BABAR SIMULATION PRODUCTION – A MILLENNIUM OF WORK IN UNDER A YEAR

D. A. Smith, *SLAC, Menlo Park, CA, USA*
F. Blanc, *Univ. of Colorado, Boulder, CO, USA*
C. Bozzi, D. Andrfeotti, *INFN, Ferrara, Italy*

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. Proper tools are created to be flexible in handling errors and failures that happen in the system and respond accordingly, to reduce 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.

Comments and questions contact Douglas Smith (douglas@slac.stanford.edu)

SOME BABAR HISTORY

In early 2003, BaBar was into its third run cycle of data taking (run 3). The experiment already had nearly 80 fb^{-1} 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 for: three times the luminosity for generic B-Bbar events; matching luminosity for generic continuum events; and various signal decay modes as requested. These three requests are roughly similar in computing effort.

The total request translates to a number of events which needs 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 cycles of data. These cycles of simulation production are numbered, and this paper will mention three cycles in detail, SP4, SP5 and SP6. In 2002, SP4 and 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 recognised 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 I will concentrate on this effort as a description of a complete large scale computing effort.

RESOURCES NEEDED

Assuming a fictional 1GHz pentium III machine, we can look at the resources needed at an event level. There is a range of the computing time 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 45kB per event. When averaging over decay modes the time per event is 8 seconds, and the data produced per event is 40kB.

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

The production of these events is divided up into computing jobs, where each job will produce 1000 or 2000 events. On the fictional 1GHz pentium III machine the jobs would take 2.3 or 4.5 hours to complete, on average. The final 2.2 billion events was created with over 1.1 million computing jobs.

The code to run these jobs was developed in BaBar to run on Solaris and Linux systems. In the SP4 production cycle, still some amount of simulation was produce using Solaris. But by 2002, when SP5 would be starting, all new cpu 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 1.1 million jobs on Linux machines, each taking 4.5 hours on average to run.

AN OVERVIEW OF PRODUCTION

Figure 1 shows a plot covering most of the time period of the SP5 production that started in Jan. 2003, including some of the time period before and after to show 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 cycle overlap in time from SP4 to SP5 and SP6. 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 generation and Geant 4 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 1.8 million jobs. Also these jobs would be shorter, so keeping the queue full efficiently was a hard problem, and there were three times more failures to track, where each failure would effect 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 effect 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 time less server load). Each job was now longer, so the batch queues could be used more efficiently. This would result 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 would now require 512 MB of memory to run (previously it was only 256MB of memory needed), and there would

