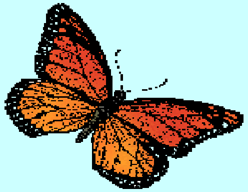
MONARC

Models Of Networked Analysis at Regional Centres

Distributed System Simulation

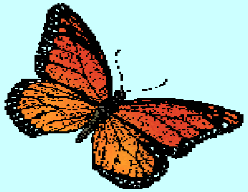
Iosif C. Legrand (CERN/CIT)

LCB, June 22, 1999



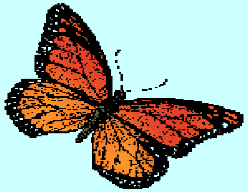
The GOALS of the Simulation Program

- To perform realistic simulation and modelling of the distributed computing systems, customised for specific HEP applications.
- To reliably model the behaviour of the computing facilities and networks, using specific applications software (OODB model) and the usage patterns.
- To offer a dynamic and flexible simulation environment.
- To provide a design framework to evaluate the performance of a range of possible computer systems, as measured by their ability to provide the physicists with the requested data in the required time, and to optimise the cost.
- To narrow down a region in this parameter space in which viable models can be chosen by any of the LHC-era experiments.



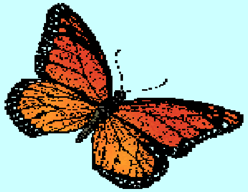
Design Considerations of the Simulation Program

- ➨ **The simulation & modelling task for the MONARC project requires to describe complex programs running in a distributed architecture.**
- ✪ ***Selecting Tools which allow one easily to map the logical model into the simulation environment.***
- ➨ **A process oriented approach for discrete event simulation is well suited to describe concurrent running programs.**
  - ❋ **"Active objects" (having an execution thread, a program counter, stack...) allow an easy way to map the structure of a set of distributed running programs into the simulation environment.**



Design Considerations of the Simulation Program (2)

- This simulation project is based on Java(TM) technology which provides adequate tools for developing a flexible and distributed process oriented simulation. Java has build-in multi-thread support for concurrent processing, which can be used for simulation purposes by providing a dedicated scheduling mechanism.
- The distributed objects support (through RMI or CORBA) can be used on distributed simulations, or for an environment in which parts of the system are simulated and interfaced through such a mechanism with other parts which actually are running the real application. The distributed object model can also provide the environment to be used for autonomous mobile agents.



Simulation Model

- It is necessary to abstract from the real system all components and their time dependent interaction.
- THE MODEL has to be equivalent to the simulated system in all important respects.

CATEGORIES OF SIMULATION MODELS

- Continuous time → usually solved by sets of differential equations
- Discrete time → Systems which are considered only at selected moments in time
- Continuous time + discrete event

Discrete event simulations (DES)

- EVENT ORIENTED
- PROCESS ORIENTED

Simulation Model(2)

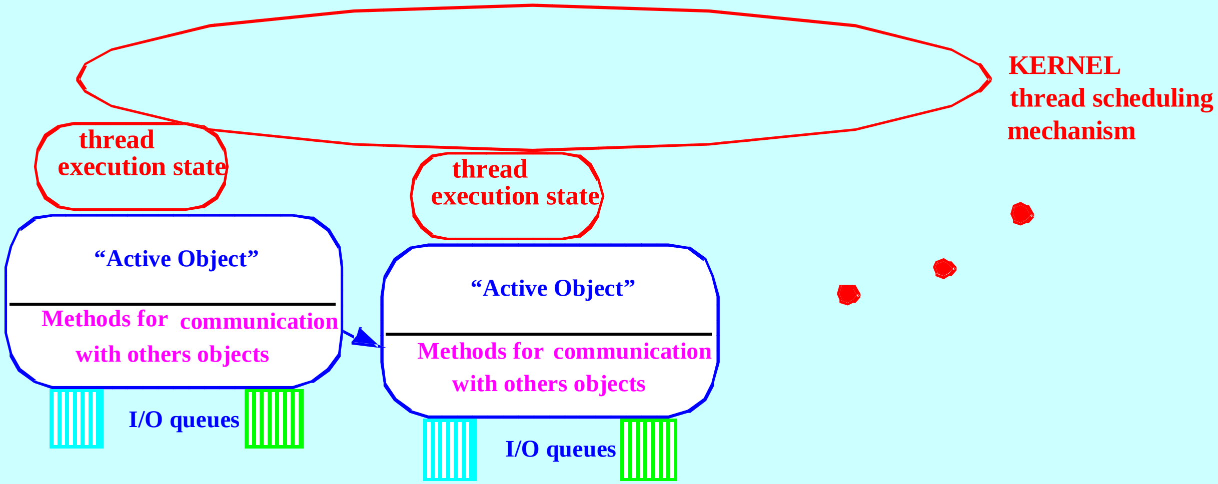
Process oriented DES Based on "ACTIVE OBJECTS"

Thread: execution of a piece of code that occurs independently of and possibly concurrently with another one

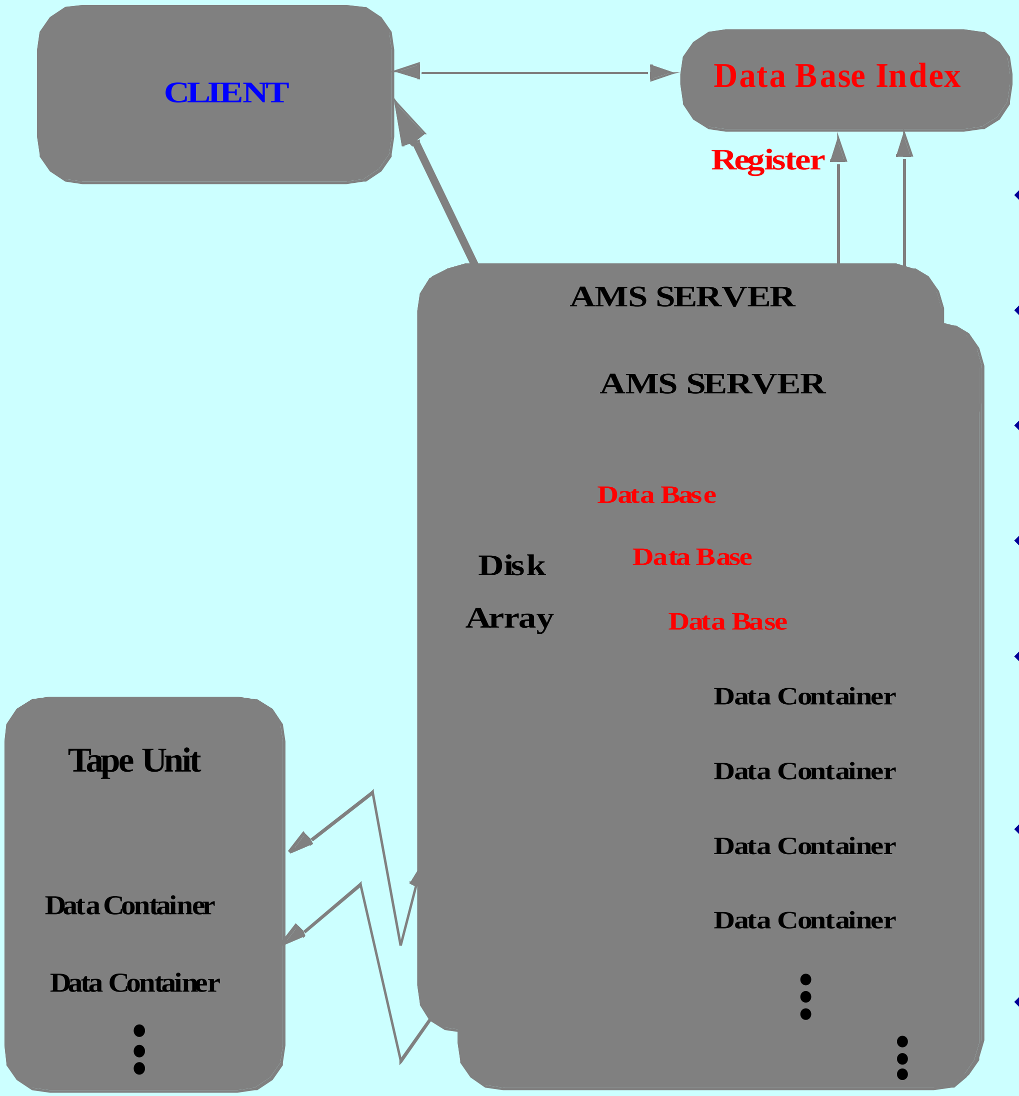
Execution state: set of state information needed to permit concurrent execution

