MONARC

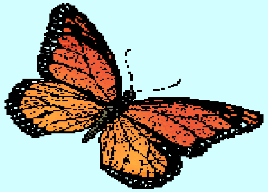
Models Of Networked Analysis at Regional Centers

Multi-threaded, discrete event simulation of distributed computing systems

Iosif C. Legrand (CIT / CERN)

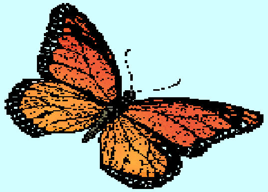
Contents

- Design and Development of a Simulation program for large scale distributed computing systems.
- Performance measurements based on an Object Oriented data model for specific HEP applications.
- Measurements vs. Simulation.
- An Example in using the simulation program:
  - Distributed Physics Analysis



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 application 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.
- To understand the performance and the limitations for the major software components intended to be used in LHC computing.

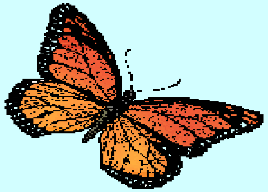


Design Considerations of the Simulation Program

- **The simulation and modelling task for the MONARC project requires to describe complex programs running in a distributed architecture.**
- ***Selecting tools which allow the easy to mapping of 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...) provide 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 built-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 all components and their time dependent interaction from the real system.
- 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)

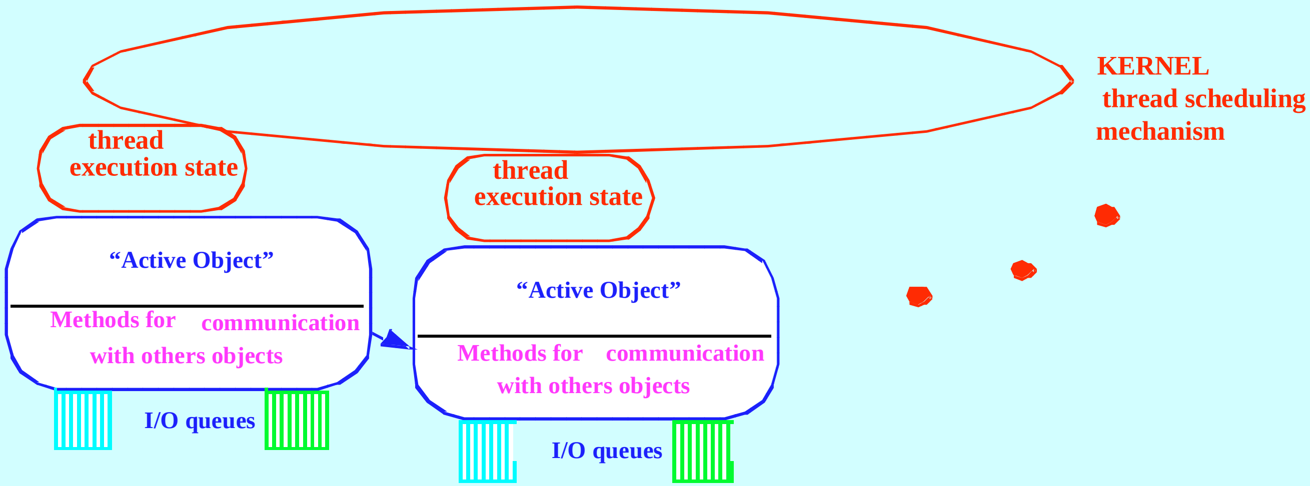
Process oriented DES Based on "ACTIVE OBJECTS"

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

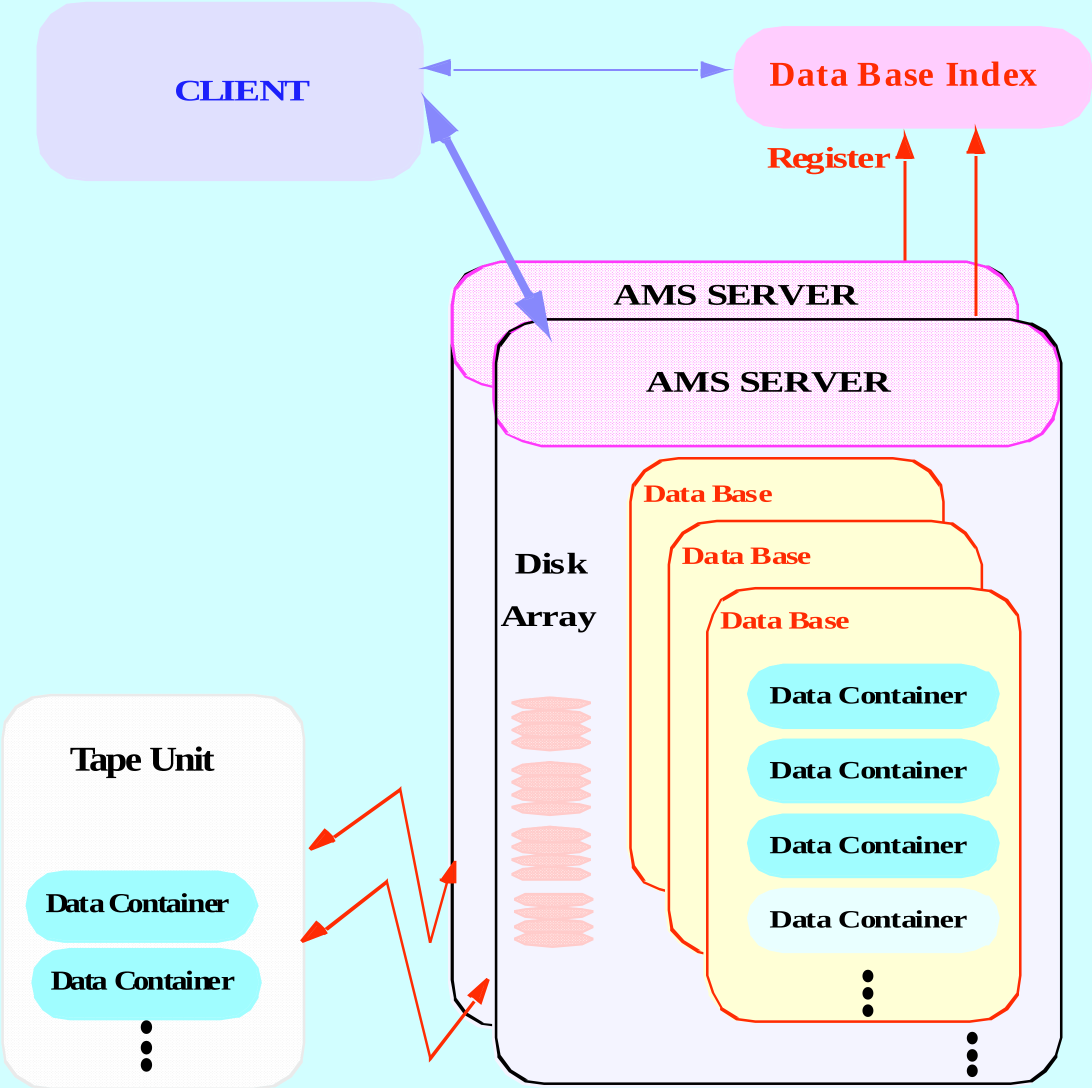
Execution state: set of state information needed to allow concurrent execution

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

Asynchronous interaction: signals / semaphores for interrupts



Data Model



It provides:

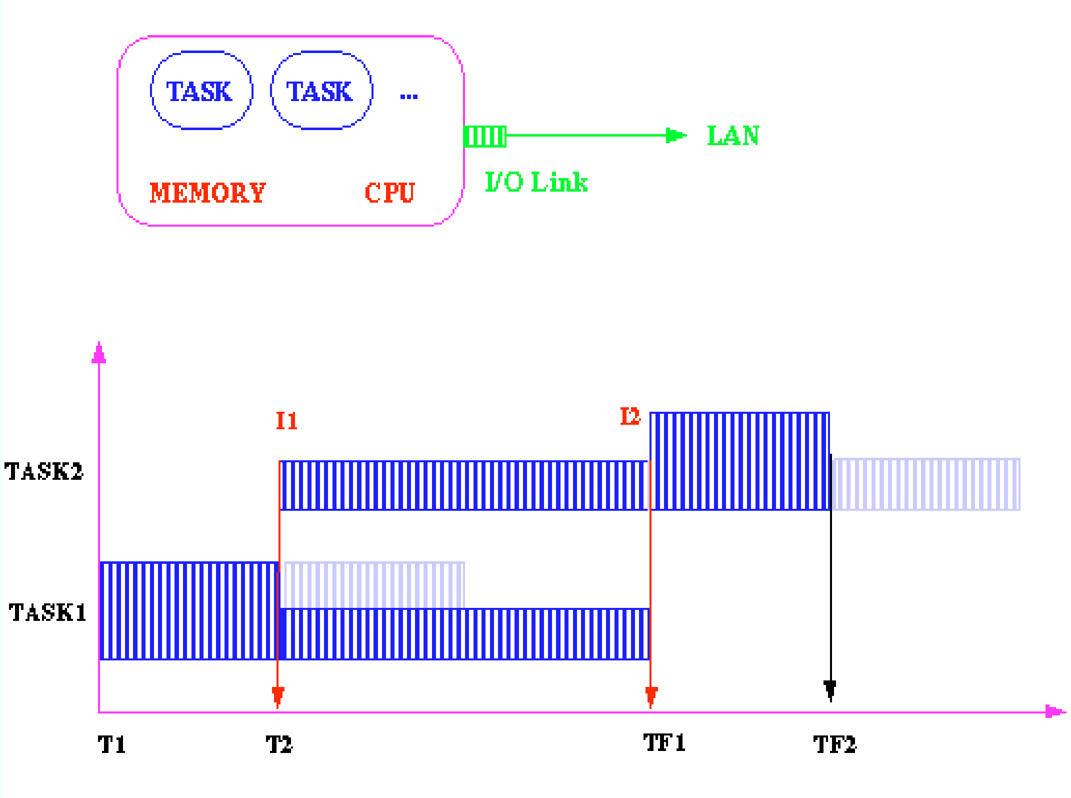
- Realistic mapping for an object data base
- Specific HEP data structure
- Transparent access to any data
- Automatic storage management
- An efficient way to handle very large number of objects.
- Emulation of clustering factors for different types of access patterns.
- Handling related objects in 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:

