Visualization Tools for Monitoring and Evaluation of Distributed Computing Systems

Developed for the BaBar Prompt Reconstruction System

Ray F. Cowan and Gilbert Grosdidier

Presented by

Tom Glanzman

for the

BaBar Prompt Reconstruction and Computing Groups

at the

International Conference on Computing
in High Energy and Nuclear Physics

7–11 February 2000

Padova, Italy

Introduction

These tools were created to aid the development and commissioning of the BaBar Prompt Reconstruction system.

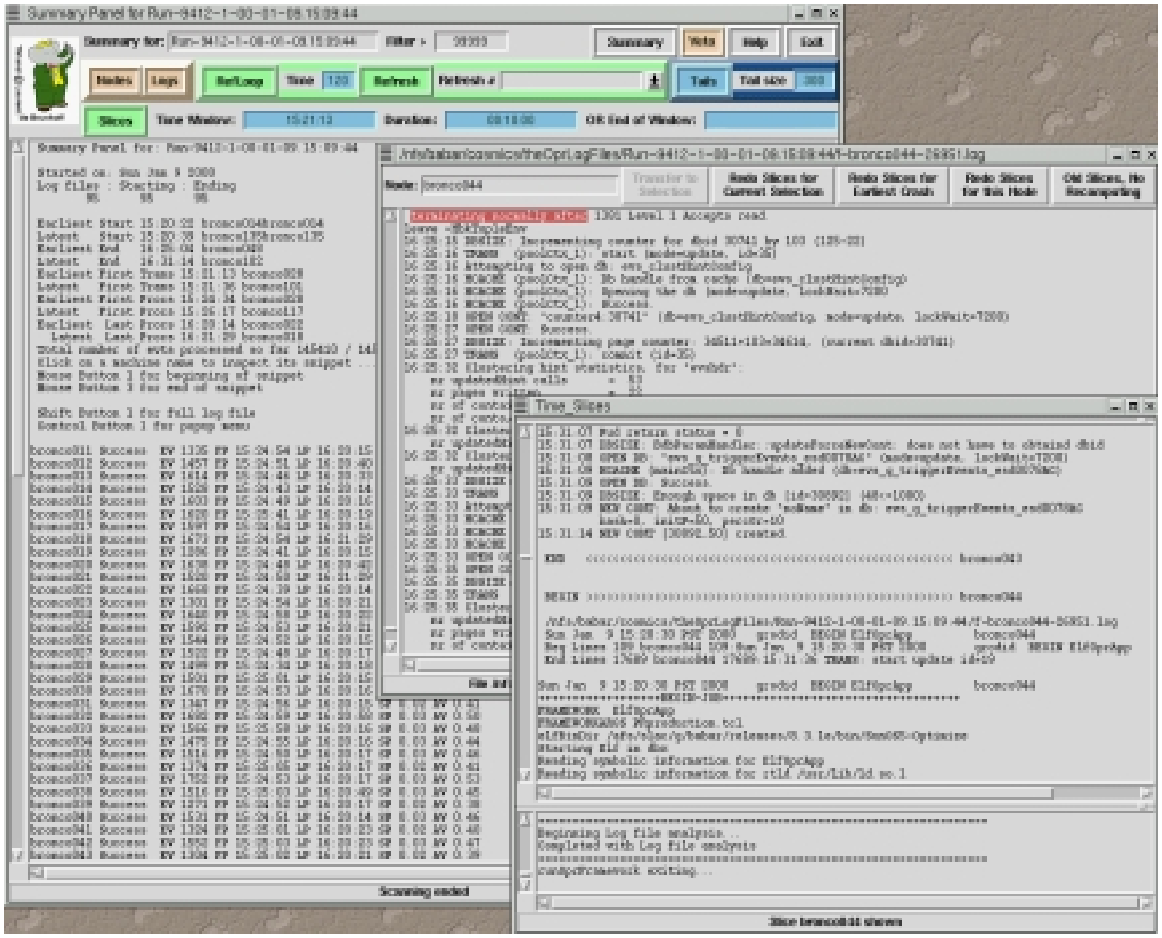
- The BaBar Prompt Reconstruction system:
  - performs event reconstruction of the BaBar datastream
  - uses a compute farm of up to several hundred nodes (Sun Ultra 5's/upgrading to Netras)
  - reads input from "tagged container" datafiles, up to 10 Gbytes per file.
  - transmits output data to an OO database:
    - output data are C++ objects
    - uses Objectivity/DB product
  - unit of processing is a BaBar "run"
    - the data taken during one PEP-II fill
    - up to 200,000–300,000 events
- Farm configuration:
  - a single node "logging manager" feeds input data to farm
  - farm nodes communicate with various servers
    - providing access to condition data
    - accept output C++ objects
    - store output on disk and in HPSS tape system