Mutual exclusion: mechanism that allow an action to performed on an object without interruption

Asynchronous interaction: signals/ semaphores for interrupts



Data Model



It provides:

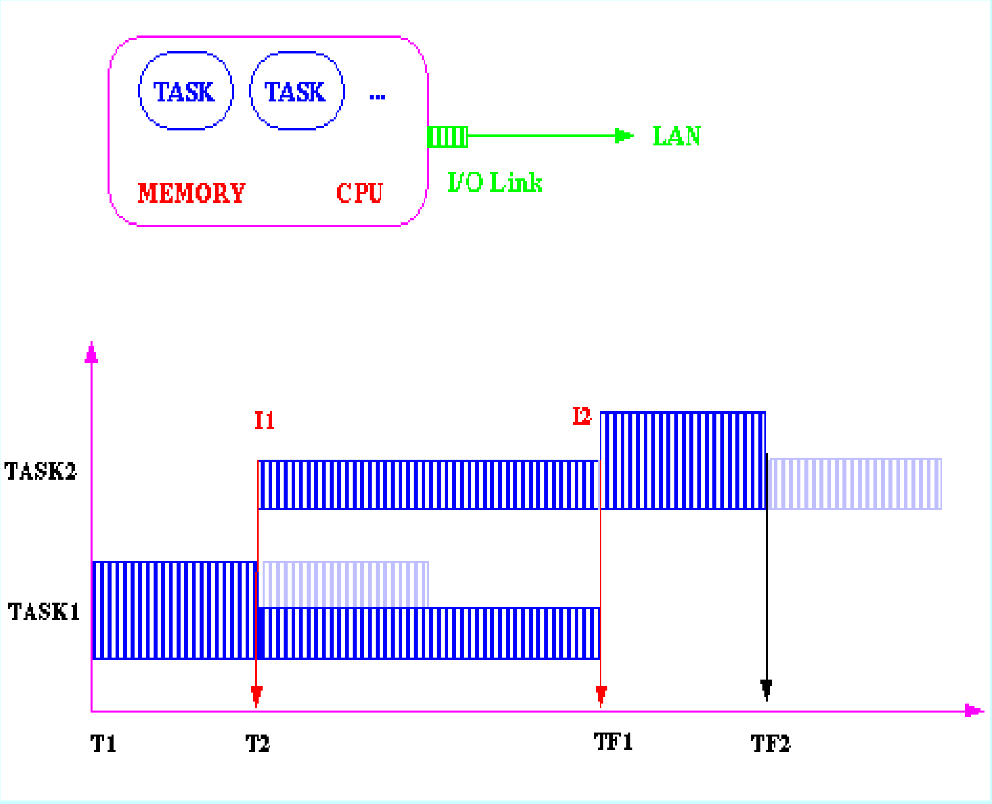
- Realistic mapping for an Object Database
- Specific HEP data structure
- Transparent access to any data
- Automatic storage management
- An efficient way to handle very large number of Objects.
- Emulate clustering factors for different types of access patterns.
- Handles related objects in different data bases.

Multitasking Processing Model

Concurrent running tasks share resources (CPU, memory, I/O)

"Interrupt" driven scheme:

For each new task or when one task is finished, an interrupt is generated and all "processing times" are recomputed.



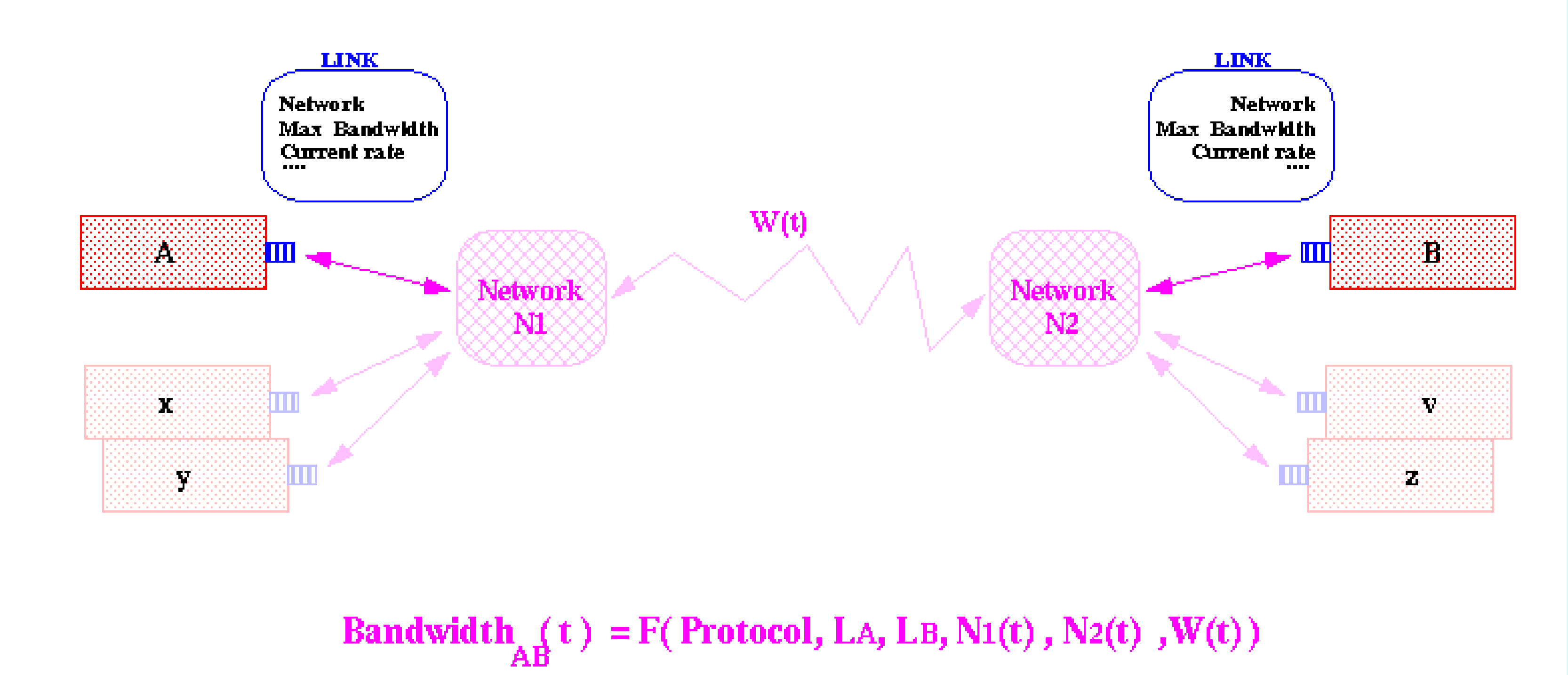
It provides:

An efficient mechanism to simulate multitask processing

An easy way to apply different load balancing schemes

LAN/WAN Simulation Model

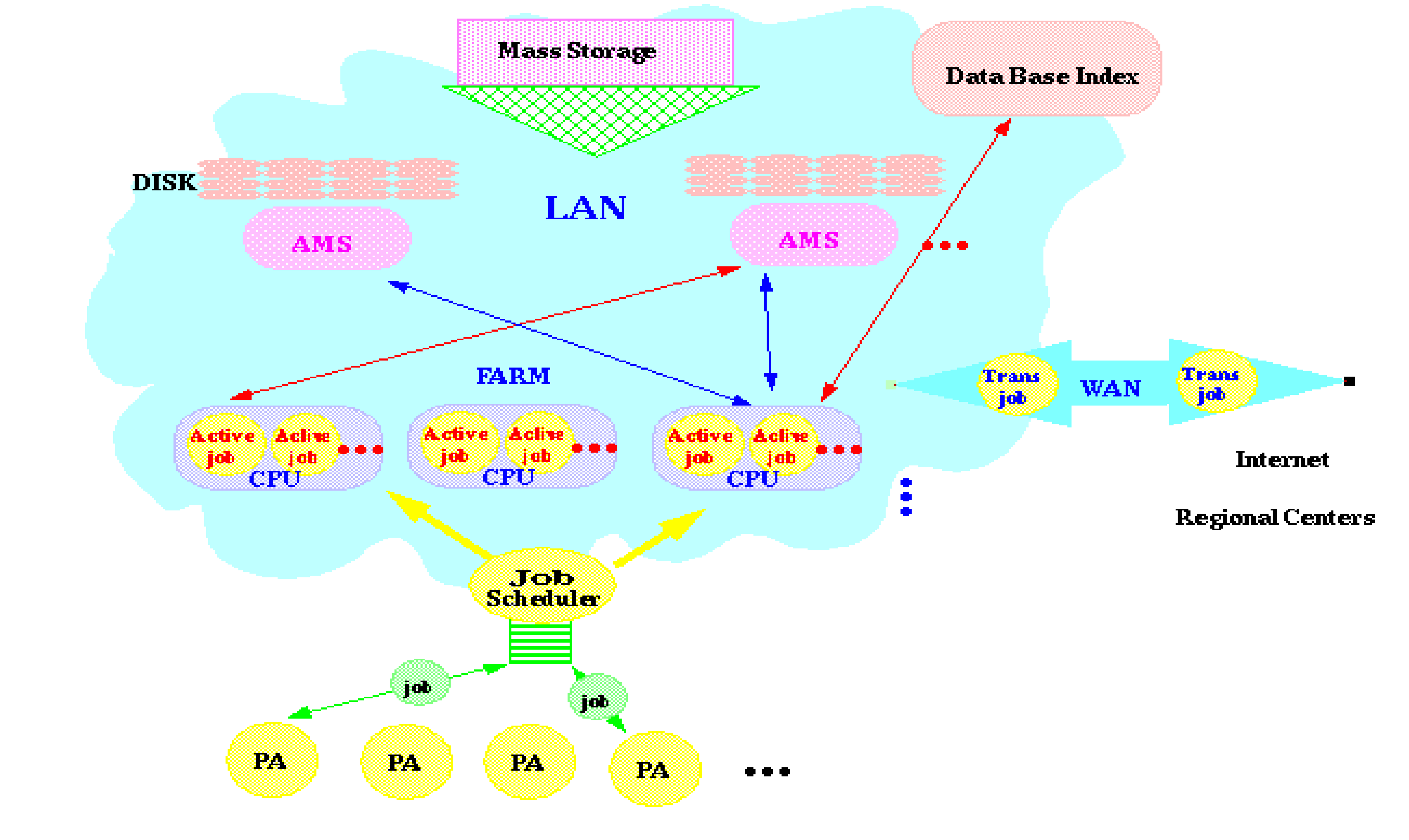
"Interrupt" driven simulation → for each new message an interrupt is created and for all the active transfers the speed and the estimated time to complete the transfer are recalculated.



An efficient & realistic way to simulate concurrent transfers having different sizes / protocols.

Regional Centre Model

Complex Composite Object

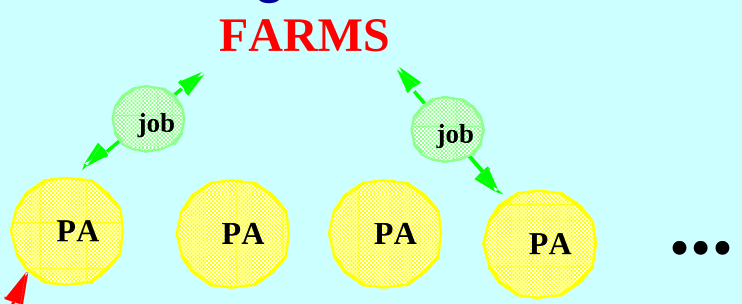


Arrival Patterns

A flexible mechanism to define the Stochastic process of data processing

Dynamic loading of "Activity" Tasks, which are threaded objects and controlled by the simulation scheduling mechanism