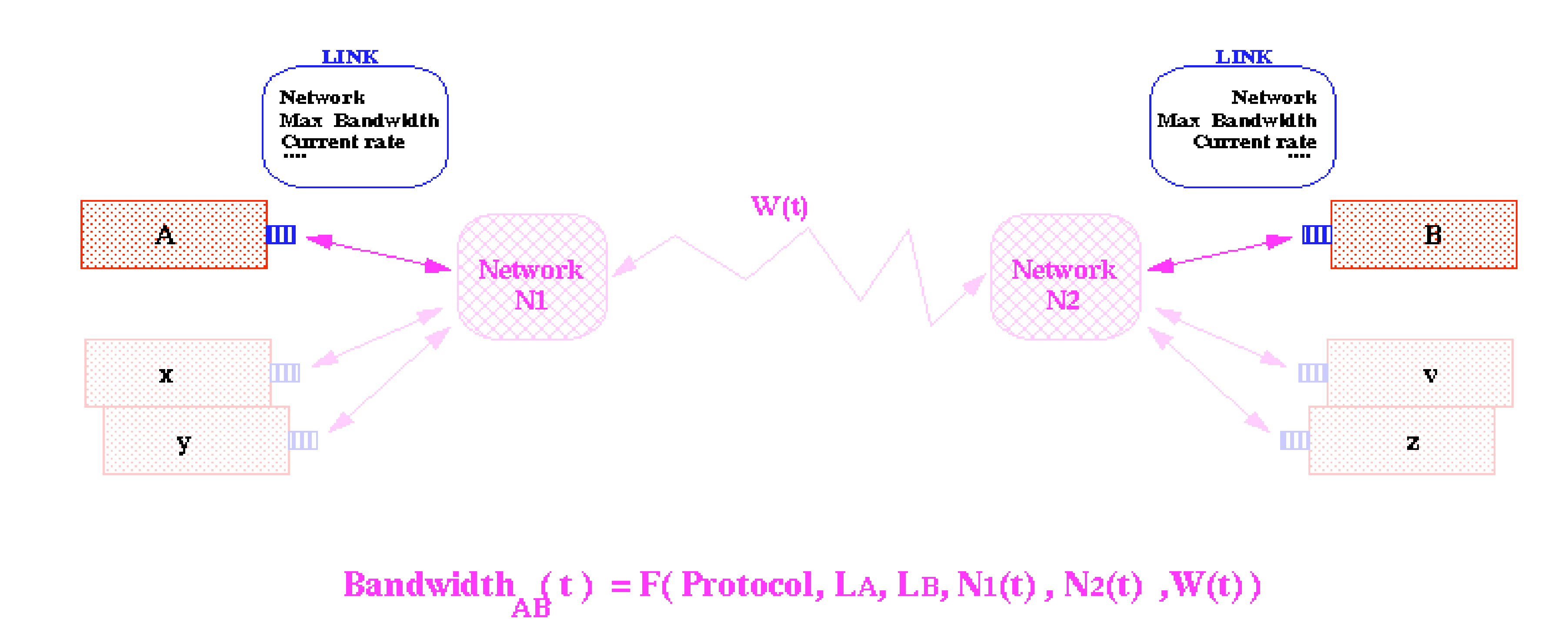
An efficient mechanism to simulate multitask processing.

Handling of concurrent jobs with different priorities.

An easy way to apply different load balancing schemes.

LAN/WAN Simulation Model

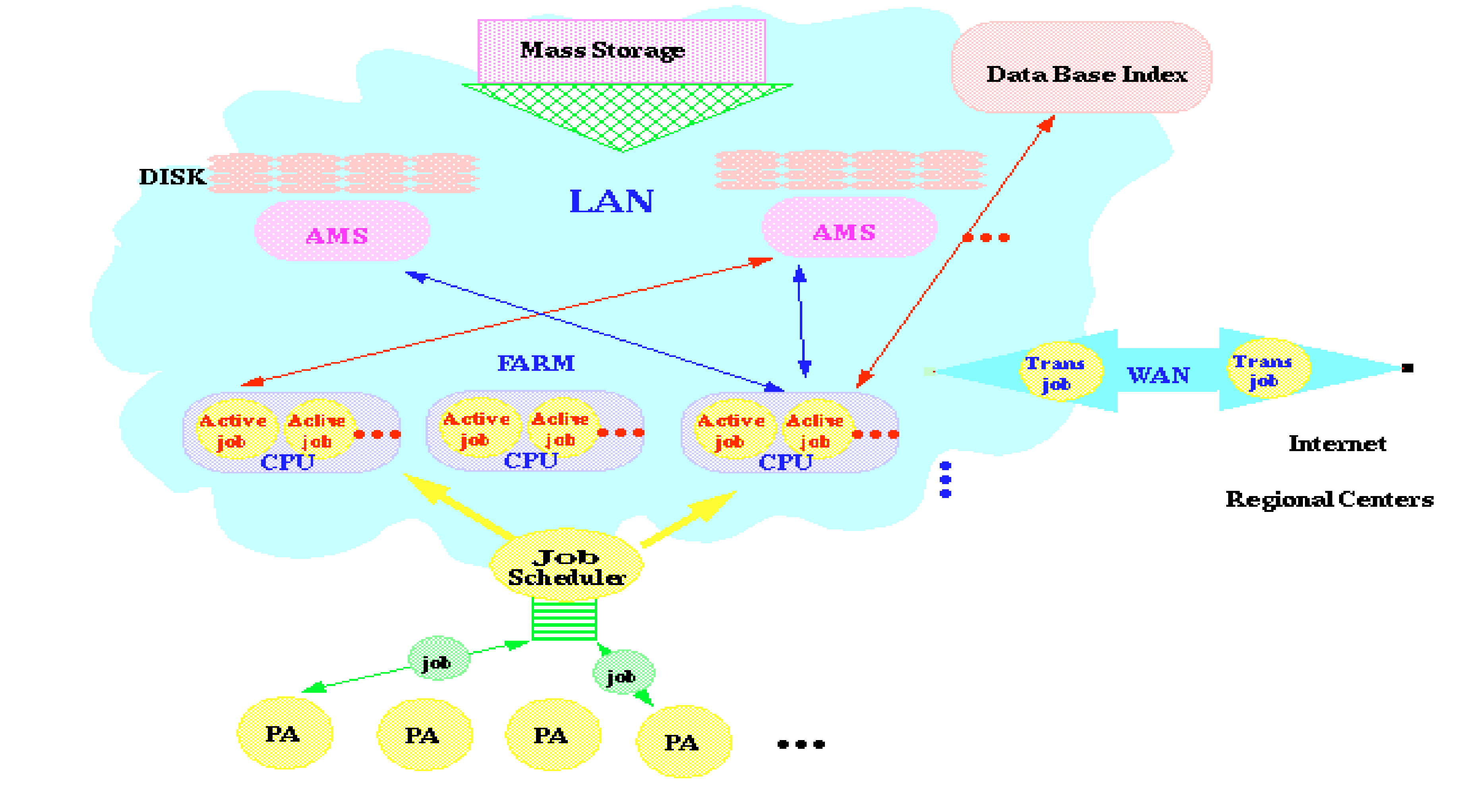
"Interrupt" driven simulation → for each new message an interrupt is created and for all the active transfers the speed and the estimated time to complete the transfer are recalculated.



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

Regional Centre Model

Complex Composite Object

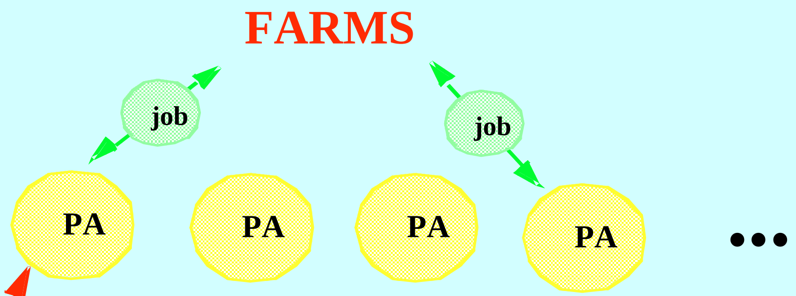


Arrival Patterns

A flexible mechanism to define the Stochastic process of data processing

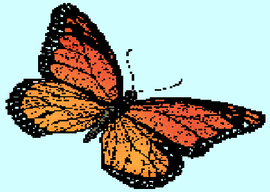
Dynamic loading of “Activity” tasks, which are threaded objects and are controlled by the simulation scheduling mechanism

