1 Databases

A pragmatic approach was taken with respect to the use of databases. Not all data are stored in a database, in particular, event data are stored only as sequential records in HPSS. The data we do store in a database, and that are considered archival in nature, are stored using Objectivity, an object-oriented database management system[17]. Interfaces have been developed on top of raw calls to Objectivity itself for some applications within PHENIX. The interface seen most commonly throughout PHENIX off-line code is that of the calibration database. Objects are stored with a validity period as a key, with each object representing a particular type of experimental calibration.

There are other databases besides Objectivity in PHENIX. Simple relational databases, such as MySQL [18], are used within the data carousel file retrieval software (see next section) and the on-line run database, for instance. We have found that the idea of a complete integration of all experimental data within one overarching database system is too ambitious to be worth the manpower required to develop and maintain it.

5.2 Resource Management

At present, storage costs are such that only a small fraction of PHENIX data can be kept on disk at any time. The RCF uses the commercial system HPSS [8] to manage the collection of tape silos, disk caches and computers that handle the near-line storage of files. Interaction with this near-line storage plays a significant role in PHENIX computing, and while that interaction has been engineered to be as straightforward as reasonably achievable for users, it has been important to keep it very efficient in order to maximize our use of limited available bandwidth. The relatively long latencies inherent in retrieving data from tape provide a strong incentive to aggregate requests for files and read multiple files from a tape cartridge once that cartridge has been mounted in a drive. A key component in achieving this in PHENIX has been the development of the *data carousel*, a batch oriented file staging system that sits atop HPSS.

One component of the carousel is a HPSS batch system written by IBM that given a list of file names and retrieves files from tape in an order that tries to minimize the number of tape mounts needed. The batch system routinely achieves a factor of three to five better throughput than does retrieval of files without its help. This batch system is fed by a separate server that implements PHENIX policies for resource allocation. To retrieve a set of files, a physicist sends the server a list of file names, while at the same time identifying the group within PHENIX for which this request is being made. The entries are stored in a MySQL database. The server starts periodically, examines the database, determines the next set of files to send to the IBM batch system, sends them and exits.

Once a file is staged to HPSS cache, a call-back is issued via ssh to the machine specified in the initial staging request. The call-back invokes a client side script which checks for available space and pulls the file from HPSS using a specialized version of ftp. Client side policy can be implemented in this call-back script. The default behavior is a LRU cache, but this can be adapted by each user as appropriate.

All of the information needed to define the state of the server is kept in the MySQL database. Because of this, the carousel server is able to run as a cron job, not as a daemon. This relieves us of the need to write a program capable of running for extended periods of time without leaking resources. This has helped make the system exceedingly stable. Having informtation in the database also simplifies the development of web interfaces that can be used to see which files have been staged to disk, which ones will happen soon and which ones have errors being staged.

We also use a file catalog, implemented as a set of Objectivity databases, in order to keep track of the locations of files. Each significant file that is generated in PHENIX is given a unique name and that name serves as a key into the file catalog. Each file may have one or more replicas (physical instantiations of the file) and a list of URIs (universal resource indicators) locating the replicas it stored as part of the file catalog. The uniqueness of each file name is important for this scheme to work properly. Until now, only limited use has being made of the file catalog as we have had sufficient disk space to keep a single copy of nearly all reconstructed output on disk. The data set for future runs is expected to dwarf any realistic hope of keeping all, or even a large fraction, of the data on disk and one would then expect the file catalog to play an increasingly indispensible role in daily data analysis activities.

5.3 Software Quality

A significant challenge facing PHENIX off-line software is that of software quality. The correctness of the physics results hinges on the quality of the code. Our ability to get any results at all requires code of sufficient quality to run without crashing while reconstructing many Gbytes of data. Our reconstruction code in particular has to run for such a long period of time to process a file that its quality requirements have much in common with those of system-level daemons like NFS servers.

Many people, with a wide variety of skills, and many of whom are no longer with the collaboration, have contributed to the PHENIX code base. This presents a substantial maintenance challenge. We have employed several tools to try and characterize our code, in order to focus the limited manpower available for improvement on the most critical pieces of code. We use the statistical profiler jprof [19] to determine in which pieces of code consume the most computing cycles. It is not at all unusual to produce a several-fold speedup in a routine once a physicist with more programming experience takes a look at it. In some extreme cases, several hundred-fold speedups in specific routines have been achieved.

