BaBar Simulation Production—A Millennium of Work in Under a Year

D. A. Smith, F. Blanc, C. Bozzi, and A. Khan, *Member, IEEE*

***Abstract*—The BaBar experiment requires simulated events beyond the ability of a single computing site to provide. This paper describes the evolution of simulation and job management methods to meet the physics community requirements and how production became distributed to use resources beyond any one computing center. The evolution of BaBar simulation along with the development of the distribution of the computing effort is described.**

As the computing effort is distributed to more sites there is a need to simplify production so the effort does not multiply with number of production centers. Tools are created to be flexible in handling errors and failures that happen in the system and respond accordingly, this reduces failure rates and production effort.

This paper will focus on one cycle of simulation production within BaBar as a description of a large scale computing effort which was fully performed, and provided new simulation data to the users on time.

***Index Terms*—BaBar, distributed computing, Linux, production management, simulation, Solaris.**

I. BaBar Simulation History

THE BaBar High Energy Physics (HEP) detector [1] is based at the Stanford Linear Accelerator Center (SLAC), Menlo Park, CA USA. The experiment investigates the subtle differences between matter and anti-matter by continuously colliding bunches of high-energy electrons and positrons 250 million times per second and searching for the creation of rare B-meson and anti-B-meson particles. In early 2003, BaBar was into its third run cycle of data taking (run 3). The experiment already had nearly 80 fb^{-1} (inverse femtobarn, a gauge of the sensitivity of the data to measure rare interactions) of data, and by the end of run 3 BaBar would have 110 fb^{-1} available for analysis. The physics community had requested a certain amount of simulated events to compare to this amount of data. The requests were: three times the data set for generic signal events; matching the data for generic background events; and various specific signal events as requested. These three requests were roughly similar in computing effort.

The total request translates to a number of events to be produced in simulation. In BaBar the simulation and reconstruction code is tagged in major software releases, and each major release is used in a cycle of production which roughly matches the data cycles. These cycles of simulation production are numbered, and this paper will mention three cycles in detail, SP4, SP5 and SP6. In 2002, SP4 had the purpose of producing simulation for data run cycles 1 and 2, and to match the physics request would require 1.2 billion events. SP5 in 2003 would produce events for run cycles 1–3, and need 1.6 billion events. For SP6 it was recognized that the new reconstruction code would not produce significantly different events than what was produced in SP5, so SP6 would only produce events for run cycle 4, and SP5 could be used for analysis of run cycles 1–3. This change resulted in SP6 only needing 1 billion events to match the request.

This resulted in the fact that SP5 would be the largest requested production cycle in BaBar, and would need a greater amount of distribution of the computing effort to get done on time. This effort was performed and finished earlier this year, and this paper will concentrate on this effort as a description of a complete large scale computing effort.

Manuscript received October 4, 2005.

D. A. Smith is with the Stanford Linear Accelerator Center, Menlo Park, CA 94025 USA (e-mail: douglas@slac.stanford.edu).

F. Blanc is with the Department of Physics, University of Colorado, Boulder, CO 80302 USA (e-mail: fblanc@pizero.colorado.edu).

C. Bozzi is with the Department of Physics, INFN Ferrara, Universita di Ferrara, I-44100 Ferrara, Italy (e-mail: bozzi@fe.infn.it).

A. Khan is with the School of Engineering and Design, Brunel University, Uxbridge, UB8 3PH Middlesex, U.K. (e-mail: akram@slac.stanford.edu).

Digital Object Identifier 10.1109/TNS.2006.873080

II. Computing Resources Needed

Assuming a fictional 1 GHz Pentium III machine, we can look at the resources needed at an event level. There is a range of the computing time needed to produce an event in production, depending on the type of decay mode to be simulated. The range is 3 to 10 sec to fully simulate and reconstruct each event, and the amount of data produced in SP5 was 30 to 45 kB per event. When averaging over decay modes the time per event is 8 seconds, and the data produced per event is 40 kB.

Looking at the resources needed to produce the requested events, it is important to remember that the requests are the starting point and not the full story. We designed the system to use at least get 80% of the given cpu, this will increase the amount of resource needed. Also, users might require more than first requested, and large amounts of production will have to be re-done for various reasons. These above reasons will increase the amount of needed resources beyond what was first requested to get the simulated events in time to users.