Physics Activities Generating Jobs

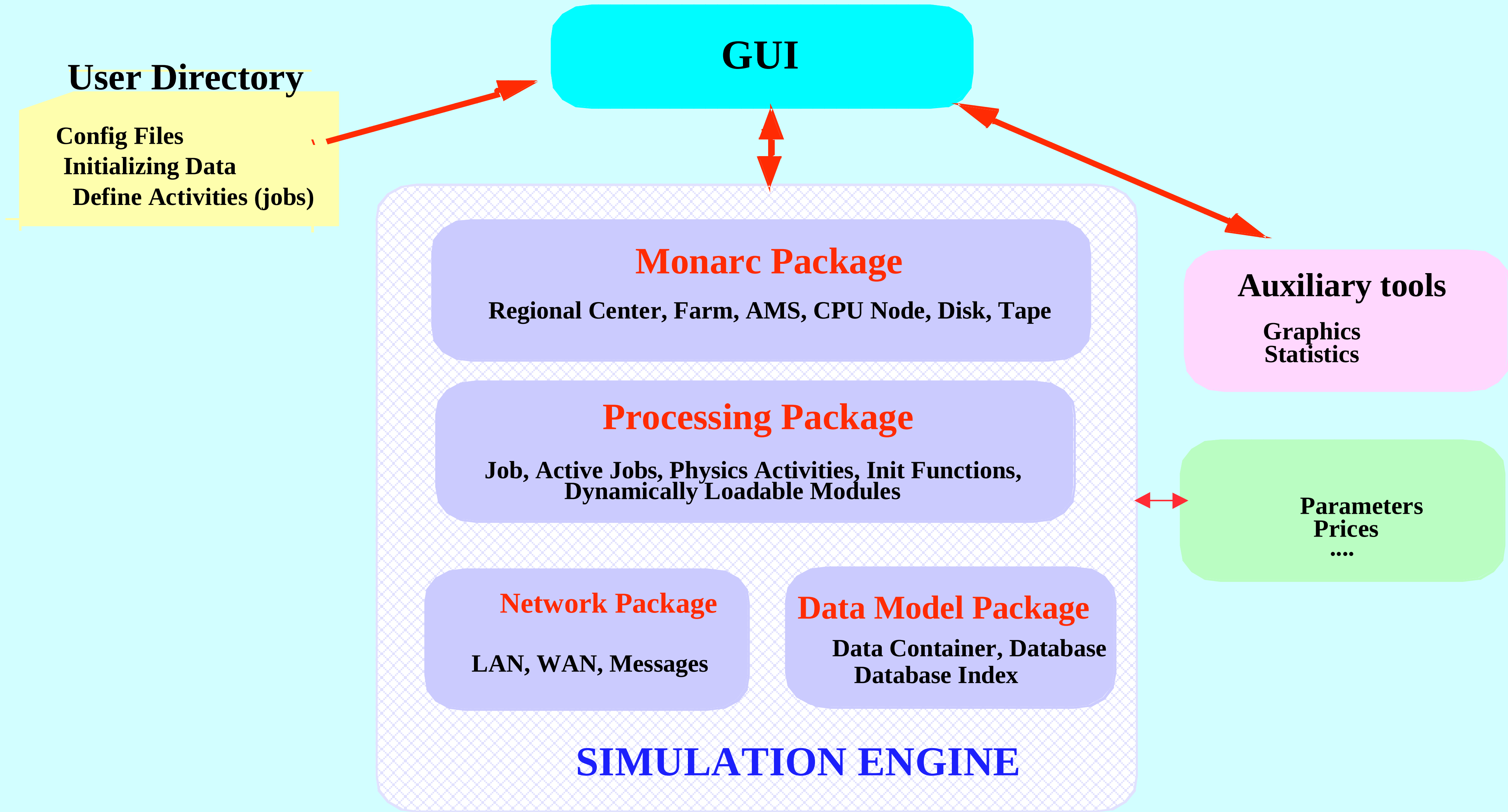


Each “Activity” thread generates data processing jobs

```
for( int k =0; k< jobs_per_group; k++) {
        Job job = new Job( this, Job.ANALYSIS, "TAG"+rc_name, 1, 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

It is important to 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.

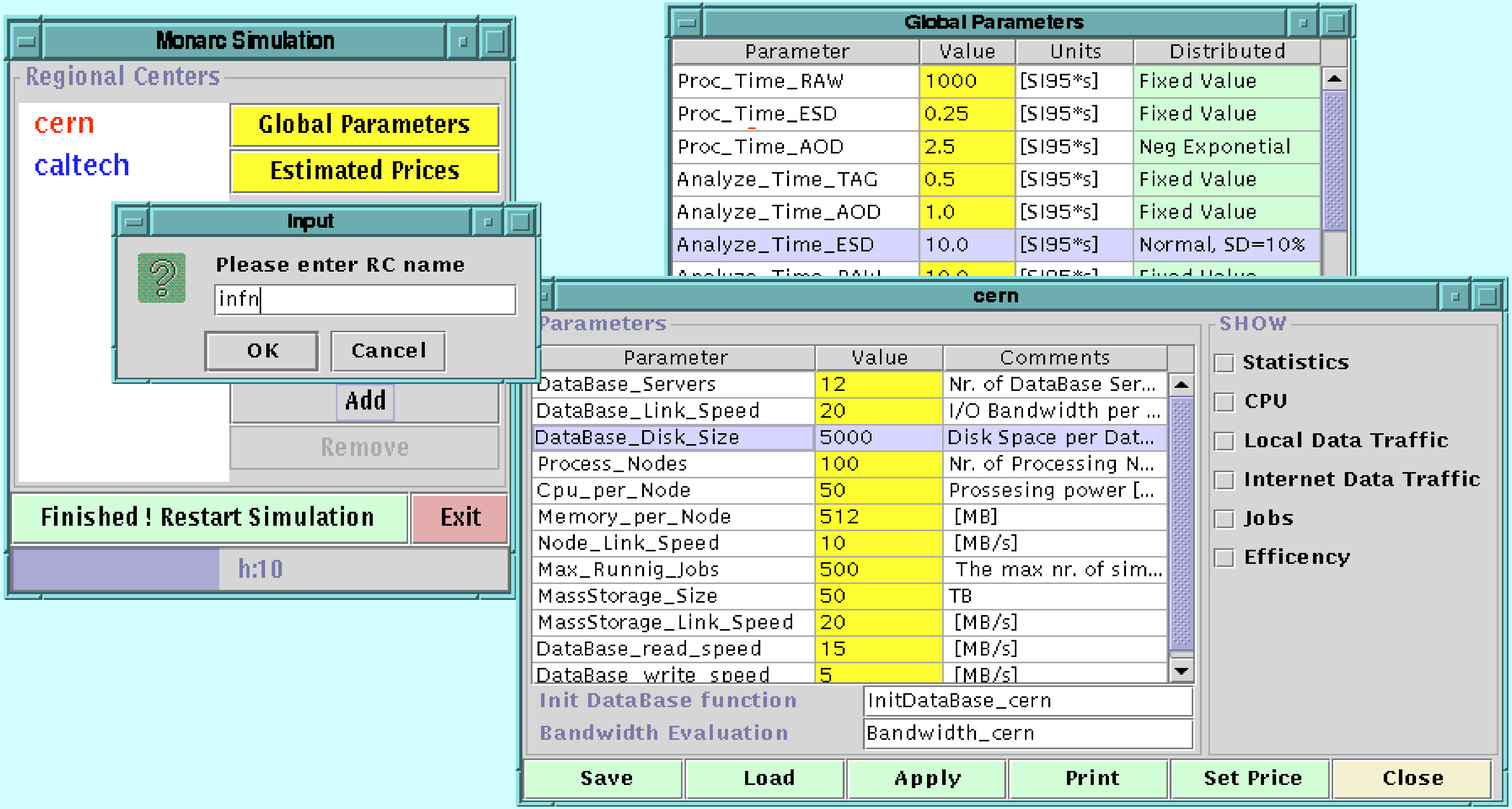
$$\delta(Ti) = F(\delta(Ti-1), \{SysP\}, \{ReqP\})$$

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).

Such a time response function allows to describe correctly Highly Nonlinear Processes or “Chaotic” Systems behavior (typical for caching, swapping…)

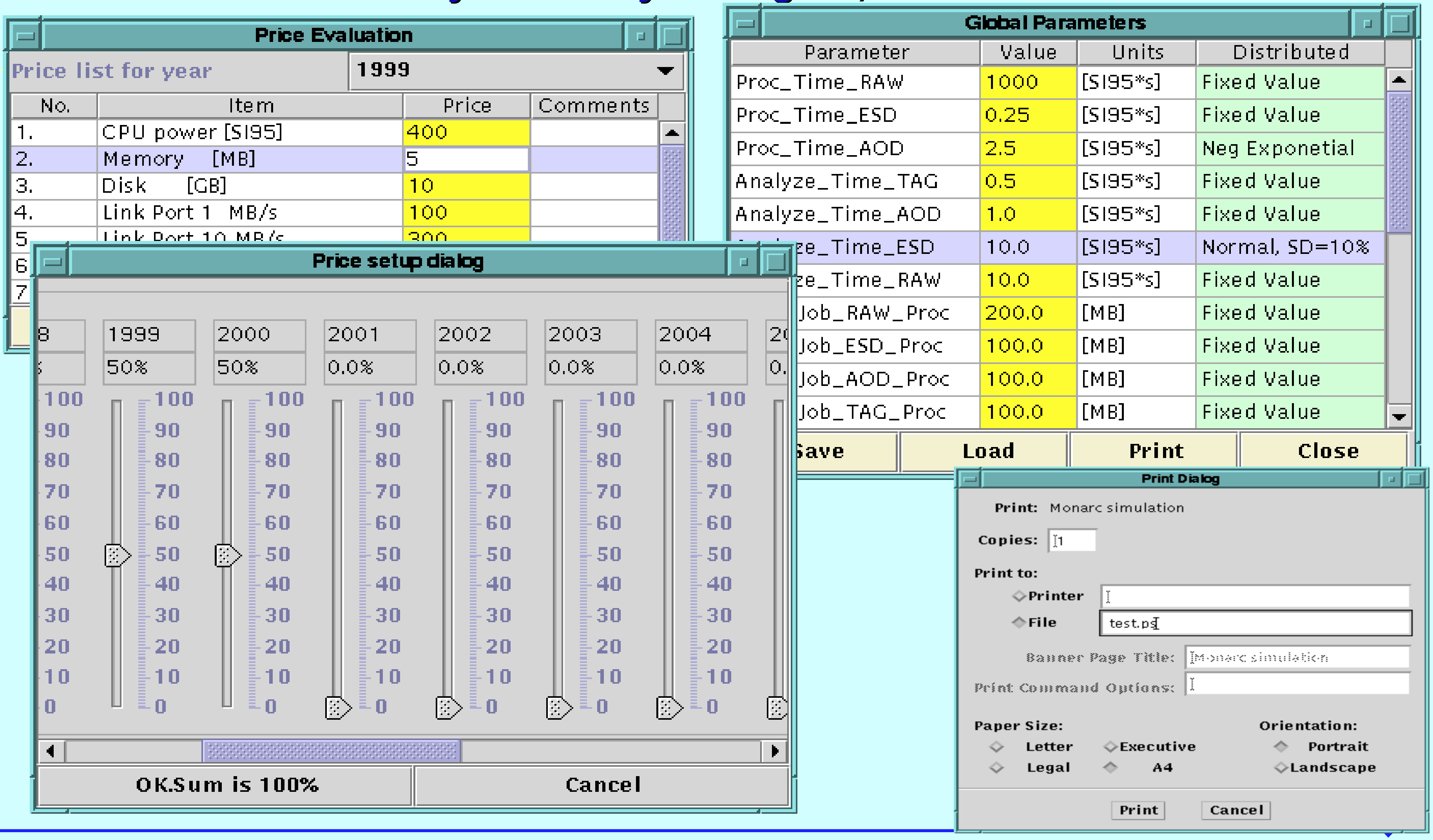
Simulation GUI

One may dynamically add and (re)configure Regional Centers parameters



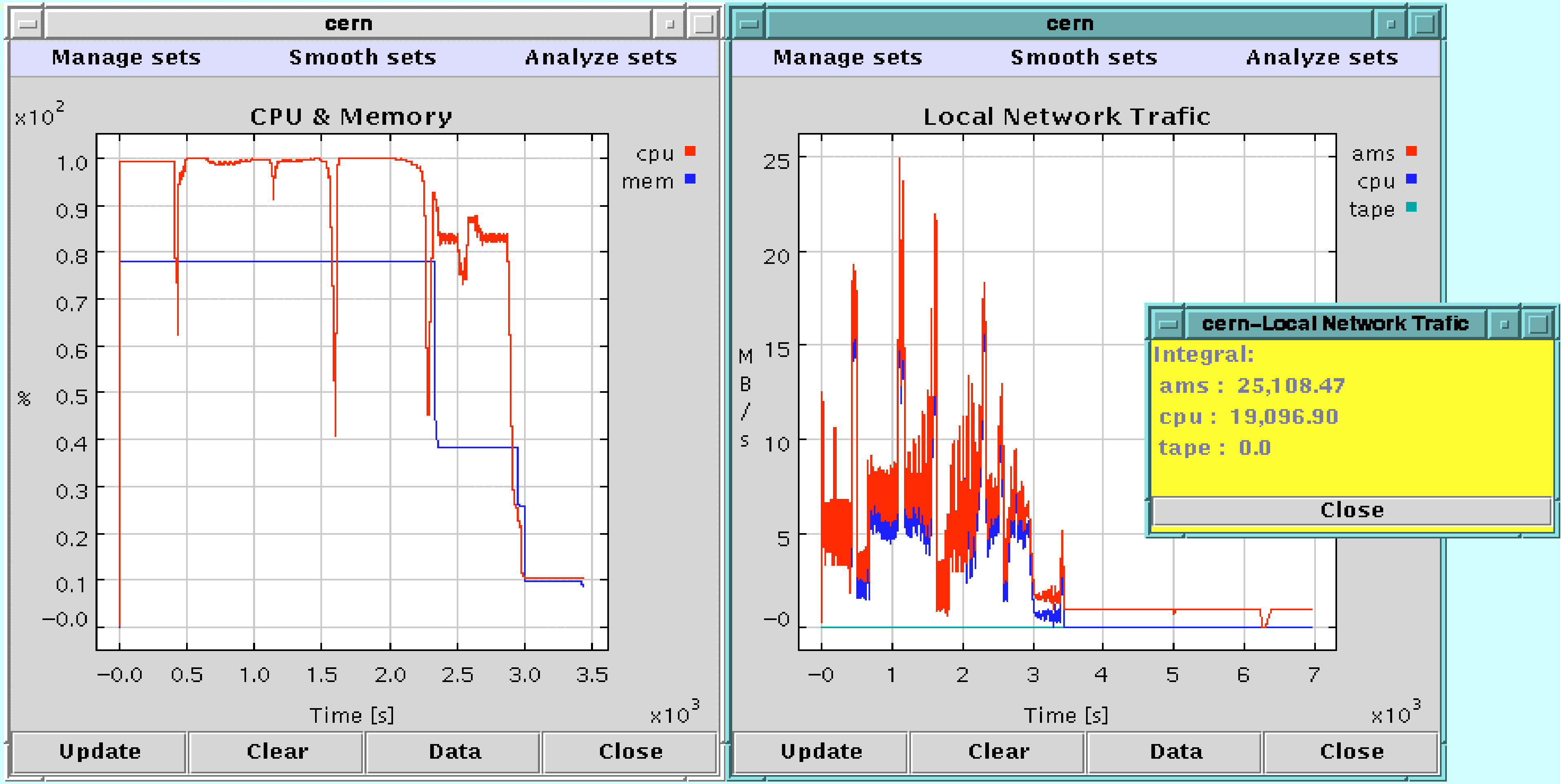
Simulation GUI (2)

Parameters can be dynamically changed , save or load from files