In addition to performance improvements, we have had to devote a fair amount of manpower to controlling memory growth of the reconstruction code. While processing a full file of raw Au-Au data our reconstruction program will loop over thousands of events and hundreds of thousands of tracks. Our code starts out with a virtual set size (VSS) of about 300 Mbytes; our reconstruction nodes have 1 Gbyte of RAM and run two jobs simultaneously. If the VSS grows much faster than about 20 kbyte per event the program slows significantly before it has a chance to finish, as the system settles into heavy swapping. Some of this memory growth is of course due to actual memory leaks and some of it is just the growth of dynamically resizeable structures in response to seeing a larger sample of data. The "knee" in the multiplicity or spectrum in Au-Au occurs at about 0.5% of the total cross section, so one would expect that the high-water mark of memory usage (assuming rough proportionality between memory usage and event multiplicity) to grow randomly, and often very rapidly, with the number of events for the better part of the first few hundred events of a minimum bias event sample. Our raw data files contain 5000-10000 events but after one or two thousand events the memory growth of a reconstruction job due to sampling the event size distribution is not significant. Finding true memory leaks is difficult; but through a combination of informal code reviews and the use of the commercial leak checker Insure [20], we have reduced our asymptotic memory growth to about 15 kbytes per event, a level which is tolerable if not ideal. .

Another aspect of code quality that has received a fair amount of attention in PHENIX is that of repeatability. In principle, reconstructing a file of raw data twice should yield the same results. In principle, it shouldn't matter whether one runs the reconstruction code under Linux or Solaris. In practice, these things matter, until effort is expended to track down the reason for the differences and deal with them. Usually this detective work uncovers an uninitialized variable or an instance of reading past the end of an array. We have written sofware modules which print out selected results as simple text files. Variations in the output between different runs of the code are used to hunt for the software cause. This procedure is being incorporated into our nightly software rebuilds and tests to help us spot new problems as they are introduced.

5.4 Off-Site Facilities

Because of the widely distributed nature of the collaboration and the enormous computing requirements of PHENIX, off-site computing facilities have an important place in the computing plan. Primary reconstruction of the experimental data takes place at BNL, but off-site facilities are the primary providers of computing for simulations and the subsequent reconstruction of those simulated events, and they are important centers for regional analysis. This is especially important as the bandwidth and latency of current wide-area networks make working at BNL from afar painful, if not outright impractical, for interactive analysis. At this stage the flow of data to and from off-site facilities is deliberately simplistic. BNL maintains the master copy of the Objectivity database and off-site facilities either connect directly to the database at BNL or make periodic slave copies for local installation. All data eventually flow to BNL from where is can be redistributed. A major addition to the computing facilities available resulted from the installation of the PHENIX Computing Center-Japan (CC-J) [21]. This facility has a storage space of 400 Tbytes and resulted in a large increase in PHENIX computational capabilities.

PHENIX off-line computing is a pragmatic hybrid of new technologies needed to handle the enormously large data sets that will be collected and concessions to real-world constraints. It has proved a reasonably successful strategy for the distributed development of a complex system of software.

6 Acknowledgements

We acknowledge support from the Department of Energy (U.S.A.)

References

[1] Article entitled “PHENIX Detector Overview” this volume.

[2] Article entitled “PHENIX Central Arm Tracking Detectors” this volume.

[3] Article entitled “PHENIX Central Arm Particle I.D. Detectors” this volume.

[4] Article entitled “PHENIX Calorimeter” this volume.

[5] Article entitled “PHENIX Muon Arms” this volume.

[6] Article entitled “PHENIX Inner Detectors” this volume.

[7] Article entitled “PHENIX On-Line Systems” this volume.

[8] http://www.sdsc.edu/hpss

[9] RCF paper this volume.

[10] M.L. Purschke et al., IEEE Trans. on Nucl. Sci. 47, (2000) 51.

[11] E. Desmond et al., Proc. of the Real-Time 99 Conf., Santa Fe, NM, (1999).

[12] C. Timmer et al., CHEP2000, Padova, February 2000, http://chep2000.pd.infn.it/short_p/spa_e058.pdf.

[13] http://www.platform.com.

[14] http://www.cvshome.org/

[15] R. Brun and F. Rademakers, Nucl. Instr. & Meth. in Phys. Res. A389, (1997) 81.

[16] J.T. Mitchell et al., Nucl. Instr. & Meth. in Phys. Res. A482, (2002) 491.

[17] Objectivity, Inc., http://www.objectivity.com.

[18] M. Maslakowski and T. Butcher "SAMS Teach Yourself MySQL", Macmillian Co., Indianapolis, IN 46290, U.S.A. (2000).

[19] http://lxr.mozilla.org/mozilla/source/tools /jprof/README.html.

[20] http://www.parasoft.com.

[21] http://ccjsun.riken.go.jp.