The resources needed to complete the requests can be determined by using the above figures as follows. In SP5 the request was for 1.6 billion events. Multiplying this by the averages, you get 420 years on the fictional machine, and 61.5 TB of data produced. Since large blocks of the production needed to be re-created and people requested more as the production continued, the actual number of events was 2.2 billion. Putting in the 80% of cpu use with this larger number of events the computing time comes to 700 years and the data produced is 84.5 TB. This sets the scale of the computing effort for the SP5 production cycle.

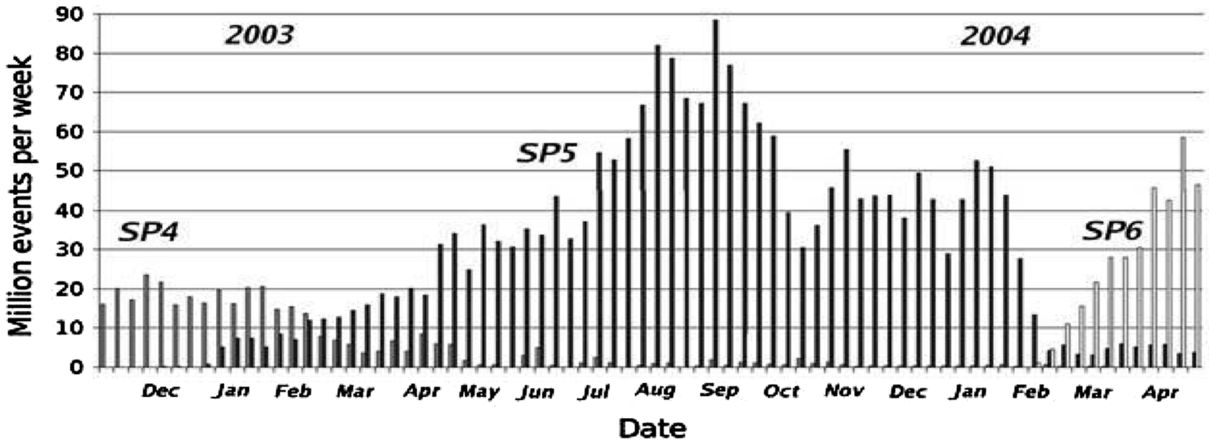

Fig. 1. Simulation production in the BaBar experiment by week, for the time period covering the SP5 cycle of production. The figure displays the different cycles of production in BaBar, and how they overlap in time. The details of the simulation cycles are described in the text.

The production of these events is divided up into computing jobs, where each job will produce 1000 to 2000 events, and the final 2.2 billion events were created with over 1 million computing jobs. On the fictional 1 GHz Pentium III machine these jobs would take on average 2.3 or 4.5 hours to complete.

The code to run these jobs was developed in BaBar on Solaris and Linux systems. In the SP4 production cycle, some simulation was produced using Solaris, but by 2002, when SP5 was starting, all new cpus purchased were commodity Intel machines running Linux, and all of the production in SP5 would be on Linux machines, this continues to be true in SP6.

In summary the SP5 computing effort was the management of over 1 million jobs on Linux machines, this continues to be true in SP6, each taking 2 to 5 hours to run.

III. An Overview of Simulation Production

Fig. 1 shows a plot covering most of the time period of the SP5 production that started in January 2003, including some of the time period before and after to show the overlap with the end of the previous SP4 production, and the start of the SP6 production. This plot displays the production in terms of millions of events per week for each week in this time period.

This plot illustrates a year and a half of the history of BaBar simulation production, and how simulation cycles SP4, SP5, and SP6 overlap in time. Further comments on the styles of each production is needed, since each cycle of production was not just a new release of BaBar software used to simulate events, but was actually a complete re-working of how production was to be done, a revolution in production style for each cycle.

In SP4, production was split into three jobs—a simulation stage with particle generation and Geant4 [2] simulation; a mixing stage to produce detector signals including measured background events; and a reconstruction stage to produce events to be used in analysis. The use of three stages of production, each having a separate job, was a harder management problem to solve, since the 1.2 billion events were produced in over 1.8 million jobs. Also there were three times more failures to track, where each failure would affect the management of the next job (i.e., simulation failures effected submission of mixing jobs).