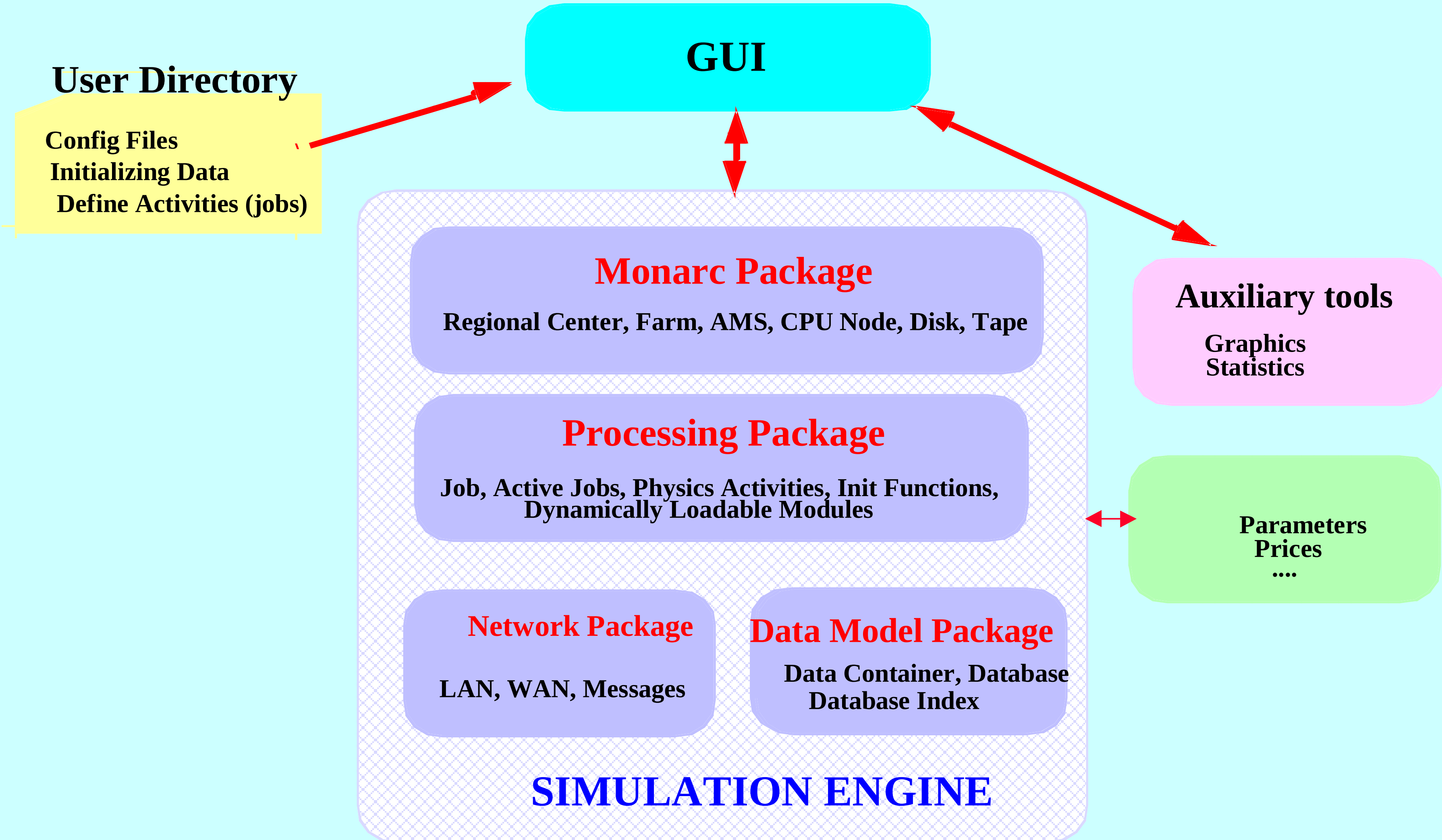
Physics Activities Generating Jobs



Each "Activity" thread generates data processing Jobs

```
for( int k =0; k< jobs_per_group; k++) {
       Job job = new Job( this, Job.ANALYSIS, "TAG"+rc_name, event_offset,  event_offset+events_to_process-1, null, null );
      job.setAnalyzeFactors( 0.01, 0.005 );
      farm.addJob( job);
      sim_hold(1000.0 ) ;
}
```

The Structure of the Simulation Program



Input Parameters for the Simulation Program

Correctly identify and describe the time response functions for all active components in the system. This should be done using realistic measurements.

The simulation frame allows one to introduce any time dependent response function for the interacting components.

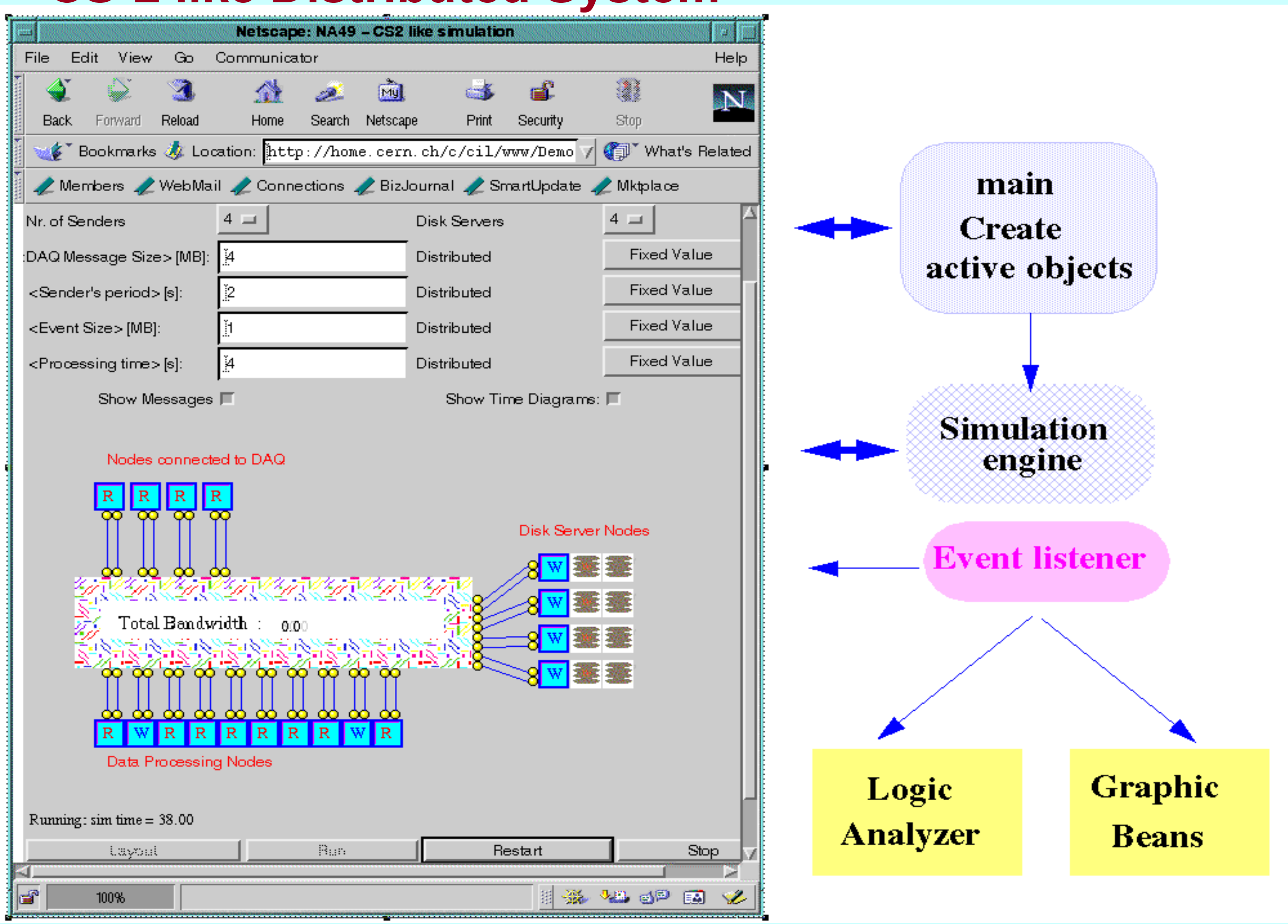
$$\delta(\mathbf{Ti}) = \mathbf{F}(\delta(\mathbf{Ti}-\mathbf{1}), \{\mathbf{SysP}\}, \{\mathbf{ReqP}\})$$

Response functions are based on "the previous state" of the component, a set of system related parameters (SysP) and parameters for a specific request (ReqP).

