Control of large scale distributed DAQ/trigger systems in the networked PC era

Gennady Briskin
Michel Clements
Dave Cutts
Sean Mattingly

Brown University

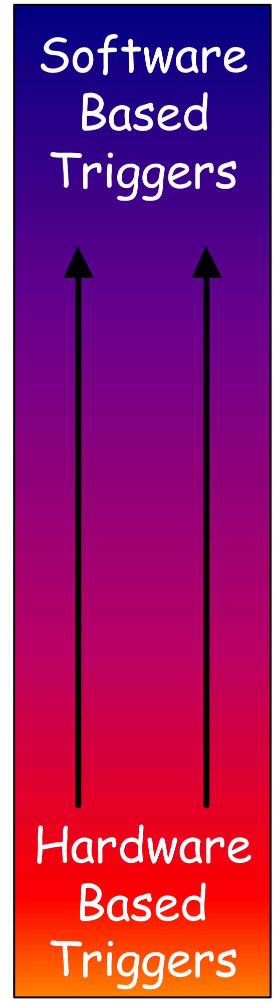
Toby Burnett
Kareem Kazkaz
Gordon Watts

DAQ2000 Workshop
Nuclear Science Symposium and Medical Imaging Conference
October 20, 2000

The Trigger Environment

- Pressure on the Trigger
  - Shorter decision times required
  - More complex algorithms
  - More algorithms
  - More Data
- Hardware Triggers
  - Inflexible, dedicated
  - Hard to change as accelerator condition change
  - FPGAs: Hardware moving into firmware.
  - (but) Well understood
- Cheap Commodity Components
  - High speed networks
  - Cheap memory
  - Cheap CPU power as never before
- The ERA of the PC Based Trigger
  - Farms of Triggers where traditional hardware exists
    - BTeV – very little in the way of a HW trigger

Speed!