Monitoring Data Sources

Development of the farm has required significant effort. Numerous tools were developed to help identify problems and bottlenecks. These use several data sources:

- Individual farm node log files
    - each node runs a copy of the Prompt Reconstruction main application
    - actions are logged to plain-text log files, one per node
        - one log file may reach 10–15 MB in size
        - contains timestamps created each time the node progresses from one processing stage to the next
- Object database usage (`oolockmon`)
    - track lock activity
        - multiple access to objects is controlled by a locking mechanism
        - can be of two kinds: "read" or "update"
    - track numbers of userids, process ids, nodes accessing data as a function of time
- System performance data from `rperf`.
    - statistics gathered periodically by `cron` jobs
- Network use statistics
    - gathered on-demand by `snoop` and displayed by `tcptrace`
        - created an alternative display method for `tcptrace` output

Monitoring Tools: Log File Browser

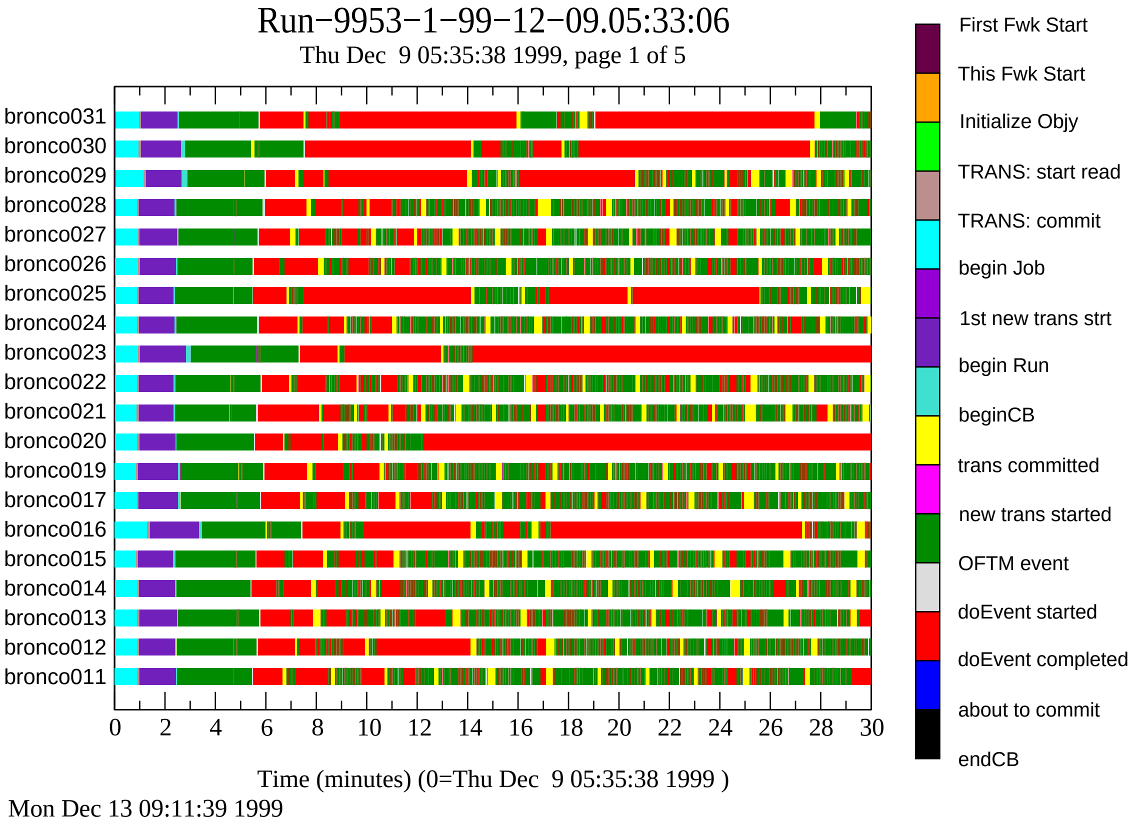


- OprLogScan is a multipanel X Window log file analyzer for BaBar Prompt Reconstruction monitoring.
- Analyzes simultaneously the hundreds of log files from the reconstruction jobs running on the Opr farm.
- It is crucial for spotting failures which could affect the running over all the nodes.
- OprLogScan is built using:
  - Perl 5.005
  - Perl/Tk 800.014