Allows to describe correctly Highly Nonlinear Processes or "Chaotic" Systems behavior (typical for caching, swapping…)

WEB GUI

CS-2 like Distributed System

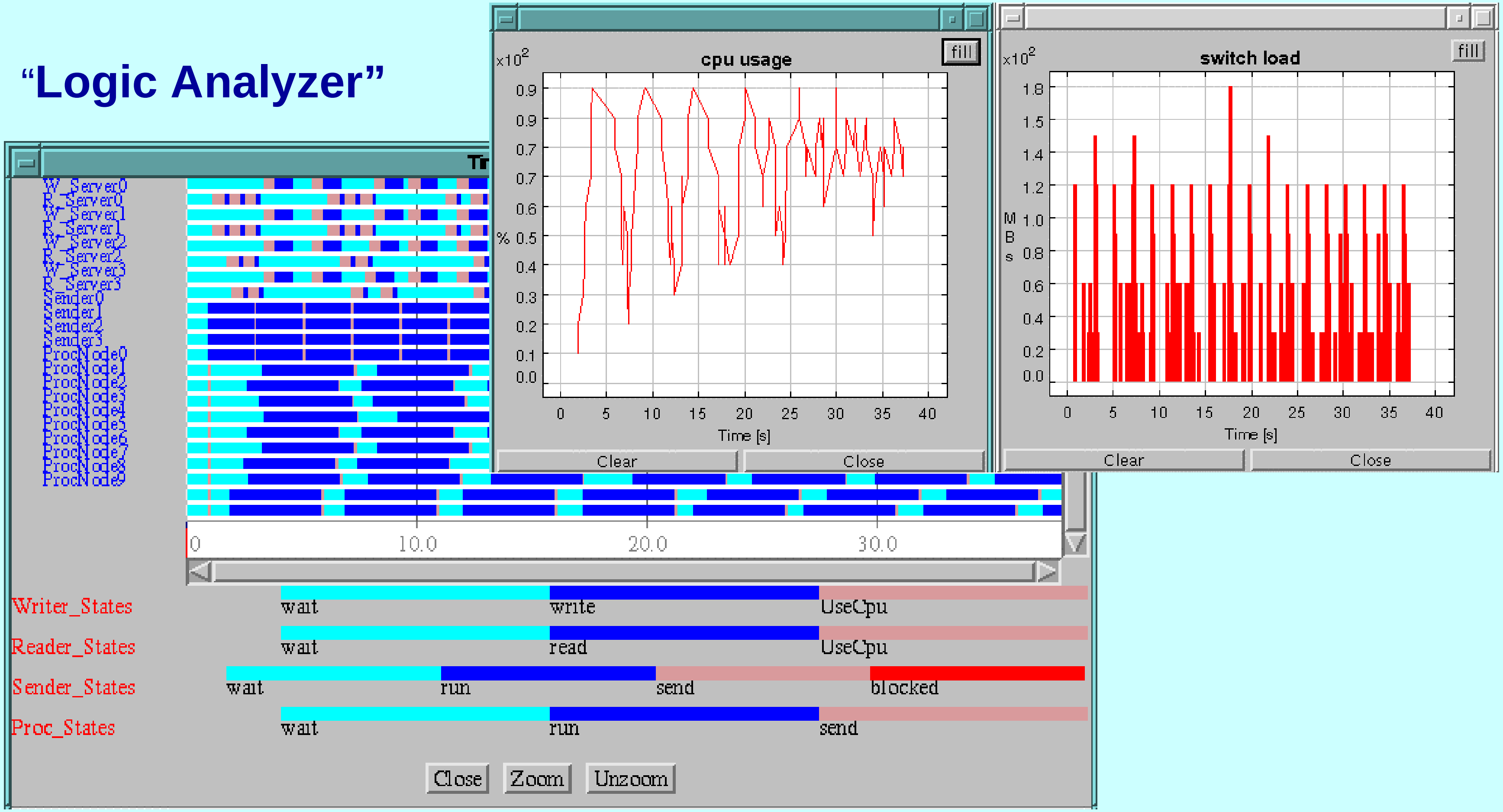


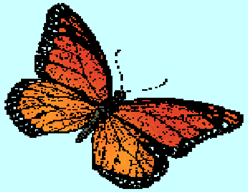
WEB GUI (2)

Graphic "Beans"