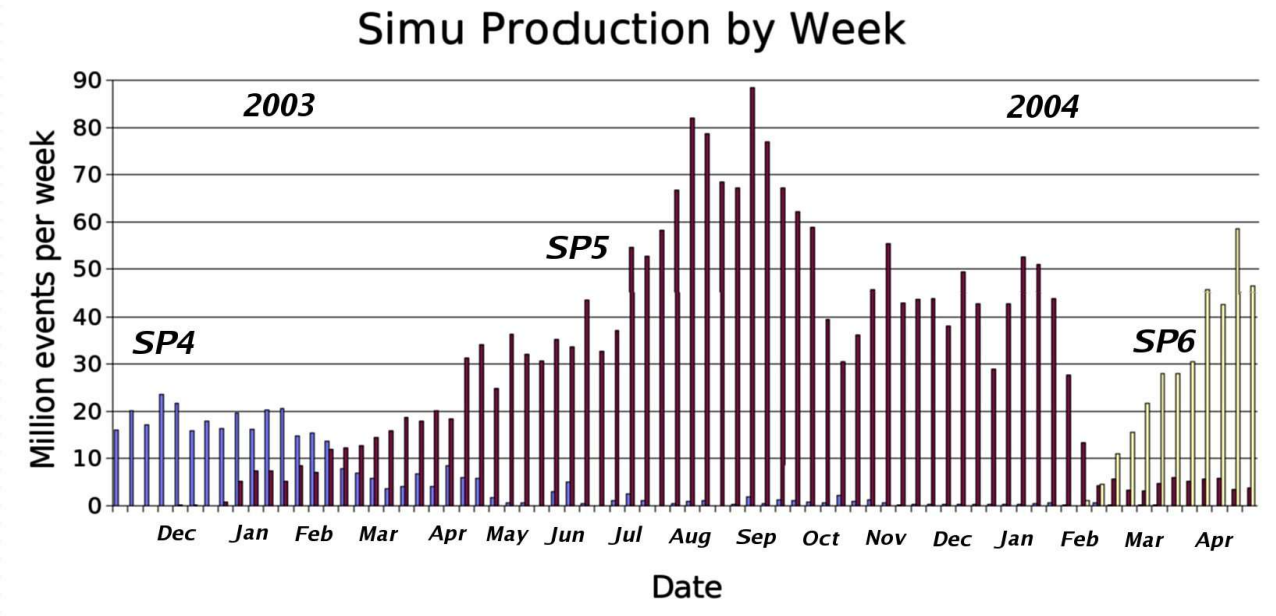

Figure 1 : Simulation production in the BaBar experiment by week, for the time period covering the SP5 cycle of production. The figure displays the different cycles of production in BaBar, and how they overlap in time. The details of the simulation cycles is described in the text.

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. This new computing model included a number of base changes in BaBar computing, but the one to most effect production was the change from an event store based on Objectivity databases, to an event store based on Root I/O. This change meant control code would now have to manage the production and distribution of root files. Although this increased the complexity of the control scripts compared to Objectivity use (where the Objy. code would control the files produced), the added control over production freed up how production could be done, and again increased efficiency. Production was now done into a file structure, and removed the overhead of maintaining a database. This drastically reduced server load again, and reduced the chance for job failure, 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.

METHOD OF PRODUCTION

Since the SP5 data would be needed by the beginning of 2004, and since 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 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 need to become distributed to more of the institutions in the BaBar collaboration. Simulation production was sited as a good candidate for distribution. Lack of required resource at SLAC would not be a problem, since the needed computing resources would be found at any BaBar institution that could provide them.

But when increasing the number of sites, one must be careful not to also multiply the effort to the collaboration. In the early days, production at SLAC was done with 3 people working in shifts. The 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 this large amount of production done, without increasing the total effort, good tools needed to be created. A set of tools providing a number of services was created called “ProdTools”. These were command line tools and libraries to help the production managers get things done. Also they would provide control for the jobs, so they could get done efficiently with the provided resources for each site. In the case of SP5, they provide also for some specific requirements to be able to use the Objectivity database in a production environment (such as only one job could start in the database per minute).

ProdTools provided an interface between the central production database at SLAC, and the local batch systems at each site (see Ref [1]). 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 submit. More information about this system can be found form earlier CHEP conferences, since the current tools are only an evolution of what was described before.

But the main point of the tools was not the submitting of jobs, but being able to recognise when the jobs fail, and how to fix these failures. In job production most of the effort was not in the setup and submission of jobs, but in recognizing failures, and getting them fixed. 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 of development 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 can 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 last unknown jobs and accept that they would not get done.

The goal of the tools is to get as much done with as little human effort. To do this the tools include as much as possible automation and error checking.

Along with ProdTools to manage the jobs, there was a tool developed to manage the transfer of the produced Objectivity databases, which was called “MocaEspresso”. This tool would recognise the closed Objectivity 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 one day, they would need to be able to handle 400GB the next day.

REMOTE SITE RESOURCES

Most of the management and control of the jobs was handled by one system at SLAC, but the production would get done at remote sites, and there were certain required resources needed to be able to run BaBar jobs. For SP5 there was the Objectivity database to setup and hold the produced events until databases could be closed for export. This database set-up 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 500GB of space.

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

The sites set-up were very diverse within these requirements. Many of the academic sites had obtained funding to be a dedicated production site, and they were set-up 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 set-up in some manner which we could not change. Batch queues would have to share resources, with varying numbers of cpu used for production. 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 infrastructure, with sites using either nfs or ams to serve the Objy. databases, nfs service could be Linux or Solaris, which have slightly different response 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 on which functions needed to exist 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, 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.