Control of large scale distributed DAQ/trigger systems in the networked PC era

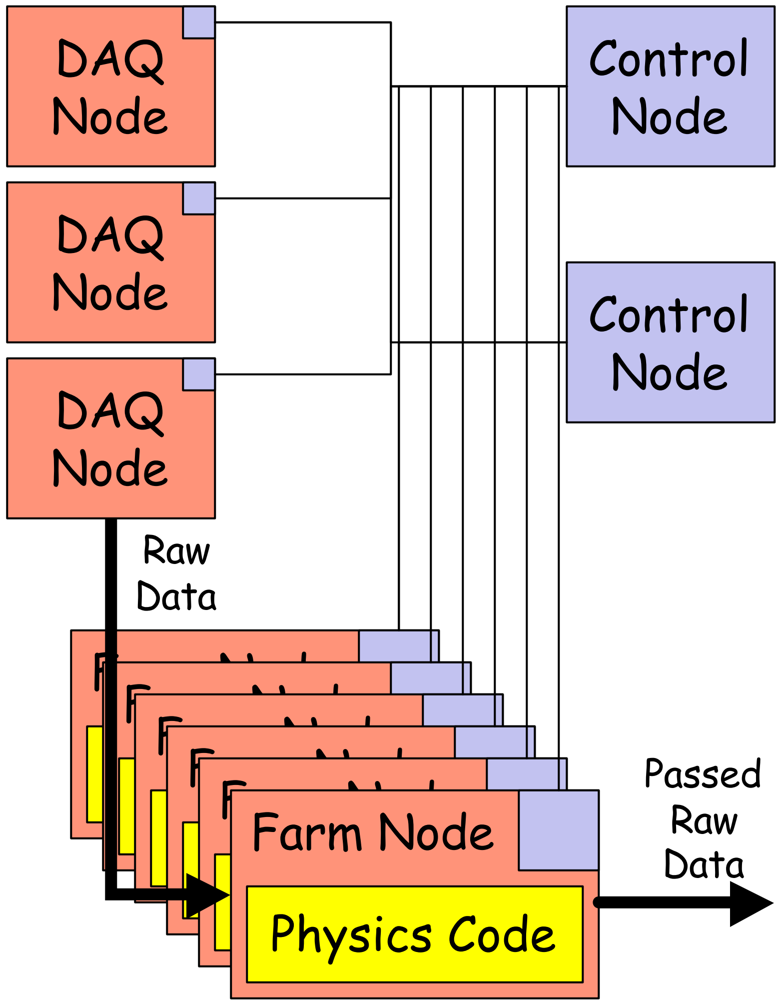
Gordon Watts
gwatts@u.washington.edu

University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000

Farm Based Triggers/DAQ

- Control Code
  - Management
  - Configuration
  - Monitoring
  - Error Recovery
- DAQ Code
  - Controls DAQ components
  - Network Control & Routing for farm
- Physics Code
  - Performs actual trigger decision
  - Frequently interfaces with others



All code, except the control code, is independent of each other

Control of large scale distributed DAQ/trigger systems in the networked PC era

Gordon Watts
gwatts@u.washington.edu
University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000

Control

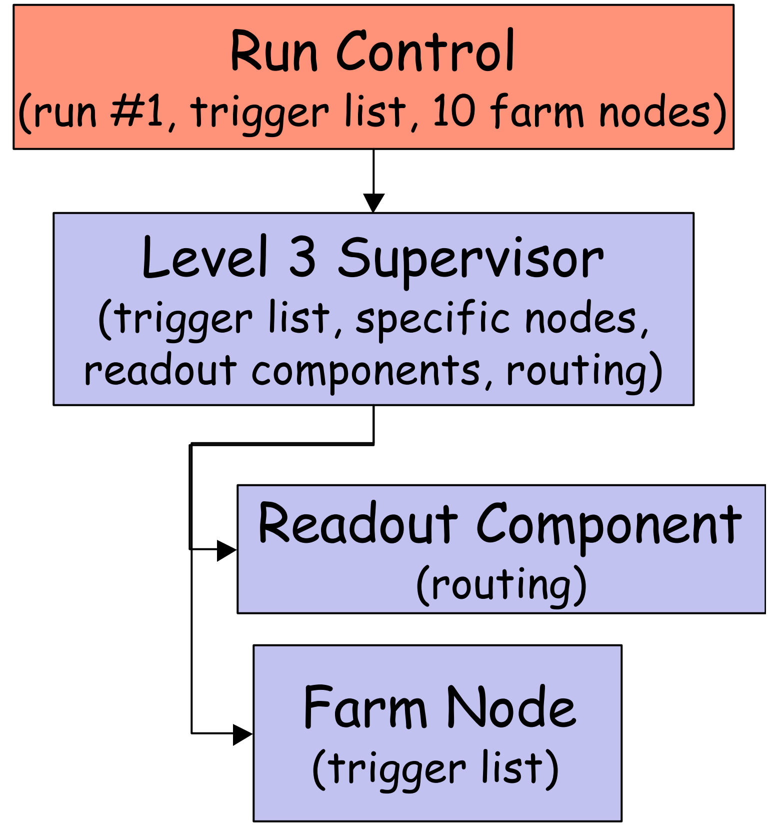
- Configuration Tasks
  - Management
    - Interface with other parts of Online System
    - Resource Management
    - Controls other components
  - Configuration
    - Event Routing
    - Farm Partitions
      - Multiple Runs
    - Trigger Code
    - DAQ parameters
  - Monitoring
    - System Performance
    - Resource Usage
    - RT Debugging (hardware & software)
  - Error Recovery
    - Asynchronous and synchronous errors
    - Crashed nodes!
    - On the fly reconfiguration
      - Must decide if possible!
    - Diagnostic information for shifters & experts

Gordon Watts
gwatts@u.washington.edu
University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000

The Environment: Distributed Run Control

- Run Control – Farm Interaction changing
  - Simple RC program used to handle whole DAQ/Trigger system
    - FECs, Trigger, DAQ, etc.
  - DØ Run 1: RC was complex
  - DØ Run 2: To complex for a single RC. → Distributed
- L3 (Farm) Supervisor has more responsibility
  - Hides farm implementation details from RC.
    - Black box for the rest of the online system
  - More flexibility in how the Trigger/DAQ system is managed.



