A Generic Multi-node State Monitoring System

James A. Hamilton
Gregory P. Dubois- Felsmann
Rainer Bartoldus

For the BaBar Computing Group
What is wrong with this picture?
Detector

SVT  DCH  EMT  DRC  IFR

Level 3 Trigger Processing Farm

Switch

Dataflow Proxy

Run Control

GUI

Component Proxies

BaBar Online Structure

What is wrong with this picture?

Run Control has no control or visibility into state of level 3 farm

Control

data

status

Level 3 Trigger Processing Farm

what is wrong with this picture?

Run Control has no control or visibility into state of level 3 farm
BaBar Online Structure

Detector

Level 3 Trigger Processing Farm

Switch

Dataflow Proxy

Run Control

Multi-node State Manager Proxy

GUI

GUI

Component Proxies

SVT
DCH
EMT
DRC
IFR

SVT
DCH
EMT
DRC
IFR

data

control

status
Alternative Implementations

- Fold into dataflow proxy
  - Destabilize existing complex subsystem
  - No GUI

- Implement directly in Run Control
  - Uses SMI (State Management Interface – Boda Franek)
  - Difficult to support desired variability
  - Need for timeouts in state monitoring
The Generic Solution

SMI Objects → Multi-node State Manager

- Unix Pipe
- DIM

... ...

Manager Daemons → Managed Subprocesses
Communication Subsystem

- Used for sending commands and reporting state
- Wanted TCP level of connection
- DIM (Distributed Information Management System) from CERN
  - Thin layer over TCP
  - Provides useful naming service
  - Robust disconnect/reconnect support
  - Would prefer TCP-like read/write API
Manager Operation

- NOT another state machine
- Adopts state machine of daemons
  - Subset of daemon states become manager states
- Understands READY and ERROR states
- Understands START and RESET commands
  - Other commands passed through to daemons
**Daemon Operation**

- Yet another state machine
- Controlled by SMF (State Machine Framework – Alex Samuel) state diagram
- Starts and monitors subprocess to do actual work.
State Reporting

- Major states
  - States that are reported back to SMI objects
    - Examples: ALLOCATED, CONFIGURED, RUNNING
- Minor states
  - States that display only on manager GUI
    - Examples: CONNECTING, MAPPED
- Micro states
  - Internal states of the daemon
    - Examples: starting
- Error states
  - Cause the manager to enter ERROR state
    - Examples: ERROR_EXIT, SIGNALED
Daemon State Transitions

- Commands from Manager
  - START, RESET, DISCONNECT, typical
- Subprocess termination
- Subprocess event reports (e.g. dataflow transition events)
Error Conditions

- Multiple daemons report either of the following:
  - Any error state
  - Disappearance (network disconnect)
- Timeout after START
- Daemons enter conflicting states
- Timeout after some, but not all, daemons change state.
- No commands sent to daemons until RESET
Detection of Hung Daemons

- If a daemon fails to respond to RESET by going to READY, it is marked unavailable, and will not be used in future operations.

- It can be restarted from the GUI.
Configurable Parameters

- Minimum number of nodes needed to START
- Maximum number of nodes to use
- Maximum number of errors to allow
- Length of state convergence timeout
Separate GUI

- One button for each node
  - Allows kill, restart, and view logfile
  - Displays node’s current state
  - States have colors, defined and reported by daemon

- Set configuration parameters
- Displays manager state and description of most recent action
Another Application

Multi-node Fast Monitoring

Currently 47 nodes

Currently 15 nodes

Event Data

Sampled Events

Monitoring Histograms

Run Control

Level 3 Manager

FastMon Control

FastMon Manager
Conclusions

- Improved performance
  - Cold start time reduced to process invocation

- Improved reliability
  - Eliminated race conditions
  - Robust error behavior

- Operational since November, 2002