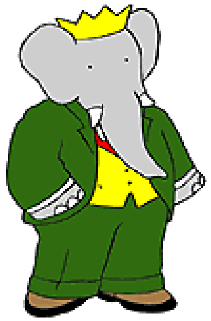


TM and © Laurent de Brunhoff

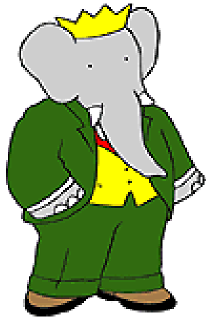
Management of BaBar simulation jobs

Global management of simulation jobs in BaBar:
an ad-hoc GRID made of spare parts.

Dr. Douglas Smith
Stanford Linear Accelerator Center

CHEP 2003

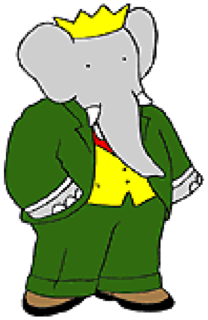
For the BaBar computing group.



TM and © Laurent de Brunhoff

Big problem, lots of jobs

- BaBar has a huge need for simulated events, a stated goal if 3 times the number of measured events.
- This is larger than any one computing site wants to handle, plus it is a stated goal to make simulation production in BaBar a distributed effort.
- In the end there are 100,000's of jobs over many months to manage world wide.

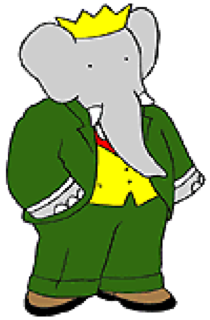


Built up over time

- A number of steps is needed for a system:
  - Centrally collect and manage requests from physicists for simulation events, and define jobs based on requests.
  - Distribute these jobs to remote production sites, and provide local management of jobs.
  - Import produced events from remote sites to main site for archiving and distribution to users.
  - Report on production in system to: track requests, monitor system, publish events for analysis.

Picture of a production site

- Jobs are resource intensive, and there are large requirements to be a production site:
  - ~0.5TB File server - Large amount of data imported for a job, but data is used for all jobs: need 36GB for conditions DB; and 10-200GB of background events, to get started. Also need Objectivity database to keep resident job information, and produced events another 300GB or so.
  - CPU to run jobs, anywhere from 12-100 cpu's per site.
  - Network access to SLAC, ~10 Mb/s, or as fast as possible for transfer of produced events (50-300GB per week).
- Now have about 2 dozen remote production sites.

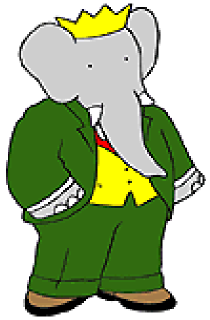

TM and © Laurent de Brunhoff

Management of requests.

- Web interface, with multiple dialog pages and web forms.
- Users can define 'decay modes', a defined set of inputs and control scripts for a job.
- Number of events can be requested for a decay modes, either a one time request, or repeated monthly.
- Production manager can accept/deny requests, set priorities, and finally divide requests of events into jobs for sites.

Tech's used:

- *Apache:*
  - *Fast scalable web server.*
- *Perl:*
  - *cgi scripts and dynamic web pages.*
- *Oracle database server:*
  - *relational database.*



TM and © Laurent de Brunhoff

Use of relational database

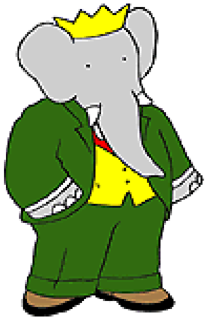
- Main bookkeeping of all requests and jobs by a relational database - we use Oracle.
- All decay modes, user requests, defined runs, and produced jobs are logged in the relational database.
- The web interface for requests and defined jobs connects to the database, and insert and updates records in database based on input to web forms.

Distribution of jobs to remote sites.

- Once a remote site is set up for jobs, actual information for jobs to be distributed per job is small (less than 1 kB).
- Utility is used to build jobs based on job information in the relational database. Job is built at the remote site based on site set up, no files are transferred.
- Once built the job is run at the remote site without any global central control.

Tech's used:

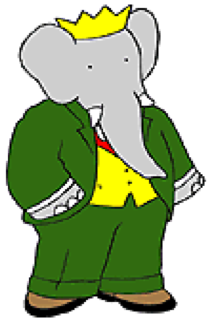
- *ProdTools:*
  - *set of utilities developed for management of jobs.*
- *Perl and Perl::DBI :*
  - *connection to relational DB through the database interface perl::DBI.*
- *Proxy server:*
  - *Manages remote connects to relational database.*



TM and © Laurent de Brunhoff

Use of database proxy

- The connections to the relational database are distributed around the world over the network using a proxy server to the relational database.
- Manages remote connections, and passes on database handles and SQL statement handles to utilities at remote sites.
- Works better than expected, handles multiple simultaneous connections, allows connection to sites world wide, more than adequate connection times and data transfer times.
- Now being used for other systems in BaBar.