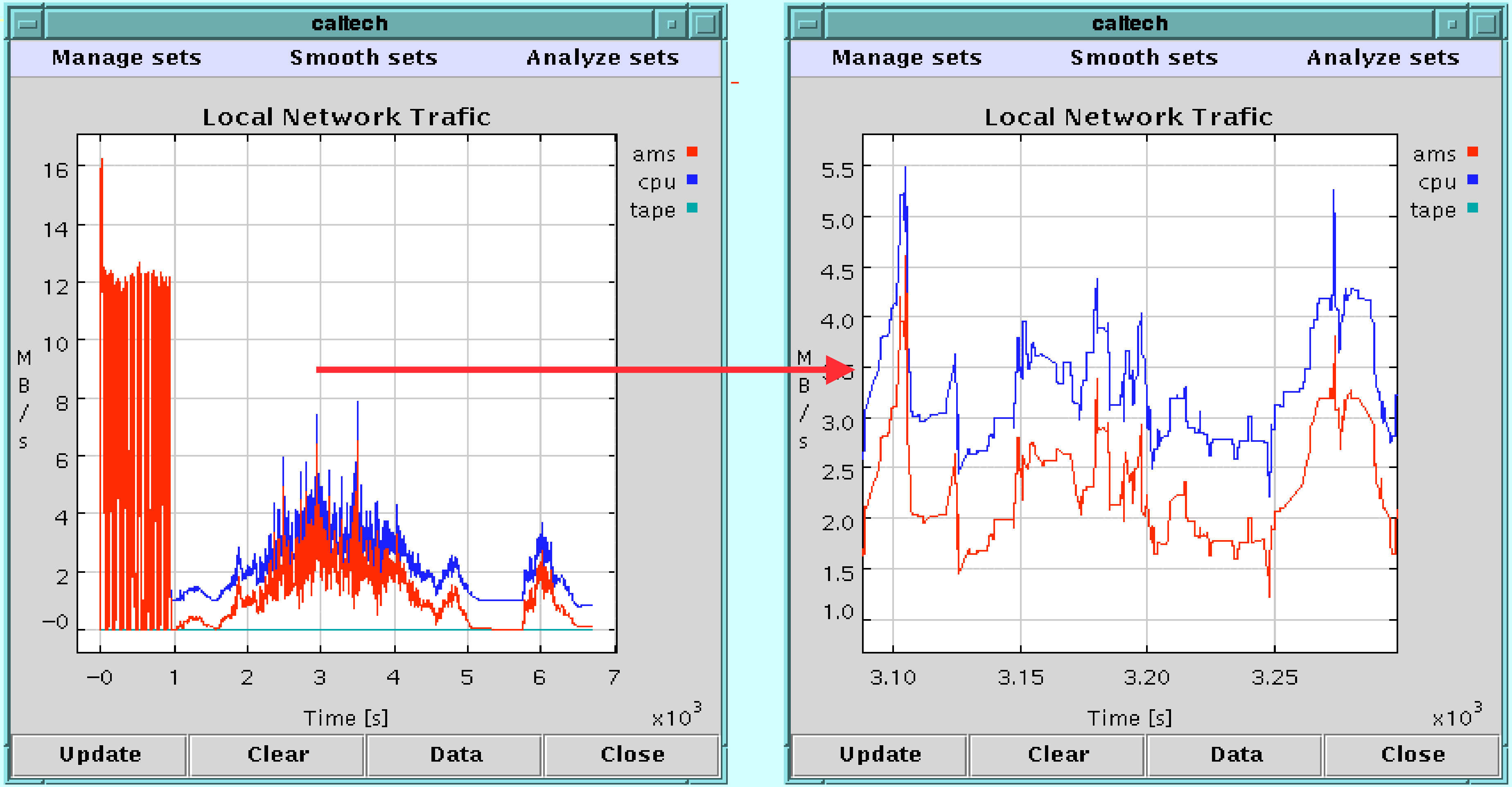
Simulation GUI (3)

On-line monitoring for major parameters in the simulation.
Tools to analyze the data.

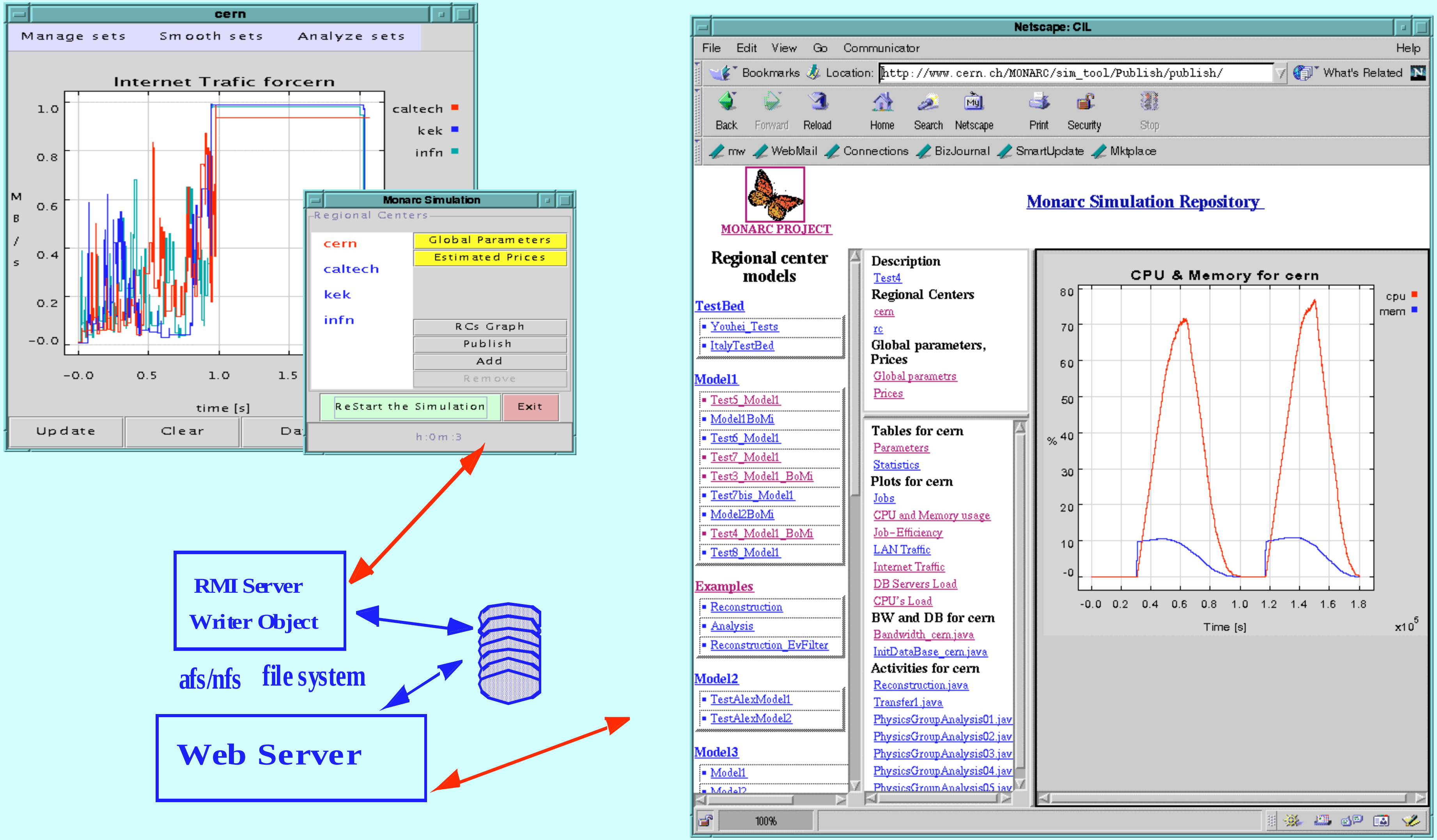


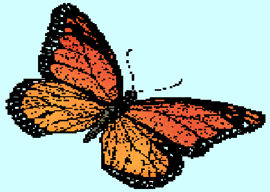
Simulation GUI (4)

Zoom in/out of all the parameters traced in the simulation

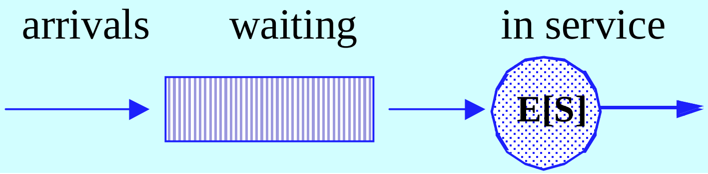


Results repository and the “publishing” procedure

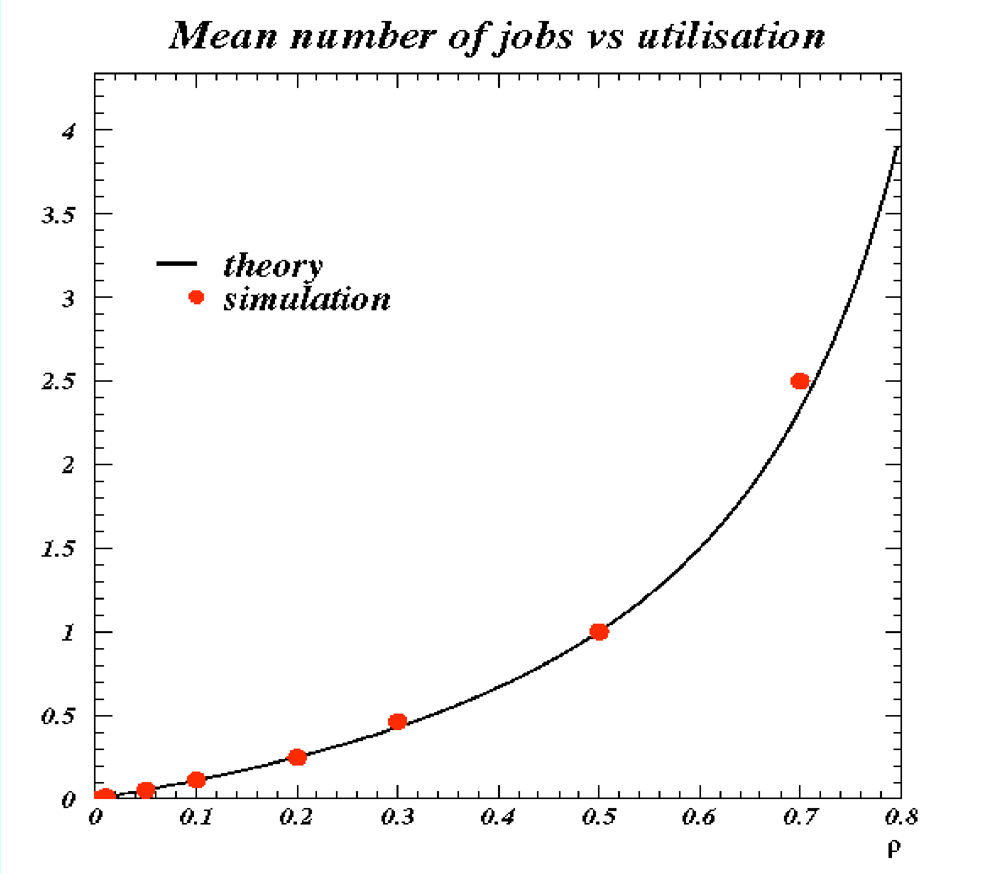




Queueing theory (1)
M | M | 1 Model

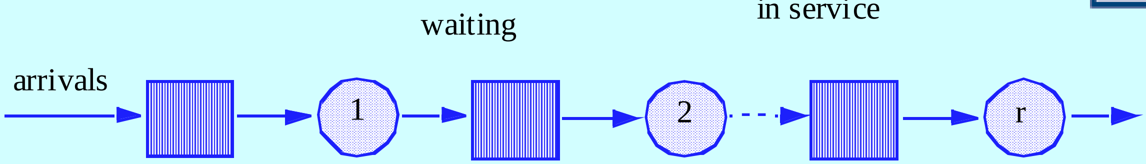


E[N] ~~Mean nu~~mber of jobs
E[R] Mean response time

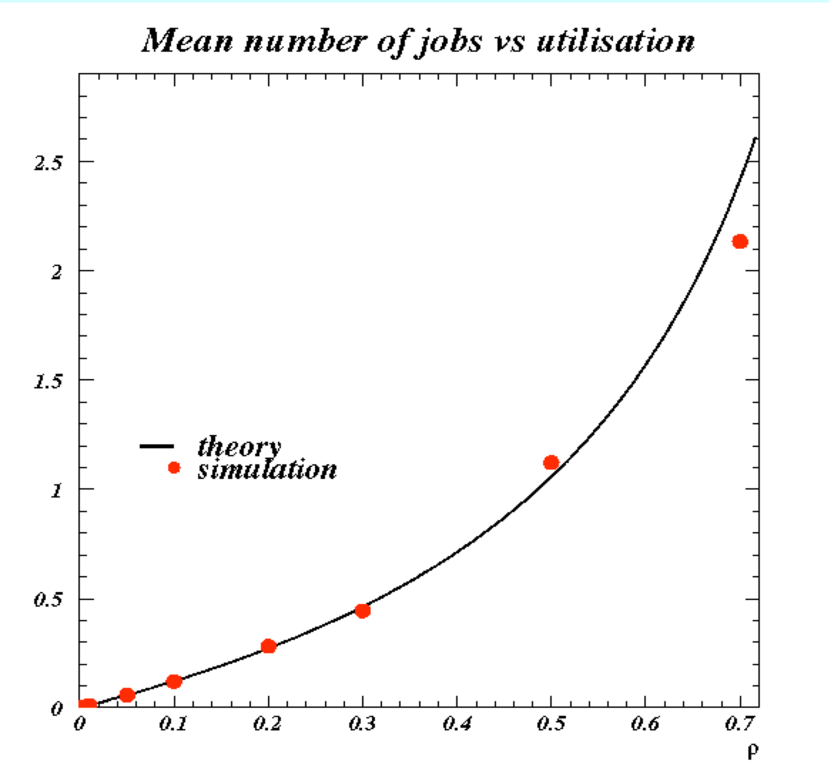


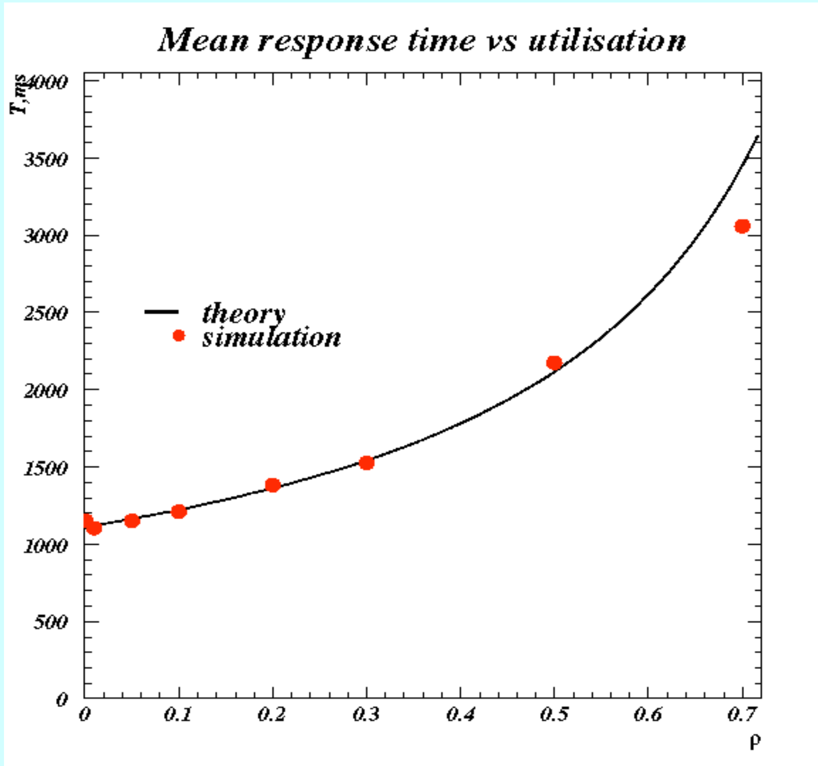
Arrival rate	E[N], sim	E[N] theory	E[R] sim	E[R] theory
1.0	0.001018	0.001001	0.001007	0.001001
10.0	0.001018	0.001010	0.010171	0.010101
30.0	0.001034	0.001053	0.032077	0.032632
100.0	0.001137	0.001111	0.112971	0.111111
200.0	0.001232	0.00125	0.246945	0.25