- OprLogScan is designed to:
  - browse the hundreds of log files
  - display a status summary for the whole job, and also for each file
  - select a given time slice over all the log files, and display it (using the power of Perl to analyze logs and to parse strings)
  - display selected full log files
  - navigate between the different windows/panels/subpanels (using the tag facilities offered through Perl/Tk)
  - operate string searches in any kind of display
  - print and/or save or ... any window content
- Operationally, OprLogScan is used to monitor the processing:
  - showing on every snapshot of the summary window differences with the previous one in reverse video display.

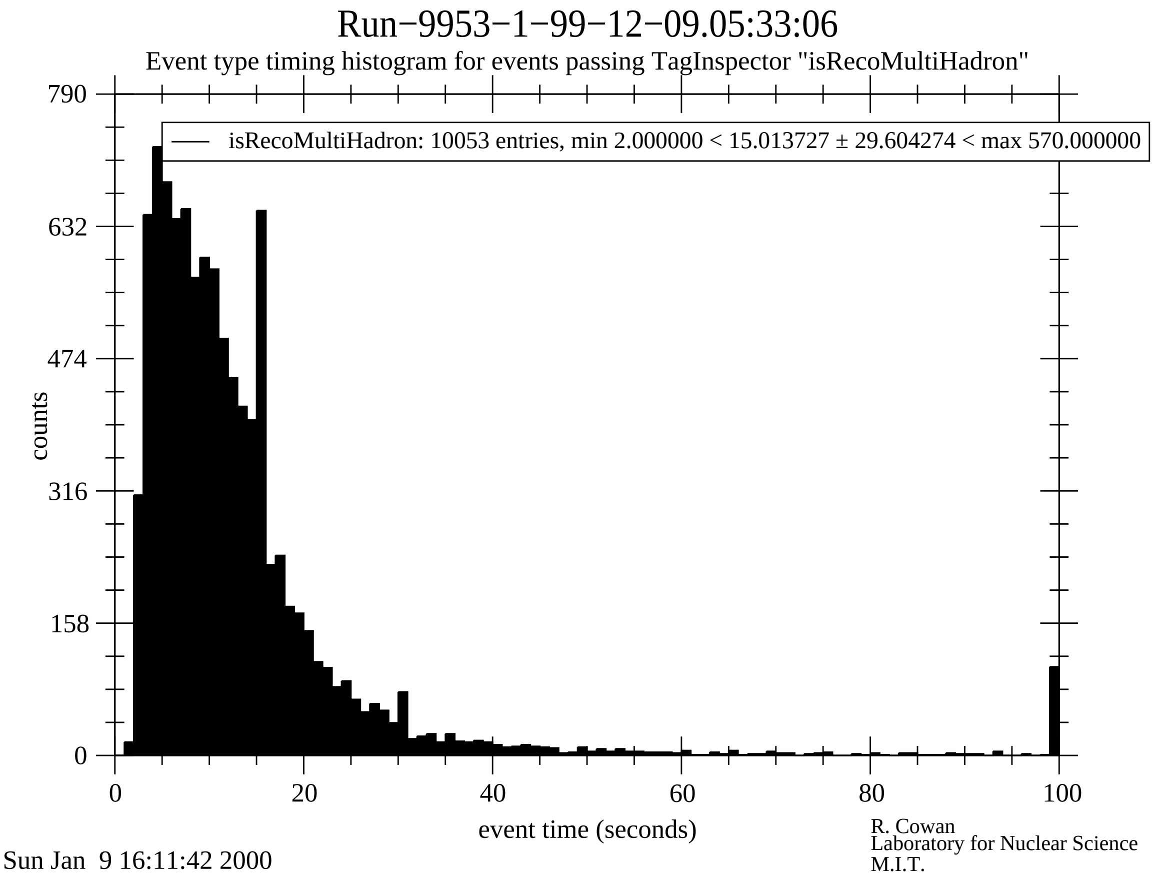
Monitoring Tools: Event Processing Times