- Components can fail in way that doesn't bring down the run
  - A node crash
- Allows it to recover from errors in farm and DAQ without having to involve RC when possible

Control of large scale distributed DAQ/trigger systems in the networked PC era

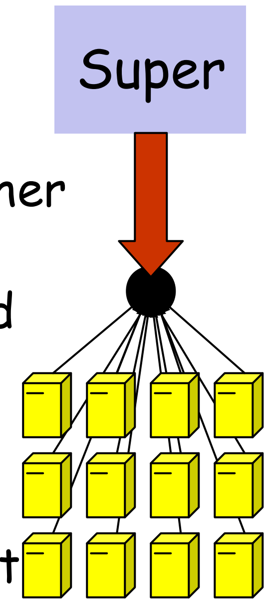
Gordon Watts
gwatts@u.washington.edu

University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000

Design of Supervisor

- Goals
  - Configuration
    - Effectively manage transitions from one state to another
      - running, paused, new run, etc.
    - A large number of separate entities must be controlled
    - Speed issues
      - Recovery, short accelerator times between shots, etc.
  - Recovery
    - Physics code will change through out life of experiment
      - Test Test Test!
      - Will still have crashes & explosions
    - One of the few systems that can suffer a major component failure and not terminate a run!
- State Diagrams – standard approach
  - Use a state definition language to define each transition in the farm
    - Reconfiguration requests that don't change the global state
    - Difficult to deal with exception conditions, asynchronous errors.

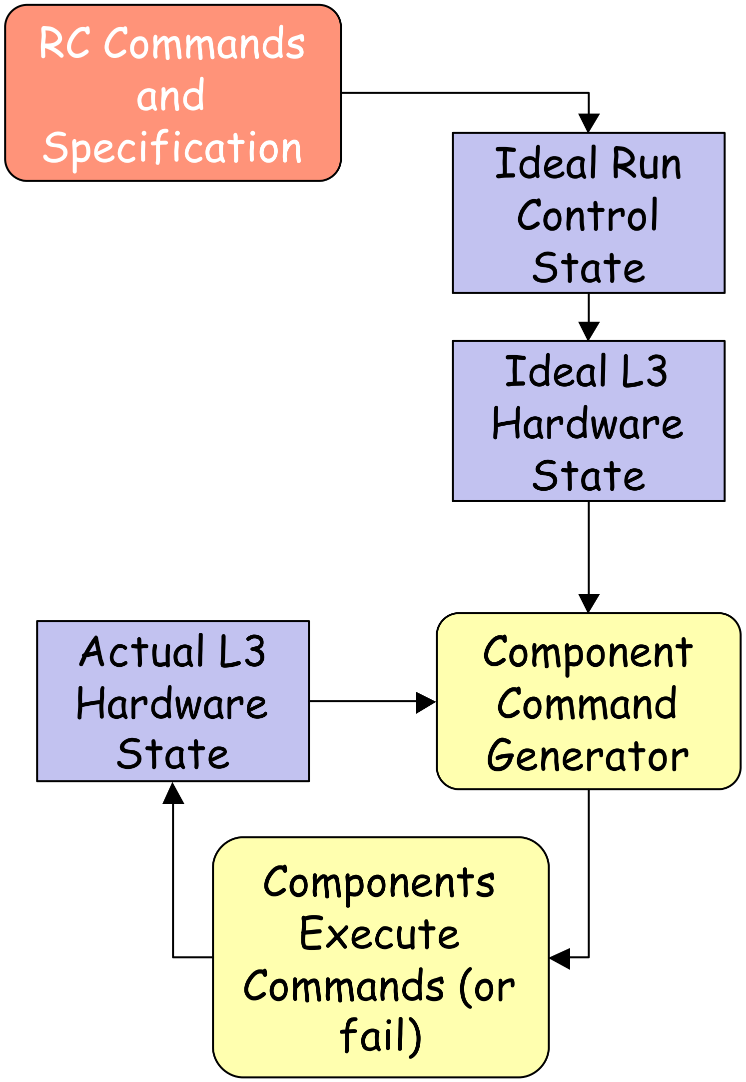


Gordon Watts
gwatts@u.washington.edu
University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000

The Ideal State

- Maintain an ideal state
  - Ideal state is built only if RC commands make sense
  - Eliminates erroneous input
  - Prevents leaving the farm in an unknown state due to a RC failure
- There is a mapping from an ideal state to a Hardware State/configuration
- Maintain Actual State
  - Comparison between ideal and actual generates commands
- Did a RC Configuration Change Succeed?
  - Are the Actual and Ideal states close enough to report success?