300.0	0.001338	0.001429	0.461039	0.428571
300.0	0.00199	0.0020	1.00087	1.0
700.0	0.003380	0.003333	2.497969	2.333333

Queueing theory(2)
M | M | 1 network queue model



$$E[N] = \sum_{i=1}^{r} E[N_i] = \sum_{i=1}^{r} \frac{\rho_i}{1-\rho_i}$$ and $$E[R] = \sum_{i=1}^{r} E[R_i] = \sum_{i=1}^{r} \frac{E[Si]}{(1-\rho i)}$$





Validation Measurements I

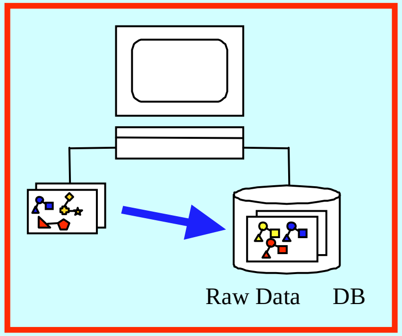
Multiple jobs reading concurrently objects from a data base.

➪Object Size = 10.8 MB

Local DB access

"atlobj02"-local

2 CPUs x 300MHz

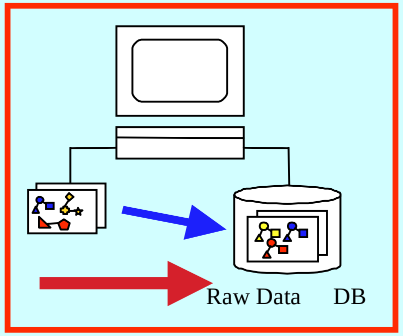


DB on local disk

13.05 SI95/CPU

"monarc01"-local

4 CPUs x 400MHz

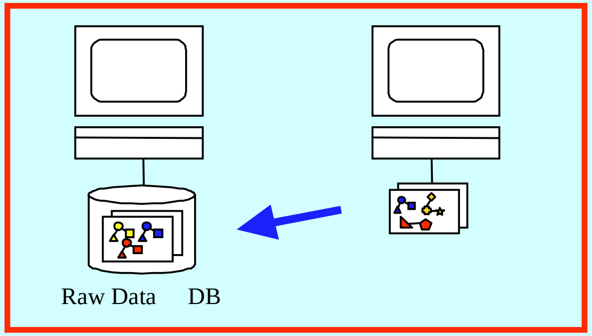


DB on local disk

17.4 SI95/CPU

DB access via AMS

server : "atlobj02"
client : "monarc01"



DB on AMS Server

monarc01 is a 4 CPUs SMP machine
atlobj02 is a 2 CPUs SMP machine

Validation Measurements I
Simulation Code

The simulation code used for parallel read test