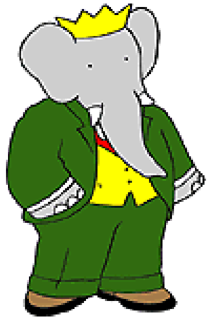
TM and © Laurent de Brunhoff

Local Management of jobs.

- Once jobs built at remote site, remote connection not needed to produce jobs.
- Submission to batch system managed by tools to make all batch system look the same, support for LSF, PBS, DQS, SGE, Codine, maybe others.
- Local tools manage jobs based on file system information and batch system output.
- All jobs produce data into one objectivity database.

Tech's used:

- *ProdTools (perl, bash)*
- *Objectivity database*
- *Network file service*
  - *using NFS or AMS*
- *Batch System*
  - *support for different batch systems, many used.*



TM and © Laurent de Brunhoff

Import of produced events.

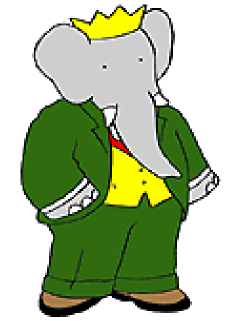
- Tool is used to look for closed objectivity database for import to SLAC. Once jobs have finished on database (no transactions) database can be closed.
- Lots of data to be transferred, 100GB-1TB, depending on site, and how often they import.
- File servers set up at SLAC for just this purpose, to keep files on disk until they can be linked into main production database and archived in HPSS.

Tech's used:

- *MocaEspresso:*
  - *Objectivity database imports.*
  - *Looks for closed database, and handles import of all files.*
- *bbFTP:*
  - *File transfer tool for large data transports over IP networks. Multiple streams, and data packet window size control.*
- *ssh:*
  - *used by bbFTP for authentication*

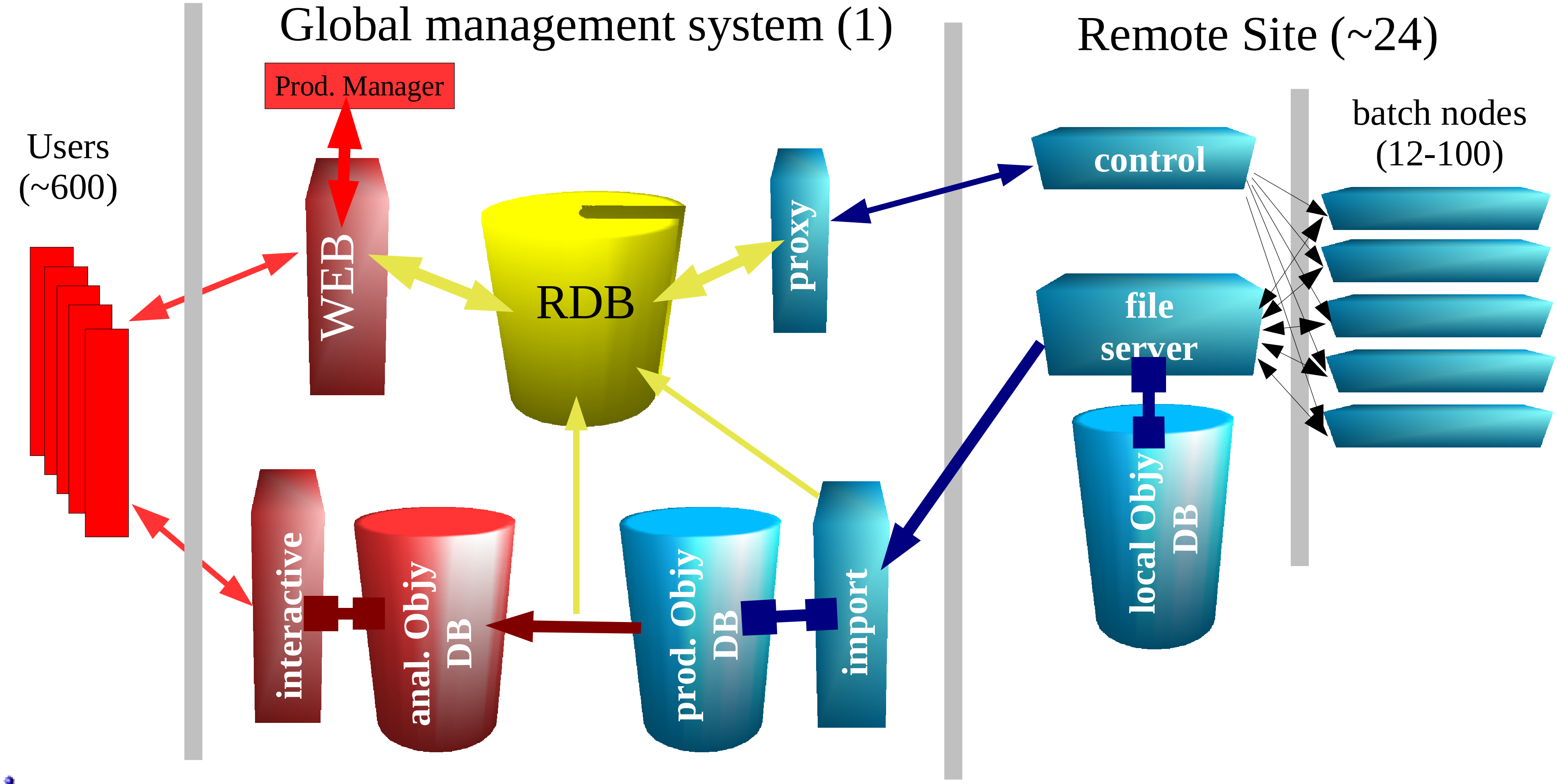
Distribution of events to users.