In SP5 the three-stage production was replaced with a new simulation executable, which would perform all three stages (simulation, mixing, and reconstruction) on each event, before producing output. This had a huge affect on the management and production of simulation. There were now one third as many jobs to manage. There was less server load since there was no output from each stage (for technical reasons it was 8 times less server load). Each job was now longer, so the batch queues could be used more efficiently. This resulted in a greater efficiency in production in SP5 in comparison to SP4, since less work was involved to produce the same amount of data. There was a trade off, since the new executable required 512 MB of memory to run (previously it was only 256 MB of memory needed), and there would be some computing overhead so each event would take 10% longer to produce. At the time, memory was becoming cheaper and most of the batch farms already had at least this amount, so it turned out to not be a serious restriction to production.

The production cycle SP6 was another revolution in method since it included BaBar's computing model 2 changes [3], [4]. This new computing model included a number of base changes in BaBar computing, but the one to most affect production was the change from an event store based on databases, to an event store based on files. This change meant control code would now have to manage the production and distribution of the produced files. Although this increased the complexity of the control scripts compared to database use (where the database system would control the files produced), the added control over production freed up how production could be done, and again increased efficiency. Production output was now stored into a file structure, and removed the overhead of maintaining a database. This drastically reduced server load again, and reduced the chance for job failures, making production much easier for production managers. The trade off was increasing the complexity of the control code, but this was something under our control, done once and perfected in testing, so it was not a concern for production itself.

IV. Production Methodology

Since the SP5 data would be needed by the beginning of 2004, and we could not wait for one machine to do the production in 700 years, we would need to run the jobs in parallel on thousands of cpus. At the end of 2002, SLAC had over one thousand cpus but these were needed for other efforts, such as data reconstruction and analysis. Before this point there was a stated desire that the computing efforts within BaBar should become distributed to more of the institutions in the collaboration. Simulation production was sited as a good candidate for distribution. Lack of required resources at SLAC would not be a problem, since computing resources would be found at any BaBar institution that could provide them.

When increasing the number of sites, one must be careful not to also multiply the effort to the collaboration. The initial production at SLAC was done with 3 people working in shifts. The BaBar collaboration could not withstand 3 people per production site. The standard which was agreed upon was that each site could only require one half-time person in the collaboration to get the work done.

To get a large amount of production done, without increasing the total effort required good tools to be created. A set of tools providing a number of services were created called "ProdTools", these were command line tools and libraries to help the production managers with their daily tasks. Also they would provide control for the jobs, so the task could get done efficiently with the provided resources at each site. In the case of SP5, the tools provided specific requirements to be able to use the database event store in a production environment (such as only one job could start in the database per minute).

ProdTools provides an interface between the central production database at SLAC, and the local batch systems at each site (see Ref [5]). The system was developed around one single database at SLAC, which would provide the global coordination for simulation requests, runs, and jobs. All sites would attach to this database to determine configuration information for the jobs to be submitted.

But the main point of the tools was not the submitting of jobs, but recognising when the jobs fail, and how to fix these failures. This requires recognizing different failure modes, and determining what is the proper response for each. As these modes were recognized, the recovery of each failure could then be coded, and improvements on the recovery procedures were possible.

In any production failures always happen, no matter how stable the computing systems can be made. In SP5 the best we were able to do was a failure rate of 4–6%. The standard was that the tools should be developed until they are able to respond correctly for all but 1 in 10 000 jobs, including failures. Also a failure of a single job should not hang all production. In SP5 there was over 1 million jobs, about 50 000 of these will fail. The tools should be able to then fix all but 100 of the total jobs, so the effort does not increase with failures. In SP5 to further reduce effort, once this level of production was reached, we would then just abandon the remaining failed jobs and accept that they would not get done. Along with ProdTools to manage the jobs, there was a tool developed to manage the transfer of the produced databases, which was called "MocaEspresso". This tool would recognise the closed databases produced in SP5 and package them for transfer to SLAC. All data produced in BaBar has to be transferred back to SLAC for archiving before it can then be distributed to other sites for analysis. The total data produced in SP5 was 80 TB over the course of a year, and this meant an average of 200 GB a day would be coming into SLAC, and there was a maximum in production of 500 GB per day over the course of the year. This required the use of a set of file servers dedicated for transfers, and local tools were developed to handle the file archiving and attaching them to the production databases. These tools also had to be careful and error correcting to keep up with the required transfer rates—if they could not handle the 200 GB on any given day, they would need to be able to handle 400 GB the next day.

