Antonio Ceseracciu

(for the BaBar computing group)

SLAC / INFN Padova

The new BaBar data reconstruction control system

CHEP 2003, San Diego

What BaBar Data Reconstruction is

- Prompt Reconstruction (PR)
  - performs Calibration and Event Reconstruction from raw data coming from the detector
  - consists of a set of tools i.e. executable programs
  - executable programs contain “physics” code
- PR Control System
  - manages production of Calibration and Event data
  - runs the PR tools as black boxes
  - organizes parallel processing in computing farms
  - monitors the processing
  - deals with common failures
  - provides a high level user interface to the system

Why, motivation

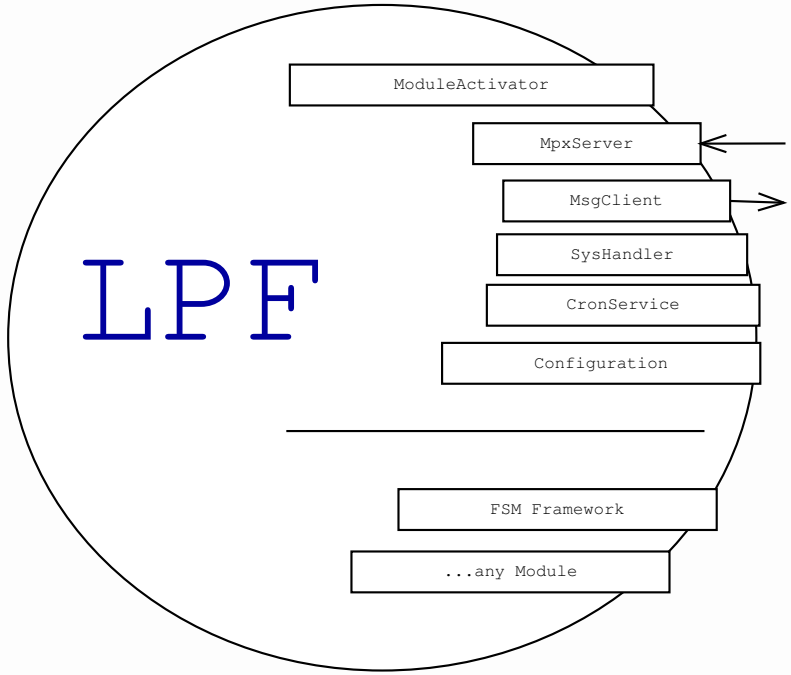
- new Processing Model
  - two passes: Prompt Calibration PC / Event Reconstruction ER
  - details in Peter Elmer's talk about BaBar Data Reco
- increased luminosity
  - more processing farms: 3 PC, $\sim$ 32 CPUs; 9 ER, $\sim$ 64 CPUs
  - scaling issues, administration burden
- need for flexibility
  - old Control System became unmaintainable
  - multiple tasks and sites (SLAC, Padova)
  - quickly adapt to evolving computing environment, tools, model
- need to minimize the human time spent in assisting the processing

How

- Software Design features:
  - Modular design based on a common framework
  - Isolation between Infrastructure and Processing layers
  - Flexible infrastructure
  - Proxy / Nameservice abstraction over physical topology
  - OO technology coupled with dynamically typed language
- Specific PR features:
  - Multi-agent distributed model
  - Active Monitoring of processes
  - Hierarchical, XML-based configuration system
  - Abstract modeling of computation (Finite State Machine)

How, infrastructure layer

- Light Processing Framework (LPF)
  - Light: delegates any task to core services
  - Provides message passing engine
  - Cooperative multitasking (single threaded)
  - Dynamic module loading
  - guarantees: modularity, flexibility
  - every agent is an LPF.