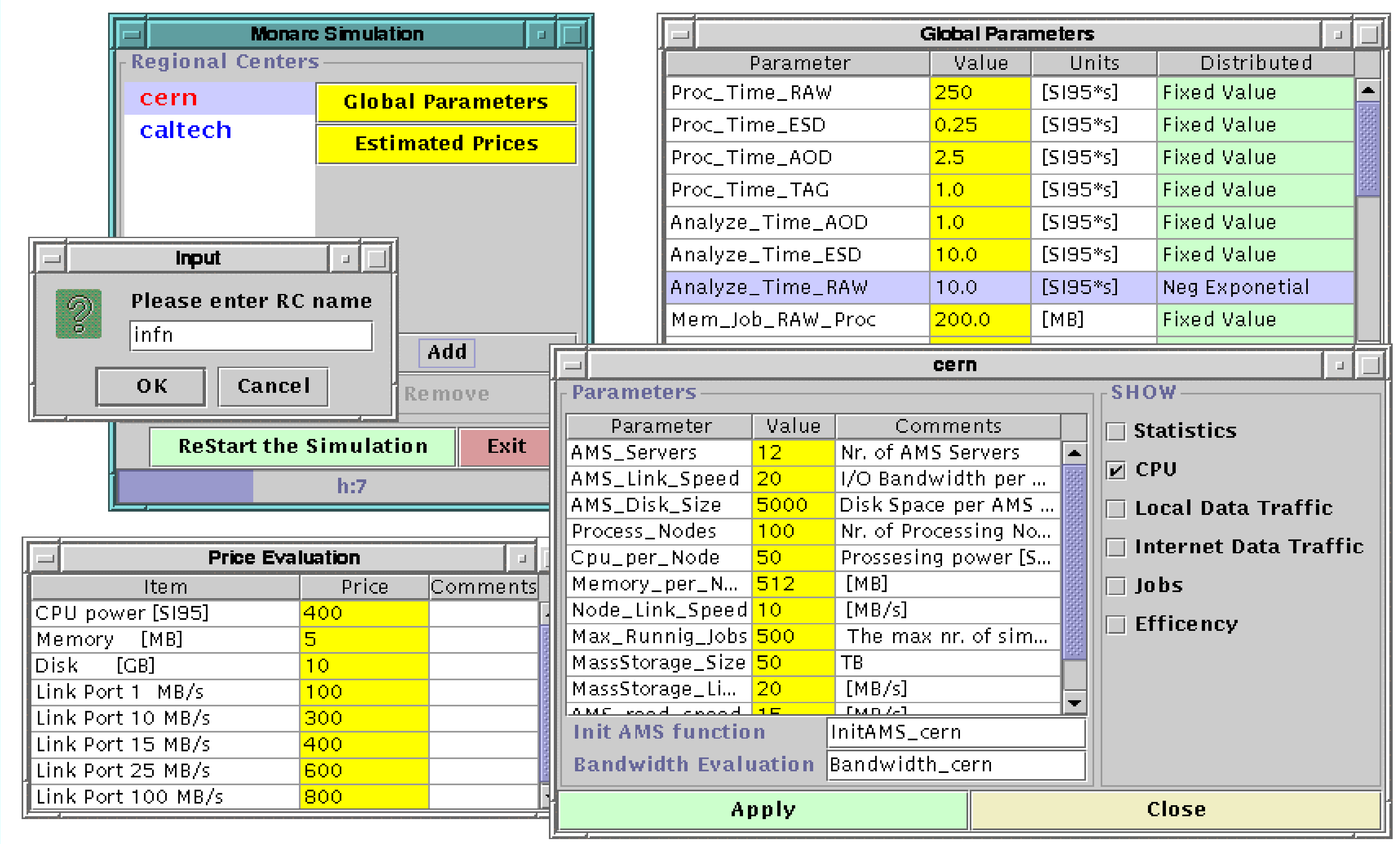
Monitoring different parameters

"Logic Analyzer"





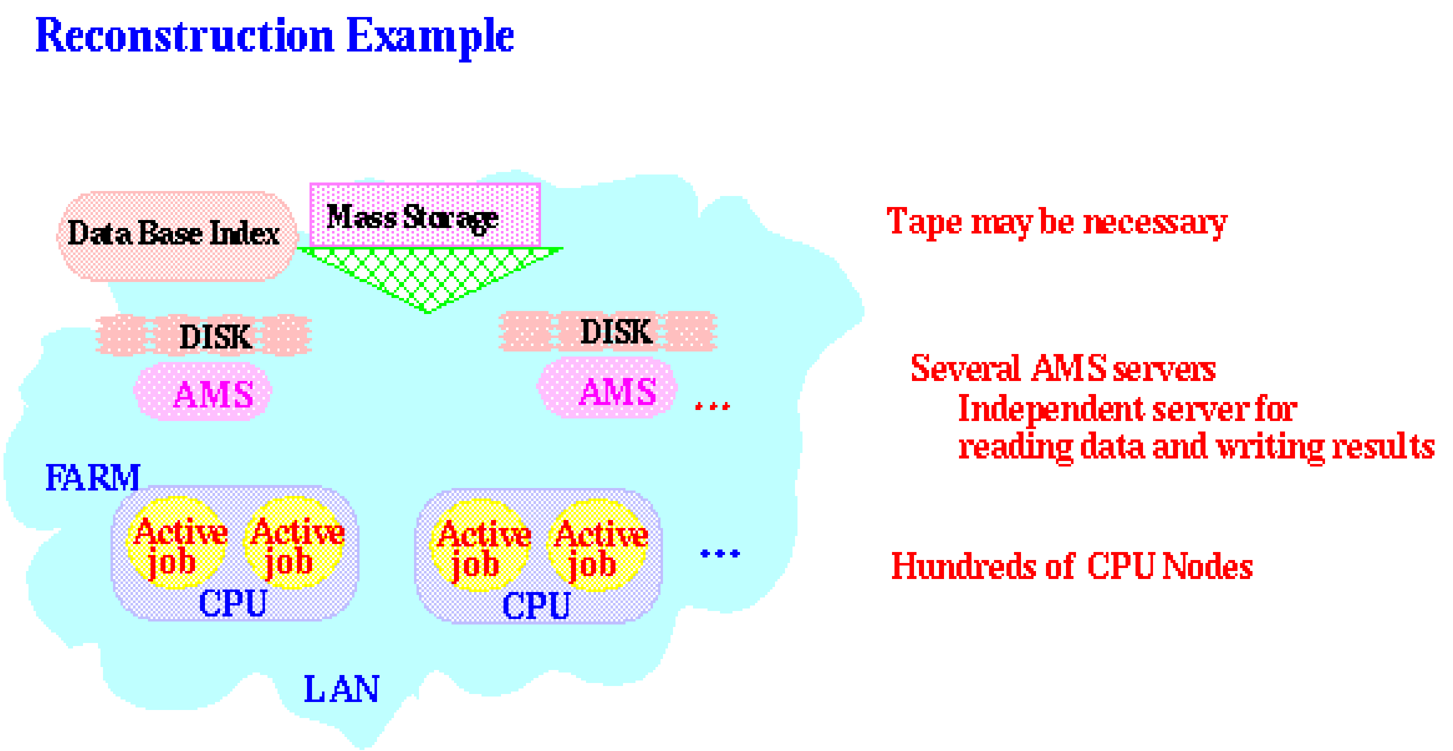
Simulation GUI



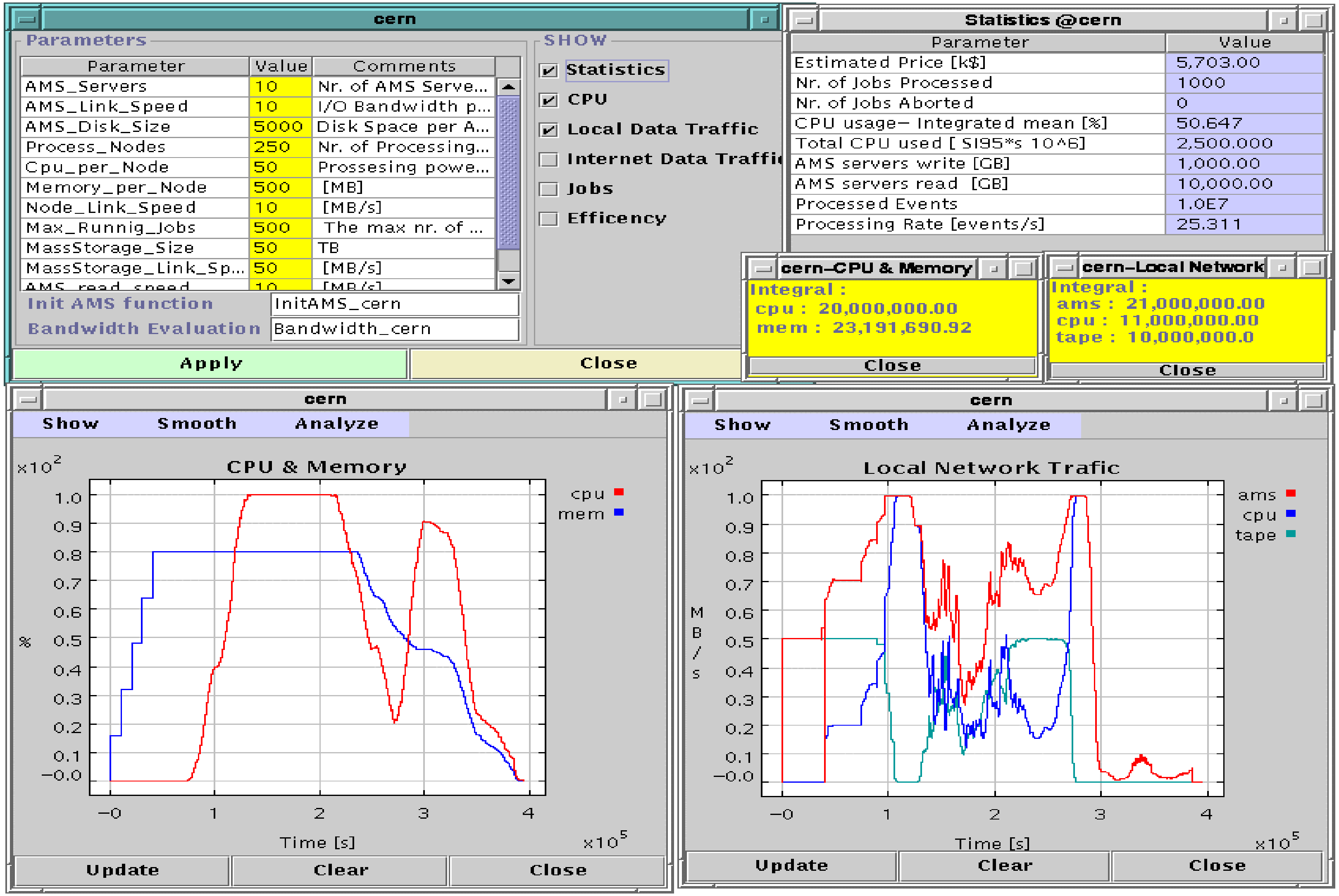
One may dynamically change parameters to control the simulation program