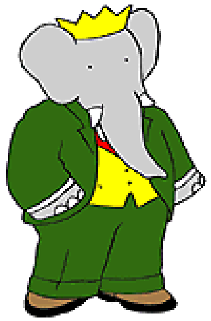
- Events from production database periodically 'swept' into analysis database for users.
- Also translation jobs used on events to produce other data formats for users.
- Reporting tool connects to relation database for people to list which events has been produced, and what jobs have failed.
- Web interface also used for reporting and system monitoring.



TM and © Laurent de Brunhoff

General cartoon of system

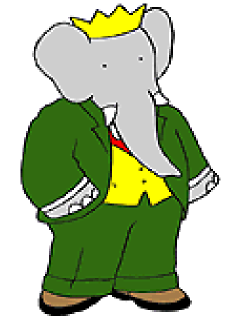




TM and © Laurent de Brunhoff

Overview of tools used:

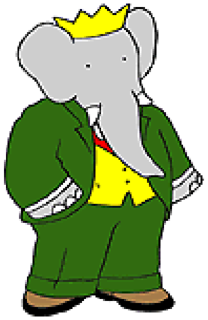
- Global management site (SLAC):
  - **Web server**: *Apache*; **Network accessed relational database**: *Oracle, proxy server*; **Event database**: *Objectivity*; **File import daemon**: *bbFTPd, ssh*; **Middleware utils**: *ProdTools,web cgi, perl.*
- Remote production sites:
  - **Event database**: *Objectivity*; **Network filesystem**: *NFS, AMS*; **Batch system**: *LSF, PBS, SGE,...*; **File transfer tools**: *bbFTP, ssh*; **Middleware utils**: *MocaEsspresso, ProdTools, perl.*



TM and © Laurent de Brunhoff

Summary of System

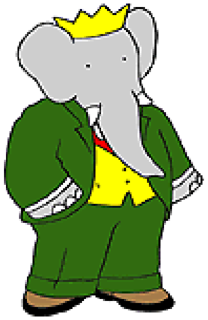
- Slowly evolving over time and growing:
  - now about 2 dozen remote sites.
  - equivalent of 1200 1GHz cpus used worldwide.
  - Import on average of 1/3TB per day for the past 1.5 years. (Based on last cycle of 1.4 billion events produced.)
  - Close to 1 million jobs have been done, where each job takes ~2 hours to do
  - Information about all jobs is the Oracle database.



TM and © Laurent de Brunhoff

Future of system

- New computing model in BaBar recently, and changes will be made to fit this model - biggest change is events not stored in Objectivity database.
- Not much change for job management, but big change in data management, and data import/export.
- Hopefully changes in import could greatly reduce amount of resident disk needed - easier to be a production site.
- In testing now, import/export in development, should in production by the fall (hopefully).



TM and © Laurent de Brunhoff

Other information:

- Please see other information about system at poster session:
  - Tuesday Session 10: “ Using Geant4 in the BaBar Simulation,” - D. Wright
  - Poster Session, Cat. 2:“ Using Grid for the Production of Monte Carlo Events in the BaBar Experiment” - E. Antonioli, et al.
  - Poster Session, Cat. 3: “ Production of Simulated Events in the Babar Experiment” - C. Bozzi, et al.