```
public void RUN() {
  int jobs_to_doit;      int events_per_job = 5;
  Vector jobs = new Vector();
  double[]  results = new double[128]; double delta_t = 10;    double start_t;   jobs_to_doit = 1;

  for ( int tests=0; tests < 6; tests ++ ) {                        // perform simulation for 1,2,4,8,16,32 parallel jobs
      start_t = clock();
      for( int k =0; k< jobs_to_doit; k++) {                   // Job submission
          Job job = new Job( this, Job.CREATE_AOD, "ESD", k*events_per_job+1, (k+1)*events_per_job, null, null);
          job.setMaxEvtPerRequest(1);
          jobs.addElement(job);
          farm.addJob( job);
      }

      boolean done = false;
      while ( !done ) {        done = true;                          // wait for all submitted jobs to finish
          for ( int k=0; k < jobs_to_doit; k++)
            if (  !((Job) jobs.elementAt(k) ).isDone() ) done=false; else results[k] = ( j.tf - start_t);    // keep the processing time per job
          sim_hold( delta_t/10);                                       // wait between testing that all jobs are done
       }
// Compute and print the results
      sim_hold(delta_t);                                               // wait between next case
      jobs_to_doit *=2;                                                // prepare the new number of parallel jobs
      jobs.removeAllElements();                                        // clean the array used to keep the running jobs
  }
}
```

Validation Measurements I

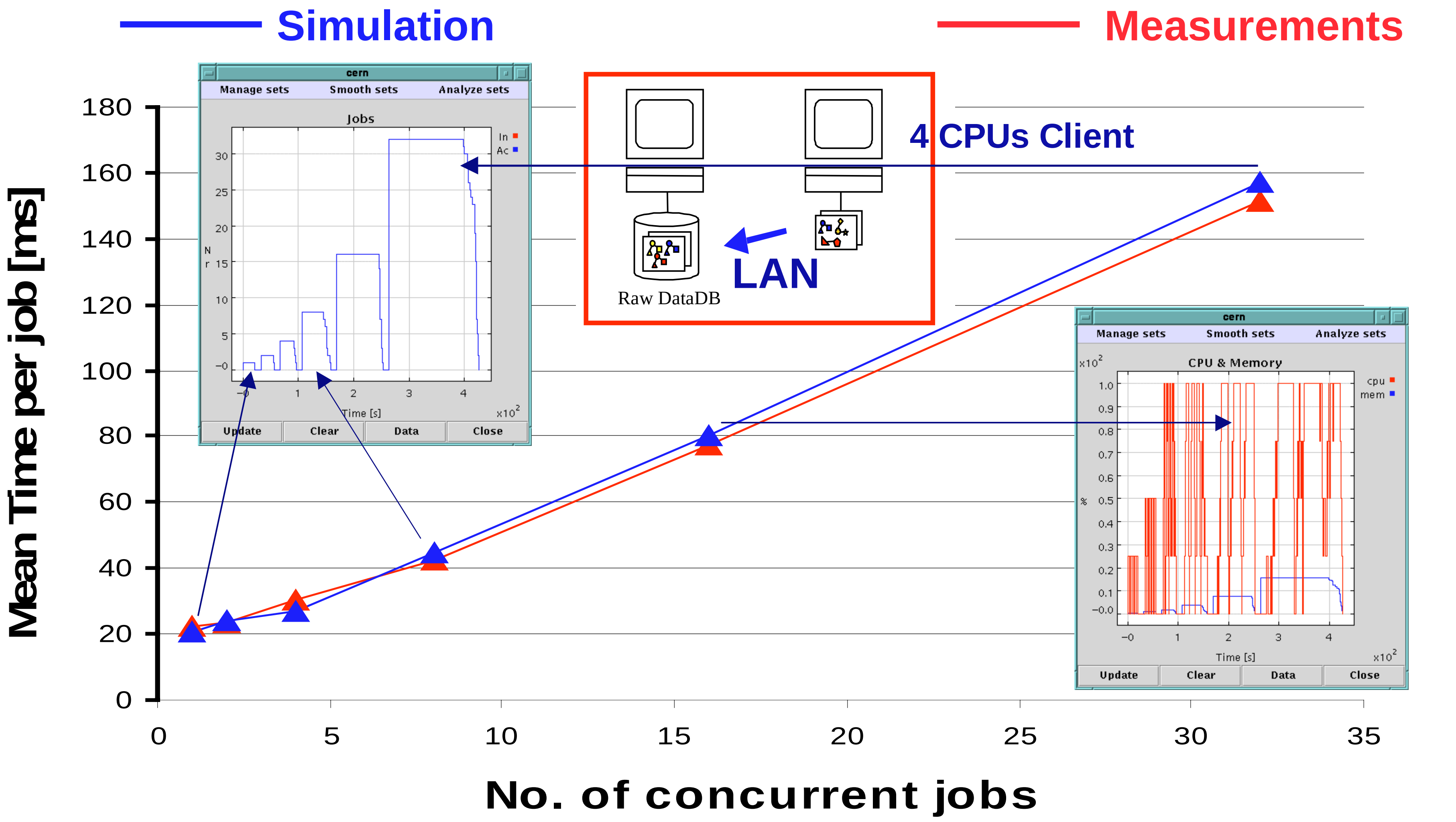
The same "User Code" was used for different configurations:

- **Local Data Base Access**
- **AMS Data Base Access**
- **Different CPU power**

For LAN parameterization :

- **RTT ~ 1ms**
- **Maximum Bandwidth 12 MB/s**

Validation Measurements I
The AMS Data Access Case

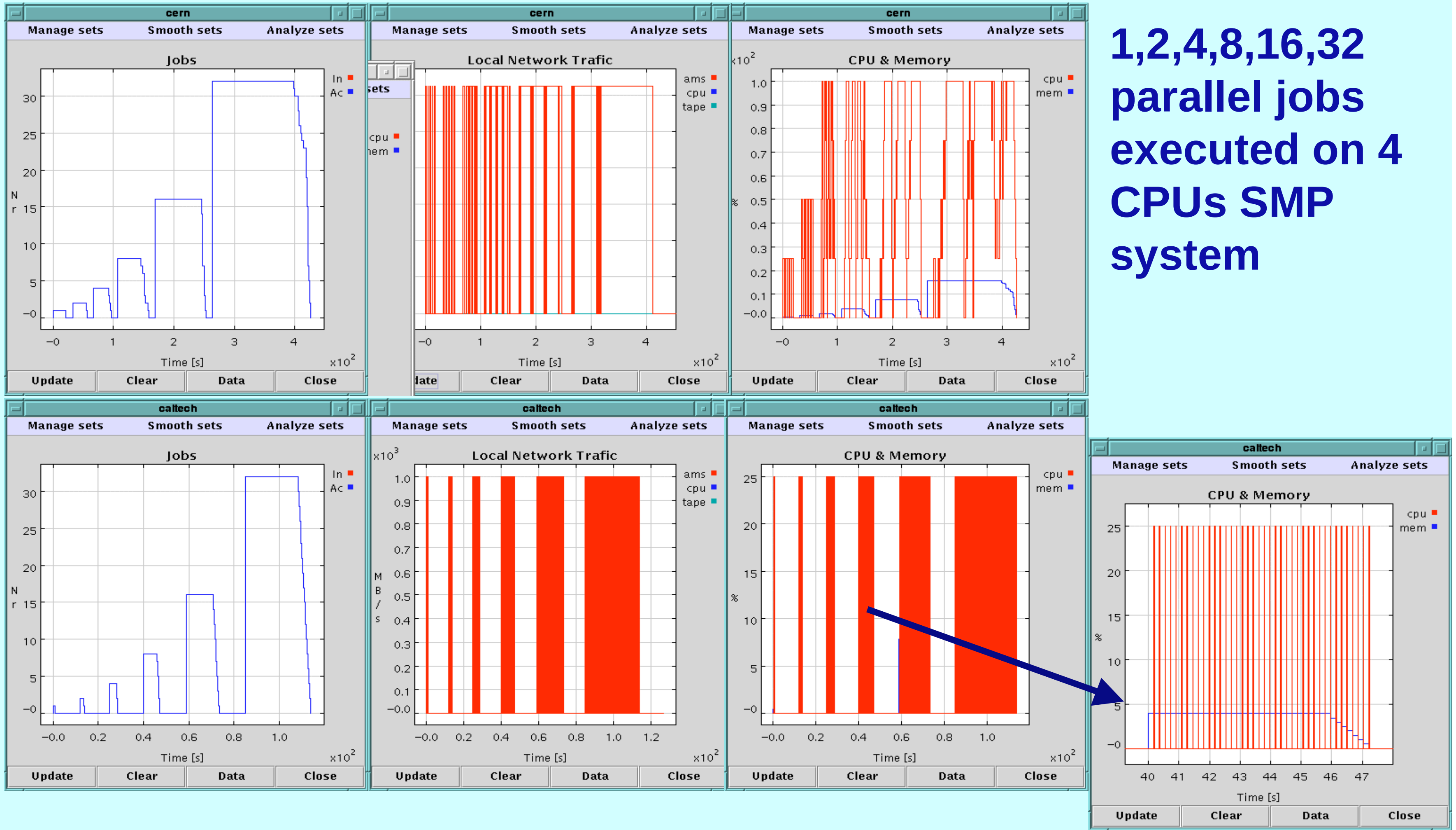


Validation Measurements I
Simulation Results

Simulation results for AMS & Local Data Access

DB access via AMS

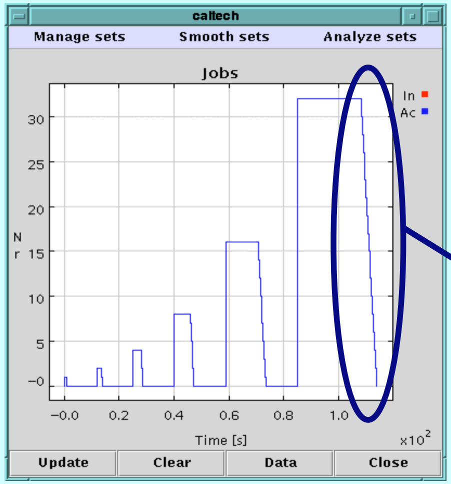
Local DB access



1,2,4,8,16,32 parallel jobs executed on 4 CPUs SMP system

Validation Measurements I

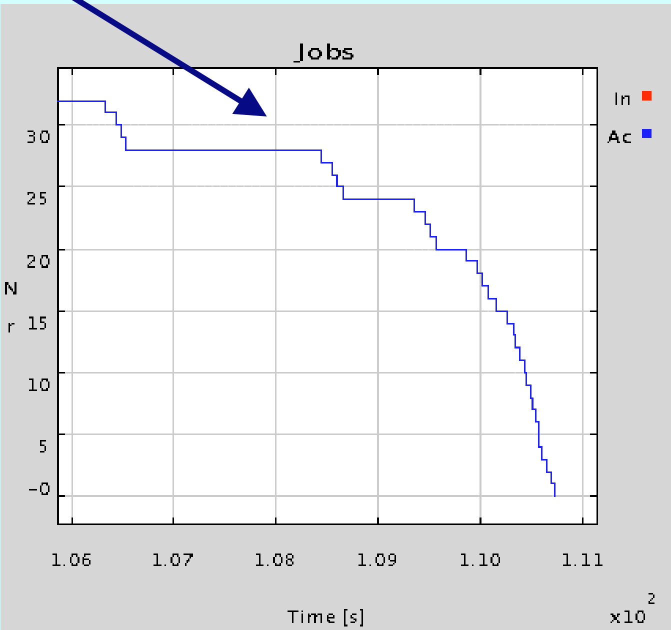
The Distribution of the jobs processing time

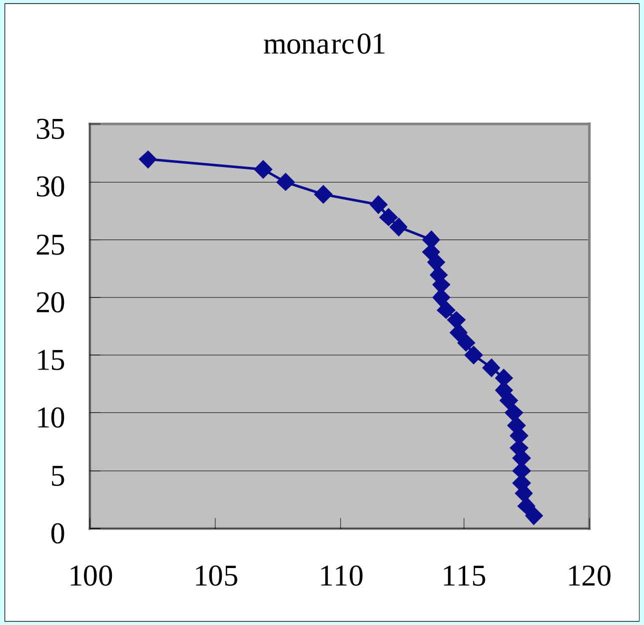


Simulation
mean 109.5

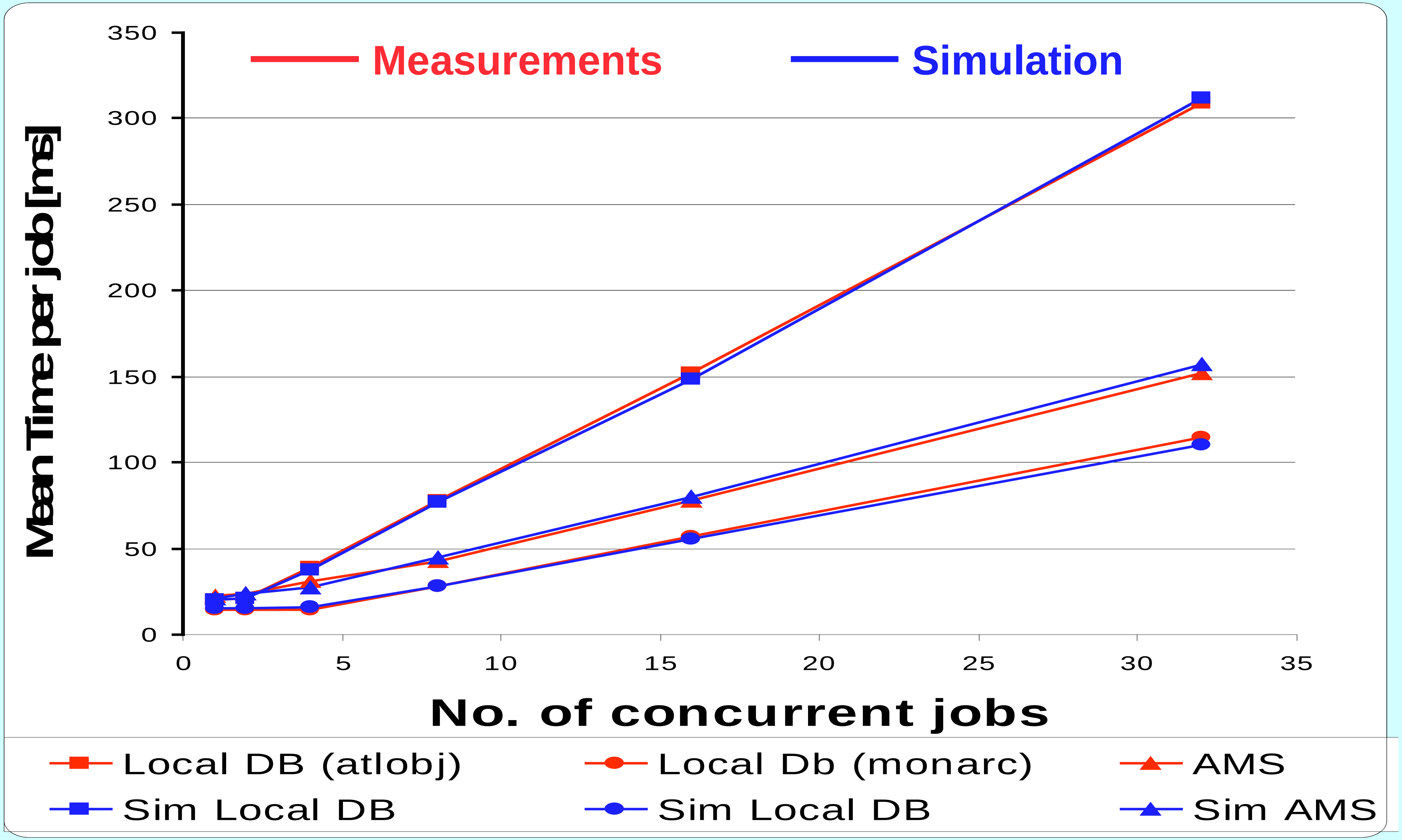
Measurement
mean 114.3

Local DB access
32 jobs





Validation Measurements I
Measurements & Simulation



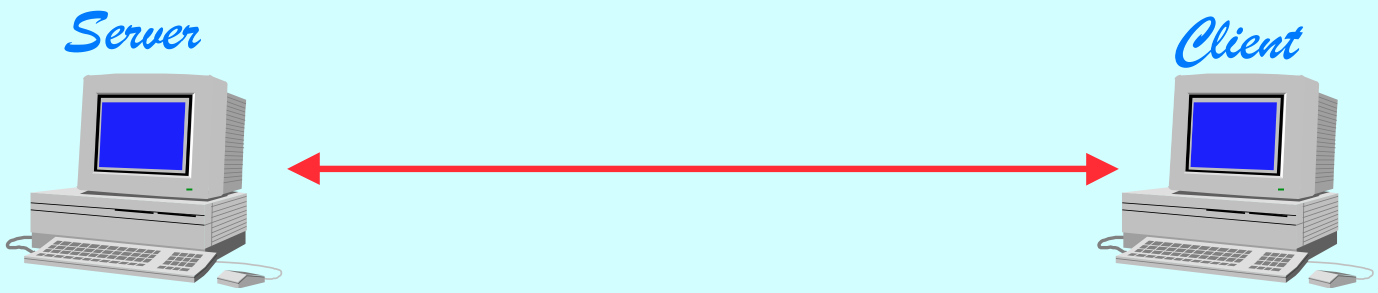
Validation Measurements II

Running multiple concurrent client programs to read events stored into an OODB.

- Event Size 40 KB , Normal Distributed (SD = 20%)
- Processing time per event ~ 0.17 Si95 * s
- Each job reads 2976 events

- One AMS Server is used for all the Clients.
- Perform the same "measurements" for different network connectivity between the Server and Client.

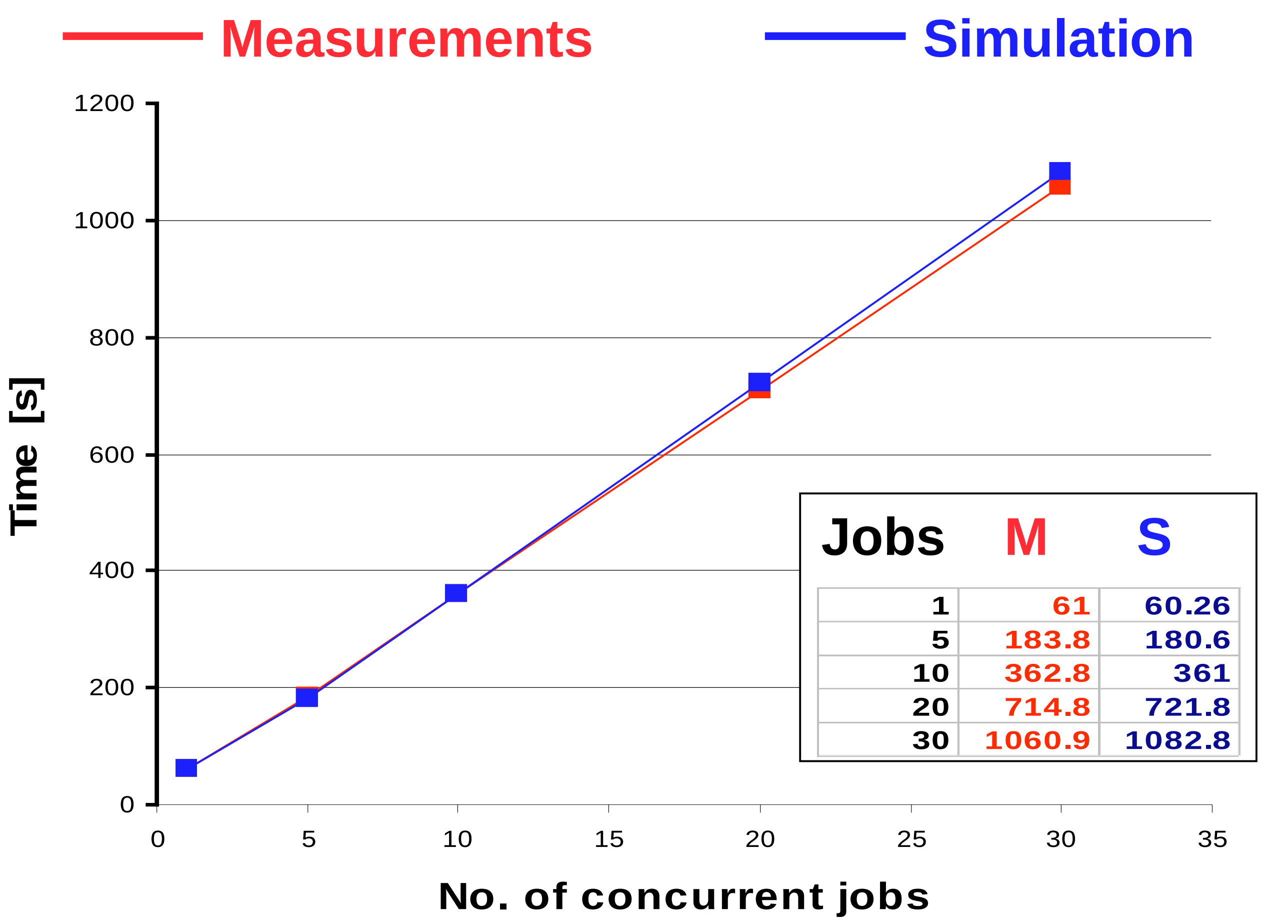
Validation Measurements II



	Server	Link	Client
Test 1 **CNAF**	**sunlab1** Sun Ultra5, 333 MHz	**1000BaseT**	**gsun** Sun Ultra5, 333 MHz
Test 2 **PADOVA**	**cmssun4** Sun Ultra10, 333 MHz	**10BaseT**	**vlsi06** Sun Sparc20, 125 MHz
Test 3 **CNAF-CERN**	**sunlab1** Sun Ultra15, 333 MHz	**2 Mbps**	**monarc01** Sun Enterprise 4X450 MHz