Control of large scale distributed DAQ/trigger systems in the networked PC era

Gordon Watts
gwatts@u.washington.edu

University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000

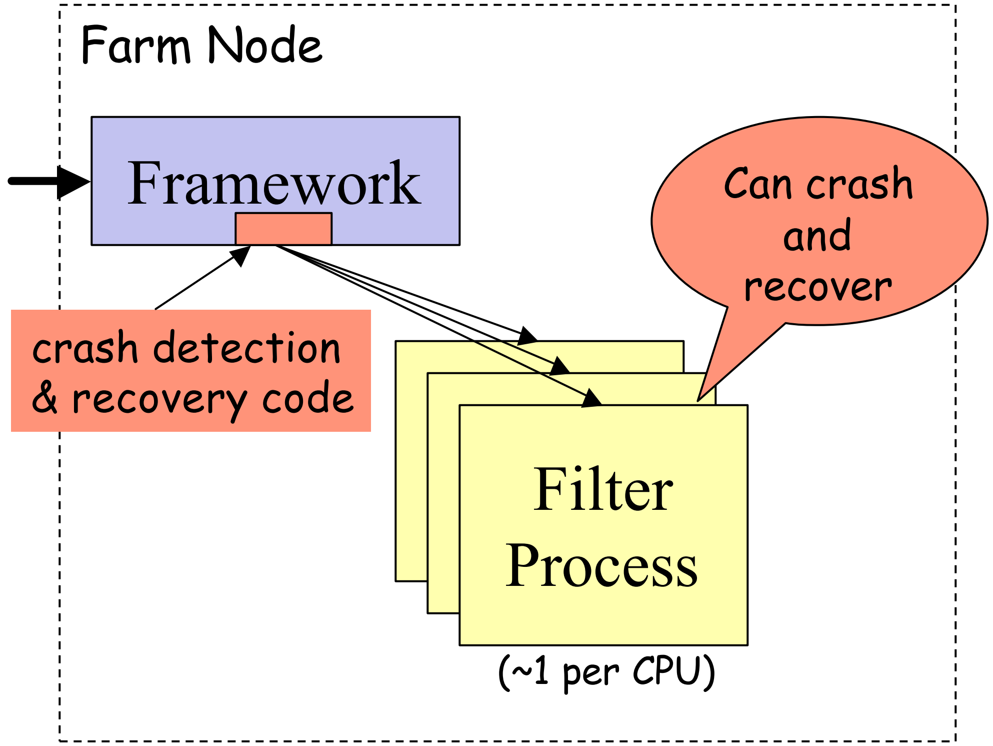
Faults & Errors

- Ultimate goal is 100% uptime; 0% trigger deadtime!
- Faults are rarely predictable.
  - Hardware: Always knew every input and output
  - In software, this is no longer always true
- Two classes
  - Catastrophes
    - Whole Farm Crashes, on every event
    - Critical Readout component fails to configure correctly
  - Minor errors
    - Low grade crashes
      - In Run 1 we lost the use of our build system due to a licensing issue for a short while: had to deal with a farm crash once per 20 minutes.
    - Non-critical component fails to configure
- Failure Mode Analysis
  - Will tell you the most likely failures
  - And the places that a failure will do the most damage.
  - Helps to focus error recovery efforts

Still need humans!