Example: Reconstruction

- It is performed in one Regional Centre
- Requires access to large amounts of data and a lot of processing power

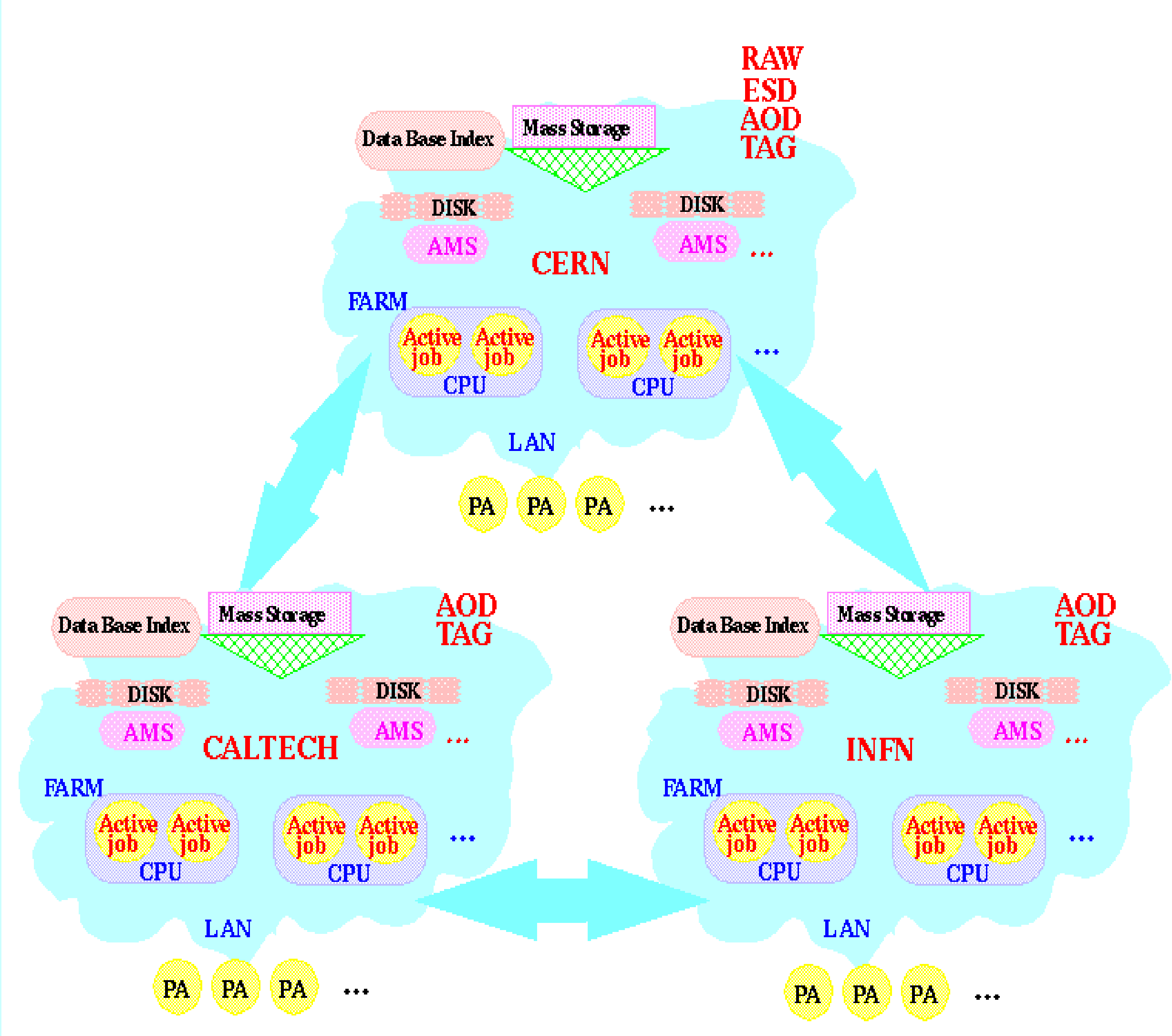


Example: Reconstruction (2)