BABAR COMPUTING MODEL 2

There was an epilogue to the SP5 story earlier this year. With the change to the event store format in BaBar's computing model 2 plans, there was a need to convert the data produced in SP5 from the Objy. database to root files. Only the minimum requested 1.6 billion events in SP5 would get converted in this effort, to reduce amount

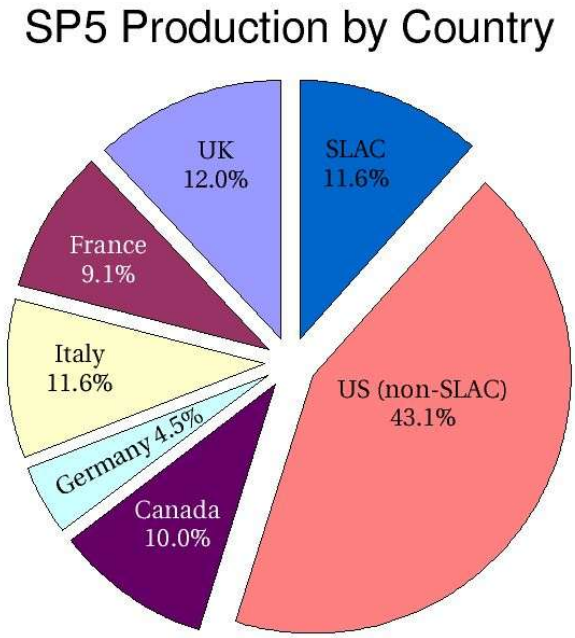

Figure 2 : How the production was distributed around the world for production cycle SP5, divided by country. The USA has more production sites within BaBar than any other country, jobs are well distributed among the existing sites.

of computing resources needed, and get data out to users in time for summer conferences. The conversion was going to take about 1 sec. per event, and there would be a different job for each output from the original production jobs. The production jobs were for 2000 events each, so the conversion jobs would be only 40 minutes long each. The conversion effort was 600,000 jobs.

This was a one-time effort within BaBar, not a continuing production effort, so it was treated differently. There was a farm of about 1400 cpus set-up at SLAC to do most of this conversion, along with some conversion to get done at IN2P3.

The project took 6 weeks, and most of the conversion was done by one person. There was quite a bit of careful setup to finally get efficient production, requiring servers for the Objectivity databases, three servers for the condition databases, and three servers for the output root files. The resources were balanced with other conversion efforts, at one point it was possible to keep 1200 concurrent jobs running. Since each job would take only 40 min. on average, this meant a new job would have to be submitted every 1.5 sec., which was a tricky problem in itself to solve adequately. The conversion was done on time for users, averaging about 40 million events per day.

The Computing Model 2 changes were put into production with SP6, as was commented on earlier. 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 stable between 4-6%, and we could never really do better than that. Most of this failure rate was because of the use of Objy. databases for production, producing some strange restrictions on production (such as only one job starting per minute into a database, carefully tuned container sizes, Objy. 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 an Objy. 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 a person per site. Many site managers have reported that things are now stable enough to let run for a week, without any interaction beyond just checking on the status of jobs.

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 [4]. 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 people in production have watch them to see what we could use. Up to recently the Grid was not stable enough for our production to be able use it, 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 on the Grid are now 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 will be easier to use, 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 years with production and Grid use should prove to be interesting. In Ref. [4] a Grid of LCG farms is now properly configured and ably to produce events. This Grid will be used soon and will be treated by the current system as another production farm.

CONCLUSIONS

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 be made, as sites add cpu and more sites come on-line. The system continues to scale well for increasing production, and we look forward to SP7 starting in the fall.

REFERENCES

[1] D. Smith, “Global management of BaBar simulation production.” proceedings of CHEP03, 2003.

[2] C. Bozzi, et al. “Production of simulated events for the BaBar experiment by using LCG.” proceedings of CHEP04, 2004

[1] P. Elmer, “BaBar computing – From collisions to physics results.” proceedings for CHEP04, 2004.

[2] M. Steinke, P. Elmer, et al., “How to build an event store – The new Kanga Event Store for BaBar.” proceedings of CHEP04, 2004.