Farm Node Failures

- Separate the physics code from the Framework and Control Code
  - Physics code changes over life of the experiment
  - Separate memory space
  - Assure that data back from the Filter Process is not interpreted by Framework at all
    - Blindly handed off to Tape
  - Automatically restart a dead filter process
    - Or start debugger on it without interrupting other filter processes



A Special Case Recovery

indicated by a failure mode analysis

Control of large scale distributed DAQ/trigger systems in the networked PC era

Gordon Watts
gwatts@u.washington.edu

University of Washington, Seattle

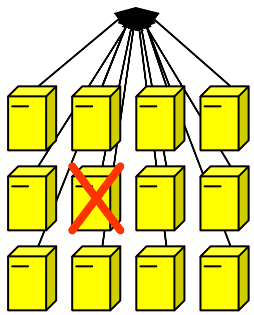
DAQ2000
NSS2000
Oct 20, 2000

General Recovery

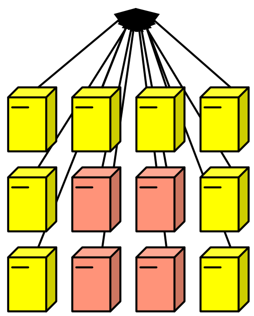
- At Failure
  - Supervisor is notified of the failure
  - The Actual State is updated to reflect the failure
    - There is an understanding of what constitutes a minimal working L3 system
    - Stop Run is issued if this test isn't satisfied.
- Recovery
  - Notify Shifter of problem, or
  - Measure current state against ideal state and re-issue commands
    - Will repair broken device/node
    - Reconfigure the farm (expensive)
    - Type of failure can effect the hardware state
      - Means special recovery commands can be automatically generated
    - Repeated Failure Detection should be implemented
    - No need to notify RC
    - Accounting (trigger information)

Local/Global Fault Detection

- Local
  - Timeout from command, etc.
  - Explicitly designed detection in control code
  - Must be careful to track the source of the error and record it in the Actual State
  - real-time detection
  - Similar to old-time hardware faults
    - Know exactly where the fault occurred
    - Limited number of fault types
    - Specific – simpler for computers to handle directly.



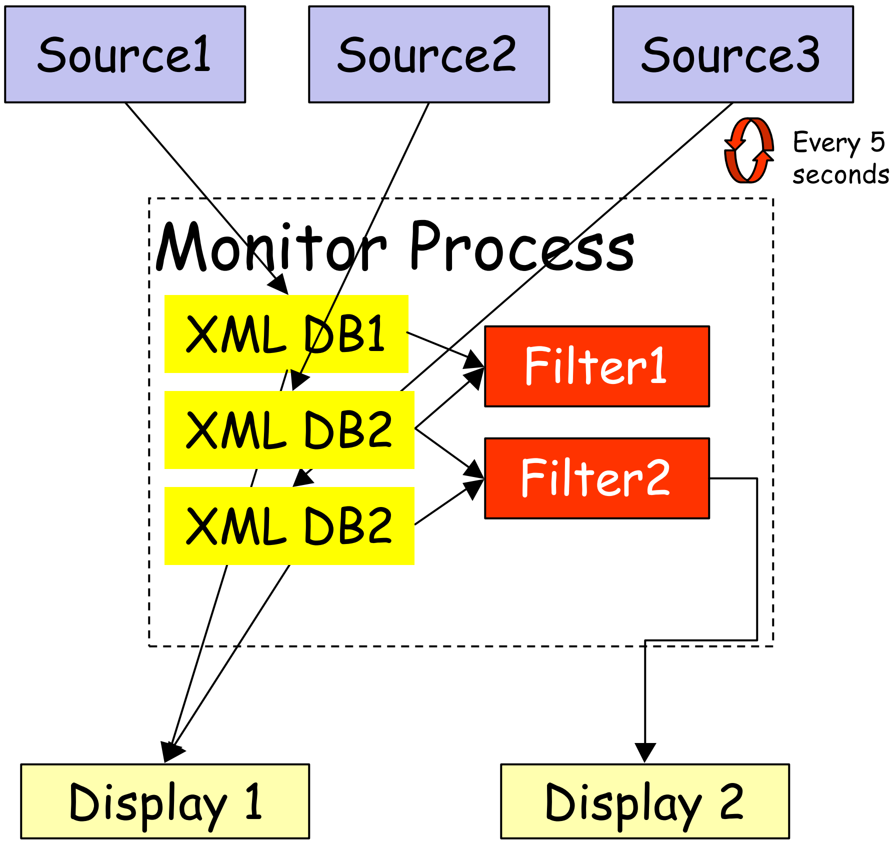
- Global
  - Single monitor point can't determine sickness
  - Global Monitoring is crucial for this
  - Useful to help diagnose problems for the shifters
    - Recommend solutions?
    - One day… it will go farther!
      - Input to rule based system??
  - Detection isn't on an event-by-event basis



Flexible Monitoring

- Monitoring
  - XML Based
  - Generic WEB front-end
    - take advantage of commodity software market!
  - Drop-in Aggregate Filters.
  - Displays query via http requests
    - Cross platform
    - SOAP?
    - Custom designed displays (graphics) or simple static web pages
  - Use actual program in limited circumstances
  - Interface for rule-based system??