- The processing steps of each event's data
  - are the same across all events and nodes
  - can be treated as a state machine
    - 16 possible states
- The small number of states
  - permits simple graphical depiction of each node's state as a function of time
- Tools were developed to capture and display the state transitions:
  - C-shell script scans the log files
    - extracts state transition times
  - These times are then used to create a graphics file
    - for the plotting utility `xmgrace`
  - These barchart-like plots can be time-sliced and zoomed
    - allowing detailed inspection

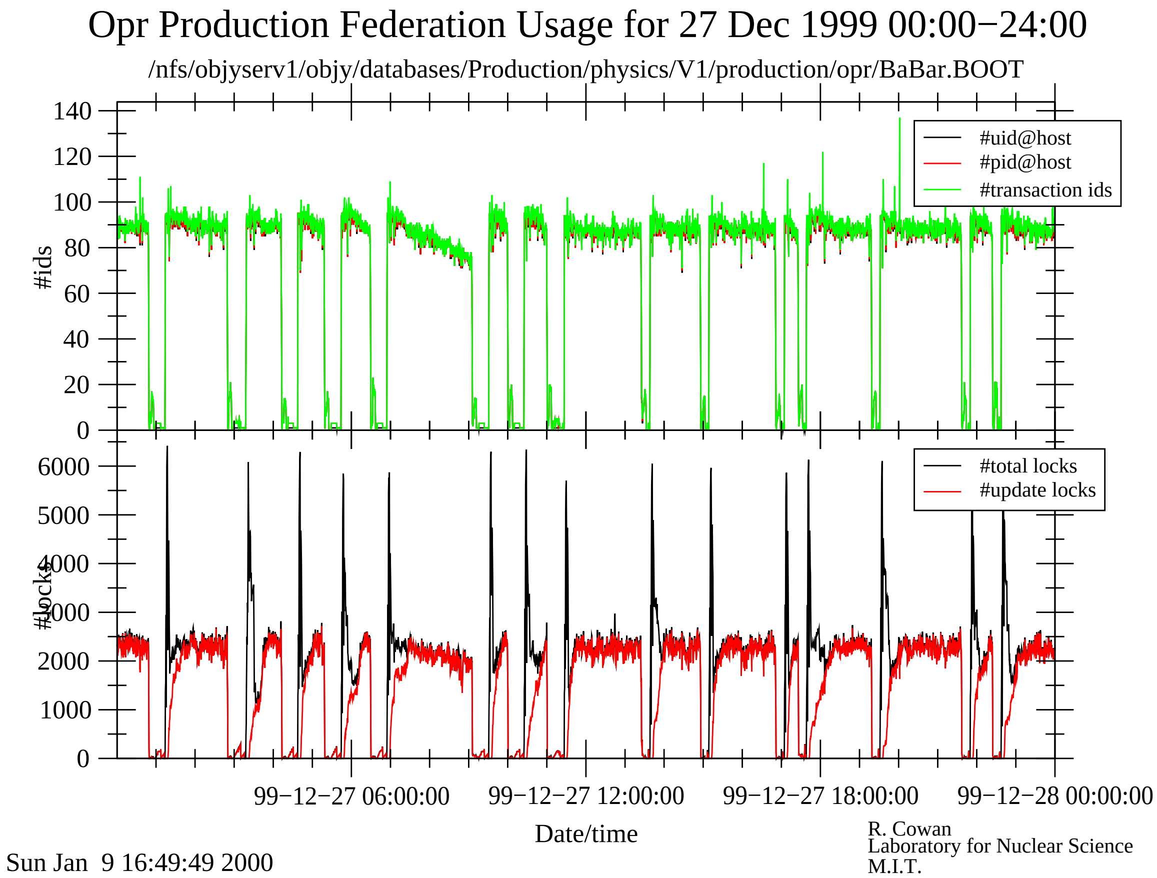


- Barchart strip display of state transitions for 20 nodes:
  - vertical key on the right identifies each state
  - individual node names are on the left axis
  - lengthy stretches of red indicate unexpectedly long times to transmit output data

- Event classification data is also extracted
  - identifies event as e^+e^-, multihadron, etc.
  - permits studies of processing characteristics based on type
  - permits histogramming processing time by event type