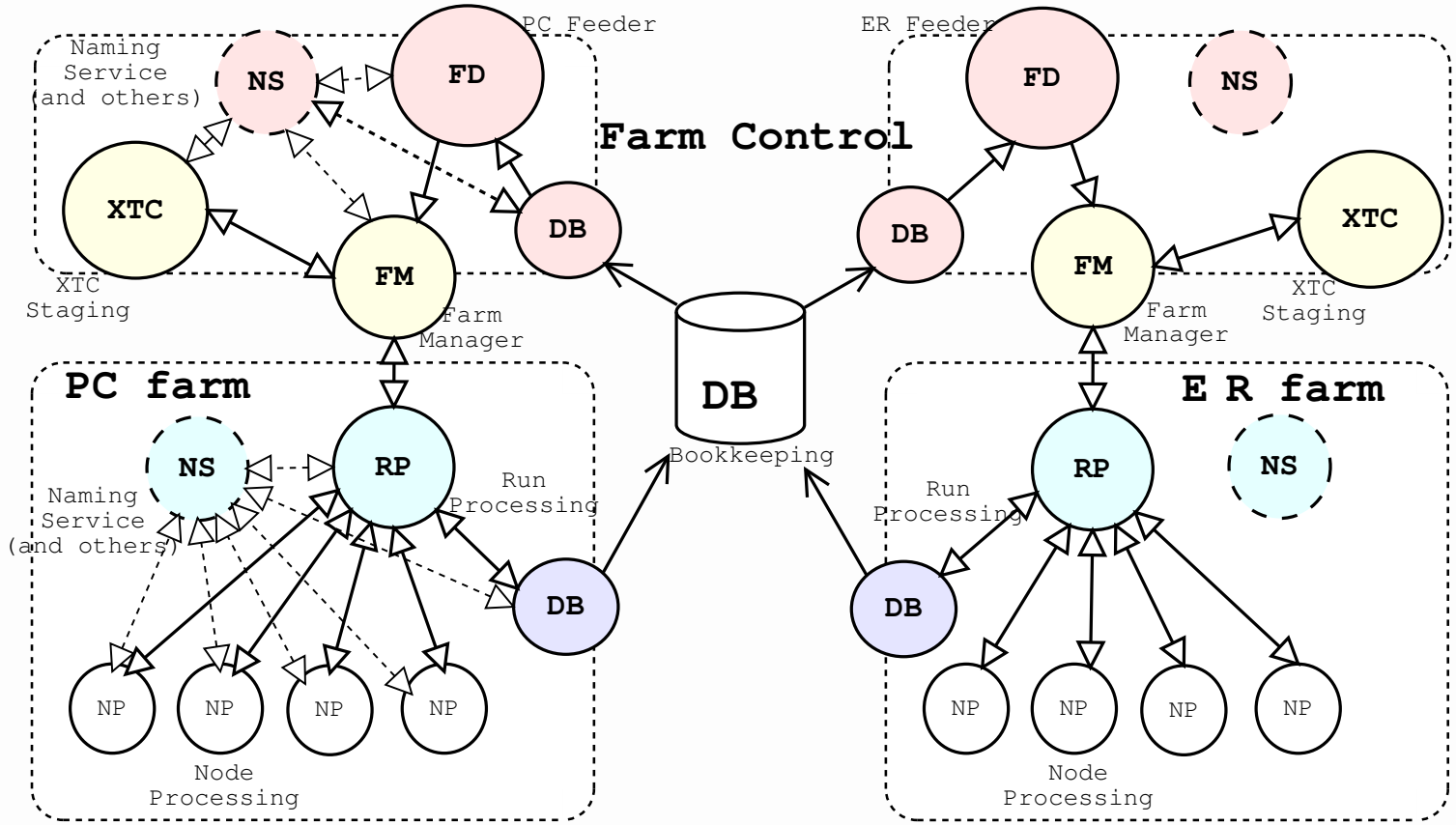


How, modules

- The Services abstraction
  - Any message is sent to a Service.
  - A service can be a local or a remote module.
  - Location and forwarding to a remote module is transparently handled by the framework.
- Core Services
  - Message Passing frontend (IMC)
  - Tcp-Ip (only!) intercommunication between LPFs
  - Dynamic allocation of modules
  - Transparent proxy system
  - Fault-tolerant naming service
  - Active handler for external processes

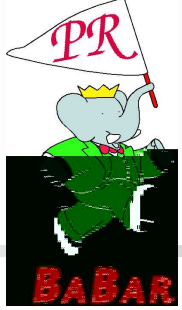
How, agents hierarchy

- Specialized agents belong to different hierarchy levels.
- The current hierarchy is a two-tier with a broker layer.