Java?



Control of large scale distributed DAQ/trigger systems in the networked PC era

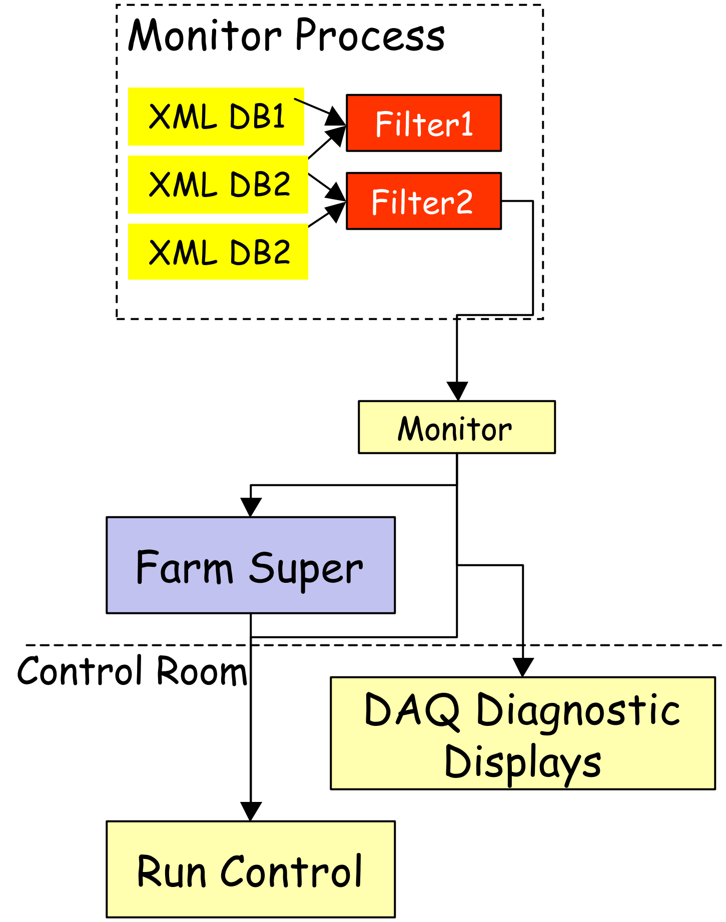
Gordon Watts
gwatts@u.washington.edu

University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000

Putting It Together

- No reason for a display to be the only consumer for Monitor Information
- Intelligent process can watch for known patterns
  - Good
    - Look for correct running patterns, warning when they aren't present
  - Bad:
    - Look for known bad patters, warning when they are present, perhaps taking automatic action, even.
    - Rule based system
      - Difficult to maintain.
      - Would like a snap-shot feature



Control of large scale distributed DAQ/trigger systems in the networked PC era

Gordon Watts
gwatts@u.washington.edu

University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000

Conclusions

- PC Based DAQ/Trigger
  - Large component count
  - Uptime must be almost 100% for each component
  - Software bugs are a reality
  - Approach system design in a unified way
- A General Approach Means
  - Flexible system
  - Ease of extension
    - Hooks for adding monitor and other capabilities, for example.
    - Hooks to deal with many error situations one can expect to encounter over 10 years of running.
  - Failure Mode Analysis
- This Requires
  - Robust, extensible Monitoring
  - Extensible Control System
    - Can handle addition of error conditions at a later time
  - Software infrastructure to tie the two together
  - Being aware of shifters!

Gordon Watts
gwatts@u.washington.edu
University of Washington, Seattle

DAQ2000
NSS2000
Oct 20, 2000