V. Remote Site Resources

Most of the management and control of the jobs was handled by one system at SLAC, but production was done at remote sites, and there were certain requirements on the resources needed to be able to run BaBar jobs. For SP5 there was the database to setup and hold the produced events until these databases could be closed for export. This database setup also needed to include the BaBar conditions database, and the background events to be used for the mixing stage. This produced the requirement of a file server with about 500 GB of space.

To run the jobs each site would need as many cpus as they could get, with a limit of about 120 possible jobs per file senrver. Each of these cpus would be put into a batch system for job submission, and they each had to read and write to the file server over the network, requiring a network switch that could handle the load. There was a requirement for one control machine per site, which would connect to SLAC and to the global database. Any of these machines could be shared with other services, but these types of machines at least had to exist.

The sites setups were very diverse within these requirements. Many of the academic sites had obtained funding to be a dedicated production site, and they were setup with a fairly basic 32 dual processor machines and a file server with attached disk array. But the other larger sites would often already be setup in some manner which we could not change. Batch queues would have to share resources with other efforts, with numbers of cpu used for production that would vary day to day. File servers would be shared with other efforts, making server load an important concern. Simulation jobs could be background processes on other production efforts. Also the batch machines could have variable amount of memory and local disk, including one site with batch nodes that had no local disk at all.

With the variability of resources, there was also a variable software infrastructure, with sites using either nfs or ams to serve the databases, nfs service could be Linux or Solaris, which have slightly different responses to load, afs could be used at a site or not. But the largest difference was in use of batch systems. We could not specify the batch systems in use at any sites, and many were put into use, with LSF, BQS, PBS, SGE, Condor, Codine, and others. To support these different batch systems a module abstraction layer to batch interaction was created, and a template with needed functions was created. Central development could not test each of the batch systems in use, but remote sites could build an interface from the template, and check in new batch system support to the code base. As other sites would modify and improve each interface, this proved to be a good development model, and produced stable interfaces rather quickly,

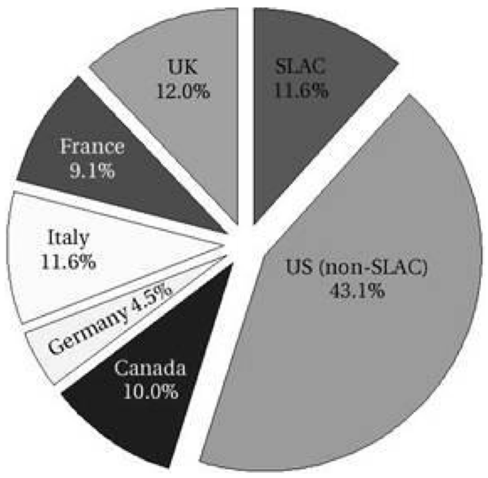

Fig. 2. How the production was distributed around the world for production cycle SP5, divided by country. The USA has more production sites within BaBar than any other country; jobs are well distributed among the existing sites.

without a need for central development to have detailed knowledge and test systems for each batch system in use.

Tools would freely get modified to satisfy different production site requirements. If a site had resources which could be used, we would find a way to use those resources. This resulted in 22 production sites in 6 countries on 2 continents: CalTech, CO State Univ., CO Univ. at Boulder, Iowa State Univ, Ohio State Univ, SLAC, SUNY Albany, Univ. Texas at Dallas, Univ. Tenn., Univ. Victoria, and Vanderbilt in North America; Birmingham, Bristol, IN2P3, FZK, INFN, Liverpool, Queen Mary, RAL, Royal Holloway, ScotGrid, and Tech. Univ. of Dresden in Europe.