How, configuration system

- Static configuration
  - fully expressed by a set of XML files
  - includes all the needed information to automatically activate and configure a farm from scratch
  - environment dependent settings are defined as part of the static configuration. No hardcoded paths!
- Dynamic configuration
  - runtime information is handled by dedicated “environment” objects
  - processing code does not have to know the origin of runtime parameters, coming from hardcoded defaults, static configuration, database, runtime.



How, Finite State Machines

- Finite State Machine: a powerful abstraction framework
  - an FSM is described by an high level, special purpose language
  - processing is split in sequences of steps, or *states*
  - a state is an object which implements the state interface
  - each state returns a status *label*
  - the FSM definition, connects each state's jump labels to another state, thus defining the computation graph aka processing model
  - for the PR computing model, this graph looks like a chain.
  - it is straightforward to modify the model by adding or removing states from some FSM. No need to tweak the underlying code.
- the FSM enforces separation between policy and mechanism.

How, processing, for real

Different duties, different Finite State Machines

- Node Processing FSM: one CPU, one FSM
  - handles any activity local to a processing node
  - runs and monitors the main reco executable, “Elf”
  - local postprocessing: sanity checking, statistics collection from logfiles
- Run Processing: one Farm, one FSM
  - by far the most complex FSM with over 30 states
  - describes the processing model
  - activates and monitors Node Processing FSMs

When, timescale

- The project of the new control system started on 01/2002.
- Along with development of the core, the processing starts to be coded in a separate package: 08/2002.
- The system started to be used in production in 11/2002.
- As planned, it processed since the beginning the new acquisition run of BaBar, Run3 (04/11/2002).
- The bulk of the development effort is concluded by 03/2003.
- The system is ported and tested in the INFN computing facility in Padova: 03/2003.
- Most of the processing logic of this control system comes from the old one. This helped to speed up development, but no code was reused.

Conclusions, performance

- The Control System is not a CPU- or IO- intensive application.
- Machine-performance rating
  - depends mainly on the processing duty cycle, which mostly depends on reliability. The processing time spent for control system failure is a negligible fraction of the total.
- Human-performance rating
  - measures the time spent by operation managers in assisting the computing farms. This figure is still too high.
  - at SLAC 2.5 FTEs are devolved to assisting 3 PC and 5 ER farms; Padova is devolving 2 to assist 4 ER farms.
- Overall, performance rating for the control system is good.

Conclusions, design

- The overall software design of the system proved consistent.
  - Most basic interfaces remained very stable while application layer was built;
  - The active agents model, while more complex than passive ones, was made manageable by the modular framework;
  - Modularity allowed the control system to adapt rapidly to ever evolving needs;
- The Control System design is a successful one.
- Control System infrastructure will be used for different production applications in BaBar.

Who

- Main developers
  - Antonio Ceseracciu
  - Martino Piemontese
  - Francesco Safai Tehrani
- Co-developers and testers
  - Peter Elmer
  - Doug Johnson
  - Richard John Parry
  - David Payne
  - Teela Marie Pulliam
- Helpers and Friends
  - Sridhara Dasu
  - Peter Elmer
  - Adil Hasan
  - Doug Johnson
  - Richard John Parry
  - Eugenio Paoloni
  - David Payne
  - Teela Marie Pulliam
  - Alessio Sarti
  - ... to name a few. Thanks!