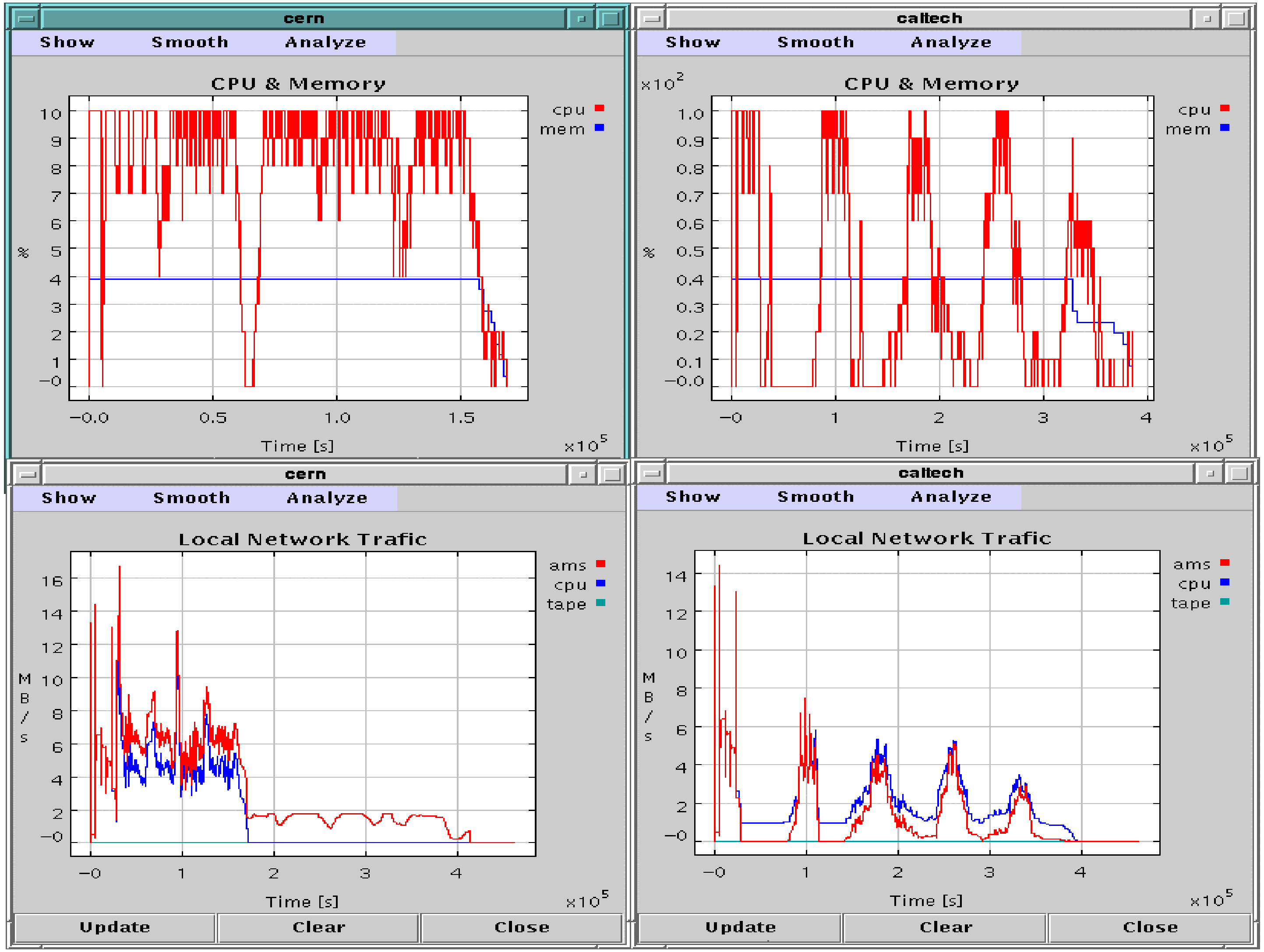


Example : Physics Analysis

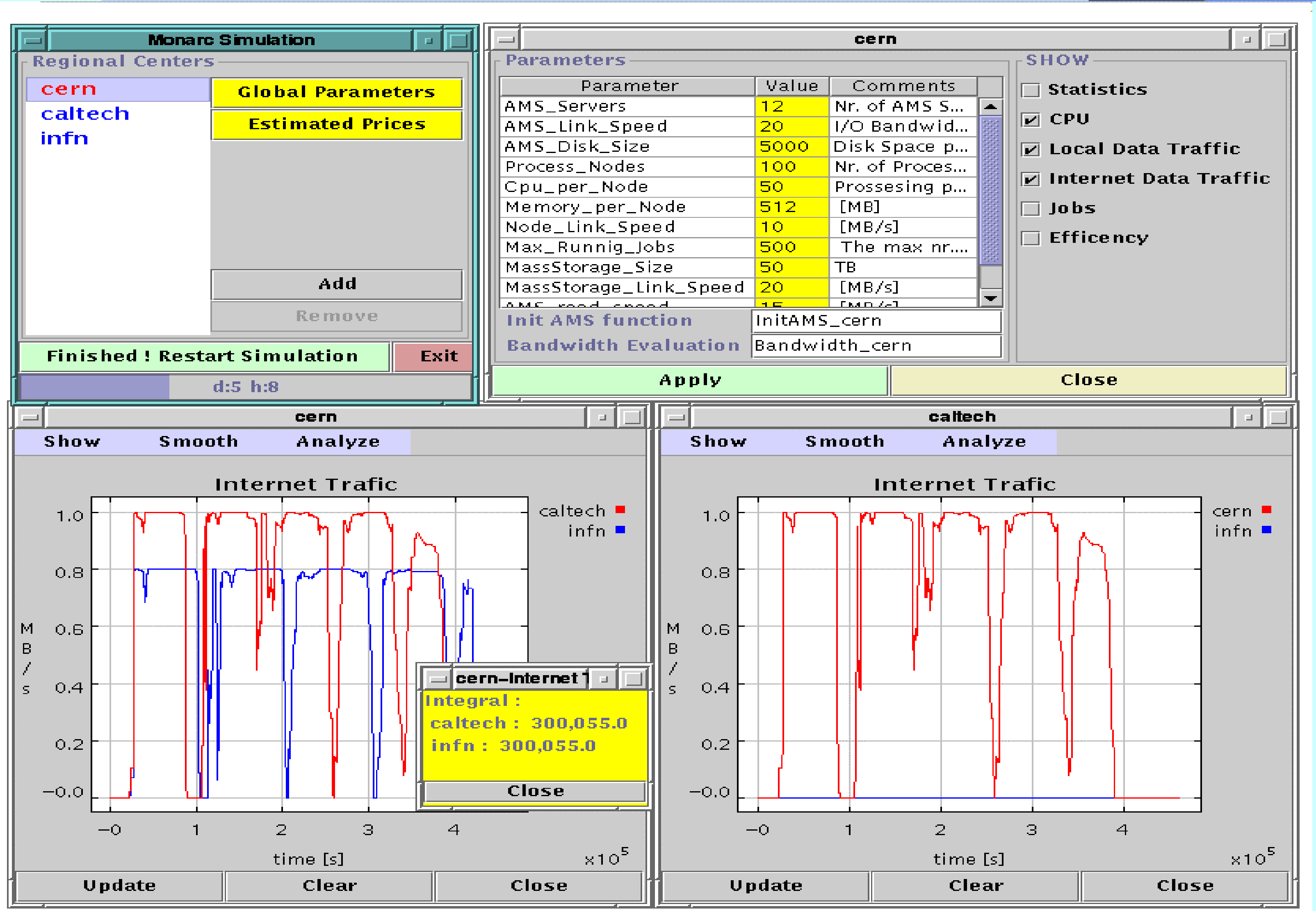
- ➨Similar data processing jobs are performed in several RCs
- ➨Each Centre has “TAG” and “AOD” databases replicated.
- ➨Main Centre provides “ESD” and “RAW” data
- ➨Each job processes AOD data, and also a a fraction of ESD and RAW



Example: Physics Analysis (2)



Example: Physics Analysis (3)



Summary

A CPU- and code-efficient approach for the simulation of distributed systems has been developed for MONARC

- ➨ provides an easy way to map the distributed data processing, transport, and analysis tasks onto the simulation
- ➨ can handle dynamically any Model configuration, including very elaborate ones with hundreds of interacting complex Objects
- ➨ can run on real distributed computer systems, and may interact with real components

❋ The Java (JDK 1.2) environment is well-suited for developing a flexible and distributed process oriented simulation.

❋ **This Simulation program is still under development, and dedicated measurements to evaluate realistic parameters and "validate" the simulation program are in progress.**