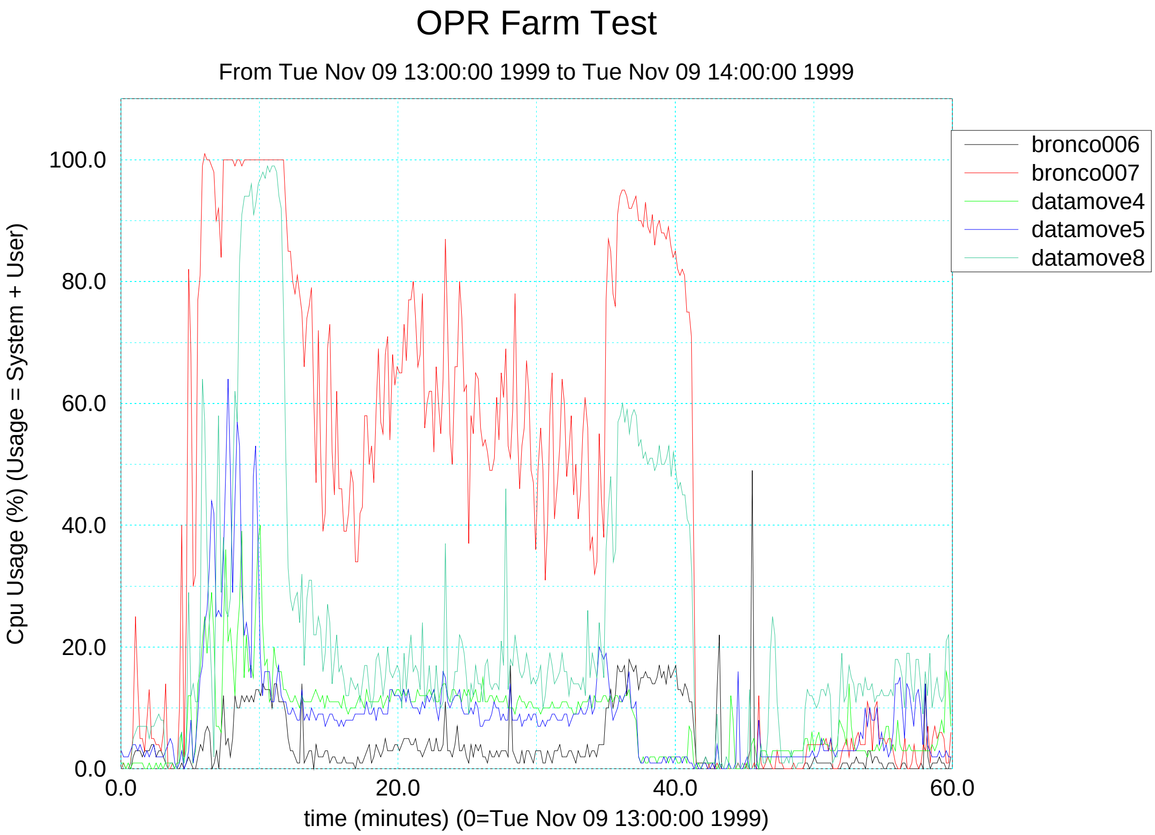
Monitoring Tools: Database Activity



- The OO database product provides an `oolockmon` utility which reports usage statistics:
  - each invocation gives a snapshot of activity at that time
  - it is queried periodically by a `cron` job
    - it reports read and update lock activity
    - it reports userids, process ids, and other usage measures
- A tool to display this information has been developed
  - scans and plots accumulated oolockmon statistics

Monitoring Tools: System Statistics

- The accumulated system statistics output from the Unix utility **`rperf`** are used
  - any **`rperf`**-reported quantity may be plotted



Conclusions

- Development of these tools was crucial to a timely understanding of the BaBar Prompt Reconstruction system
  - each addressed a particular set of problems
  - each provided information critical in overcoming development difficulties
  - cumulative effect was very successful in allowing the project to move forward
  - also provide an archival history of the system
- These tools are adaptable
  - to monitor other quantities
    - replacing front-end data extraction portions is all that is necessary
  - have been used to study behavior of other BaBar processing environments
- Tools are still undergoing development
  - as software changes occur
  - new diagnostic capabilities added every month or two to address new behavior