The push to get work distributed among sites in SP5 was very successful, with no one site dominating production. The relative amount of data produced in SP5 by country is shown in Fig. 2.

VI. SP6 Improvements

BaBar's Computing Model 2 changes were put into production with SP6, as was commented on earlier [3], [4]. This increased the amount of control that production could have over what was produced, and with this control we were able to achieve a much lower failure rate for jobs. In SP5 the failure rate for jobs was between 4–6%. Most of this failure rate was because of the use of databases for production, producing strange restrictions on production (such as only one job starting per minute into a database, carefully tuned container sizes, data caches to be tweaked, etc.). But with more control over the output files, we were able to get the SP6 failure rate down to 0.2–0.5%, where most of this failure is now due to hardware. With this lower failure rate and the removal of a database to support, the effort for production managers was also reduced. The average of one half-time person per site has now been reduced to only one tenth-time person per site. Many site managers have reported that things are now stable enough to run for a week, without any interaction beyond just checking on the status of jobs.

VII. Comments on GRID Use

This has been a presentation of a large scale distributed computing effort, and it sounds like it should be a GRID talk, but it is not. There has been some activity within the BaBar simulation production involving GridPP resources in the UK, and INFN-Grid resources in Italy [6]. Both approaches are converging to a unique model based on the LCG middleware. These are development projects and at this point only provide for a small amount of the current production.

The production effort within BaBar started well before there were any Grid projects, but as the Grid project continue to mature the production team have watched them to see what we could use. Recently, the Grid has not been stable enough for our simulation production to be useable, since we need to get production out stably, day after day. Also, until recently the Grid was not installed on enough resources throughout the world to be able to provide the needed production for BaBar. The resources now installed are more than adequate, although shared with several other large scale efforts. But even so the Grid requires more effort at this time than the use of the current tools, and current dedicated cpus already in use.

Grid production has been proven to work for BaBar, but only with heroic efforts. Until the Grid proves to be more stable in development than it currently is, and easier to use for our production effort, these will have to be development projects. For now the Grid is not the answer for BaBar simulation production. But this will change, with future development within BaBar to better match Grid models, and within the Grid to have more stable tools which will be easier for large projects to use. The next couple of years with production, Grid use should prove to be interesting.

VIII. Conclusion

Production of simulated events for BaBar is a large computing effort requiring over 1000 cpus throughout the world (we are now at 1800 cpus and growing). Even though this is a large and difficult computing effort, it was done and on time for physics analysis, using a reasonable number of people.

To reduce the effort to a reasonable level, good tools are required, and need to be robust to handle failures without causing more work for producers, and be stable for at least three days. In any production most of the effort is spent in recovering from problems, tools need to be designed with automatic recovery if effort is truly to be reduced.

The system in BaBar is working well, producing needed events in a timely manner with a supportable effort in the collaboration. Improvements continue to be made, as sites add cpus as more sites come on-line. The system continues to scale well for increasing production, and we look forward to SP7 starting in the fall.

Acknowledgment

This work builds on a large body of development by the BaBar Computing Group, and would not have been possible without a strong collaborative effort.

References

[1] B. Aubert, "The BaBar detector," *Nucl. Instrum. Methods Phys. Res. A*, vol. A479, pp. 1–116, 2002.

[2] S. Agostinelli, "GEANT4: a Simulation Toolkit, ," *Nucl. Instrum. Methods Phys Res. A*, vol. A506, pp. 250–303, 2003.

[3] P. Elmer, "BaBar computing—From collisions to physics results," presented at the 2004 Conf. for Computing in High-Energy and Nuclear Physics (CHEP04), Interlaken, Switzerland.

[4] M. Steinke and P. Elmer, "How to build an event store—The new kanga event store for BaBar," presented at the 2004 Conf. for Computing in High-Energy and Nuclear Physics (CHEP04), Interlaken, Switzerland.

[5] D. Smith *et al.*, "Global management of BaBar simulation production," presented at the 2003 Conf. for Computing in High-Energy and Nuclear Physics (CHEP03), La Jolla, CA.

[6] C. Bozzi, "Using the Grid for the BaBar experiment," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 5, pt. 1, pp. 2045–2049, Oct. 2004.