Validation Measurements II
Test 1

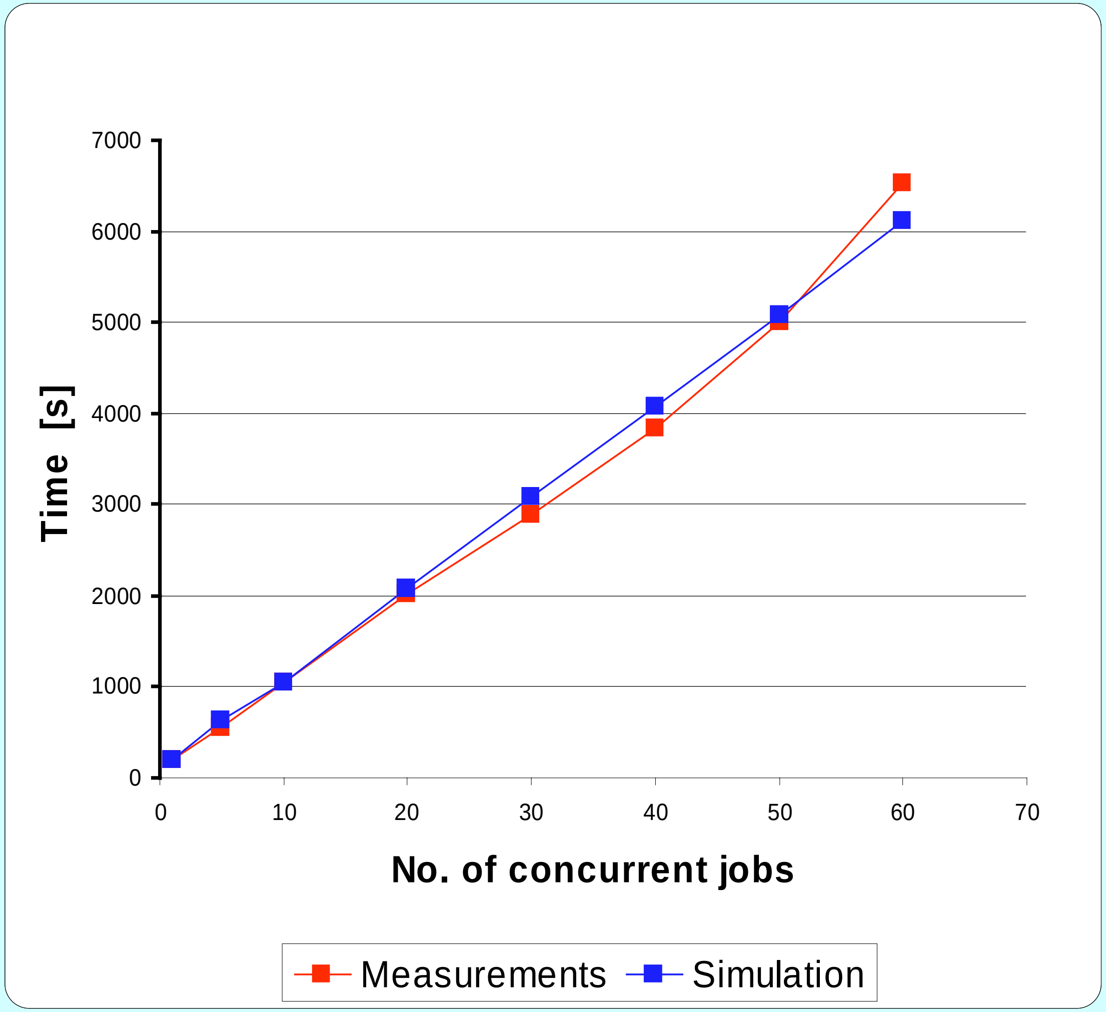
Gigabit Ethernet Client - Server



Jobs	M	S
1	61	60.26
5	183.8	180.6
10	362.8	361
20	714.8	721.8
30	1060.9	1082.8

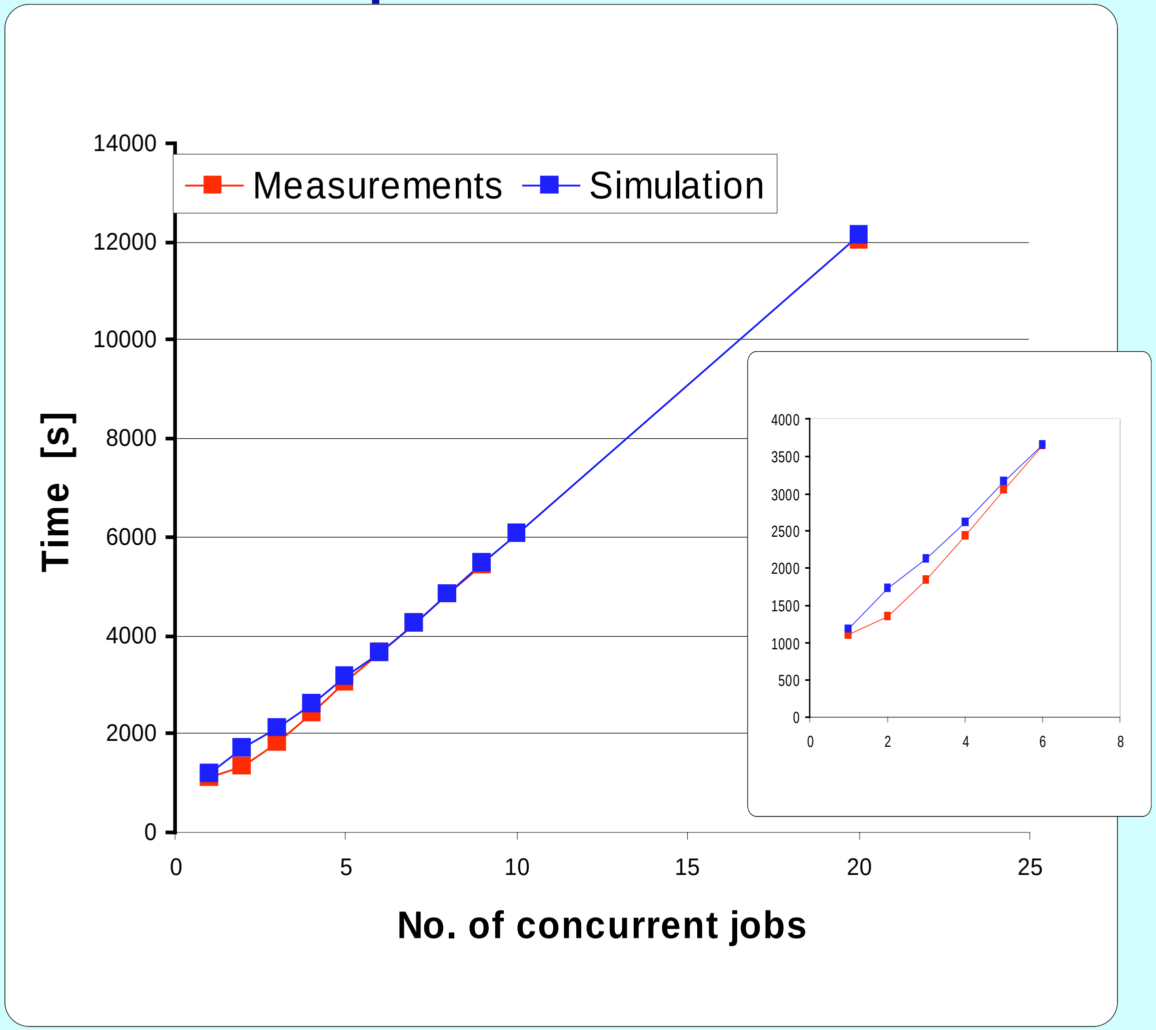
Validation Measurements II
Test 2

Ethernet Client - Server

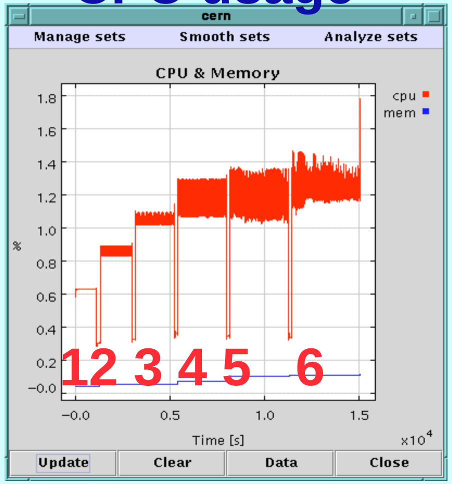


Validation Measurements II
Test 3

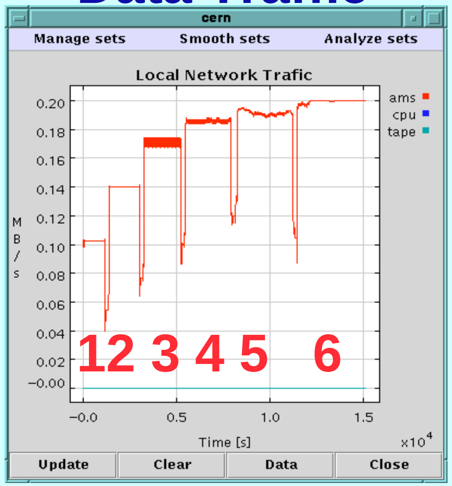
2Mbps WAN Client - Server



CPU usage

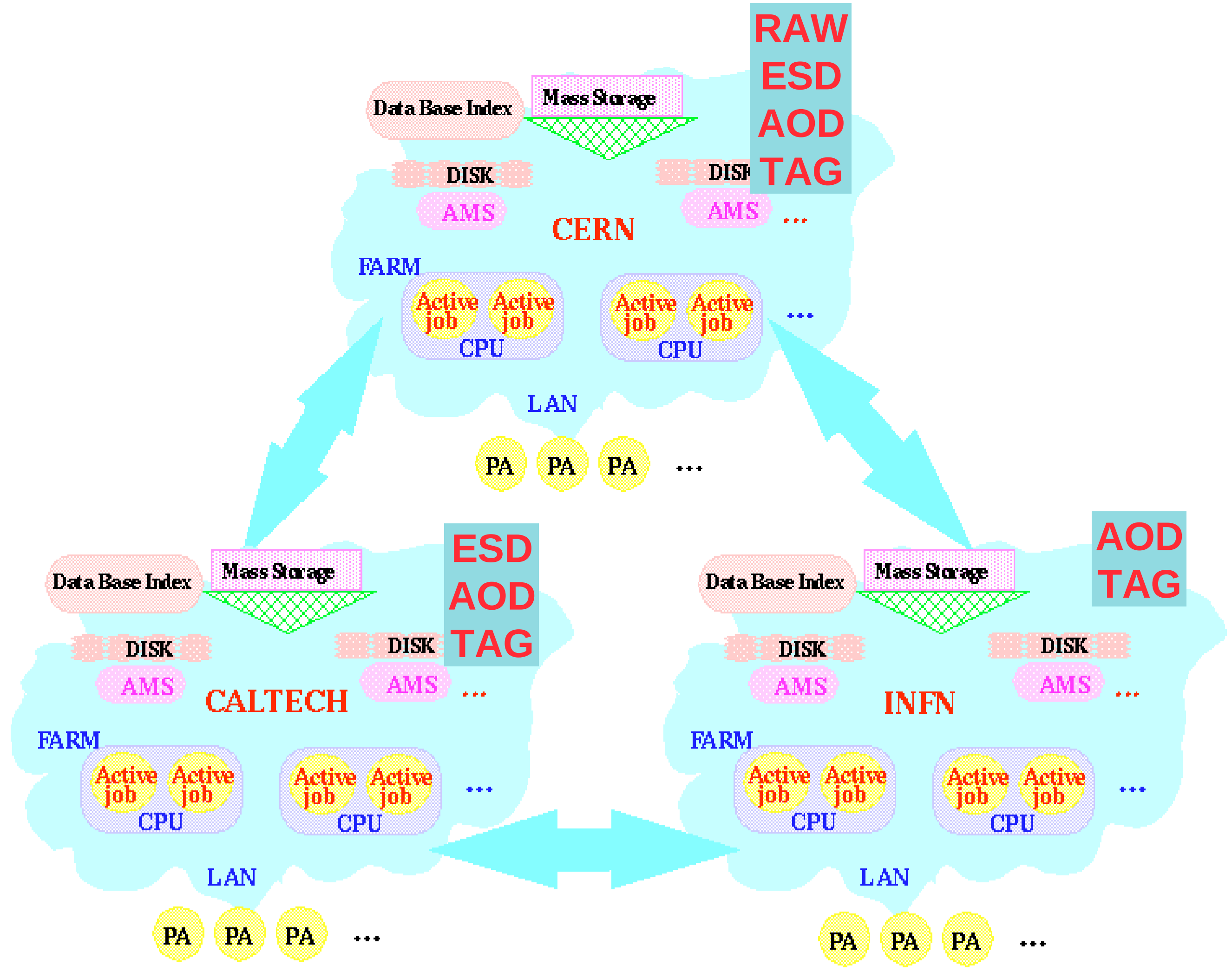


Data Traffic



Physics Analysis Example

- Similar data processing jobs are performed in three RCs
- "CERN" has RAW, ESD, AOD, TAG
- "CALTECH" has ESD, AOD, TAG
- "INFN" has AOD, TAG



Physics Analysis Example

One Physics Analysis Group:

- Analyze $4 * 10^6$ events per day .
- Submit 100 Jobs (~40 000 events per Job)
- Each group starts at 8:30 local time. More jobs are submitted in the first part of the day.
- A Job analyzes AOD data and requires ESD data for 2% of the events and RAW data for 0.5%of the events.

Physics Analysis Example

- **"CERN" Center (RAW ESD AOD TAG):**
  - 10 Physics Analysis Groups --> access to 40 *10^6 events
  - 200 CPU units at 50 Si95
  - 1000 jobs to run
  - half of RAW data --> on tape

- **"CALTECH" Center (ESD, AOD, TAG)**
  - 5 Physics Analysis Groups --> access to 20 *10^6 events
  - 100 CPU units
  - 500 jobs to run

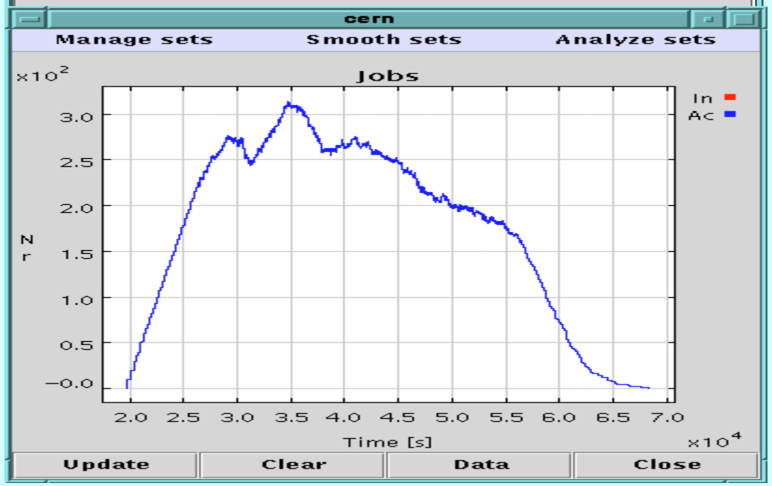
- **"INFN" Center (AOD, TAG)**
  - 2 Physics Analysis Groups --> access to 8 *10^6 events
  - 40 CPU units
  - 200 jobs to run

Physics Analysis Example

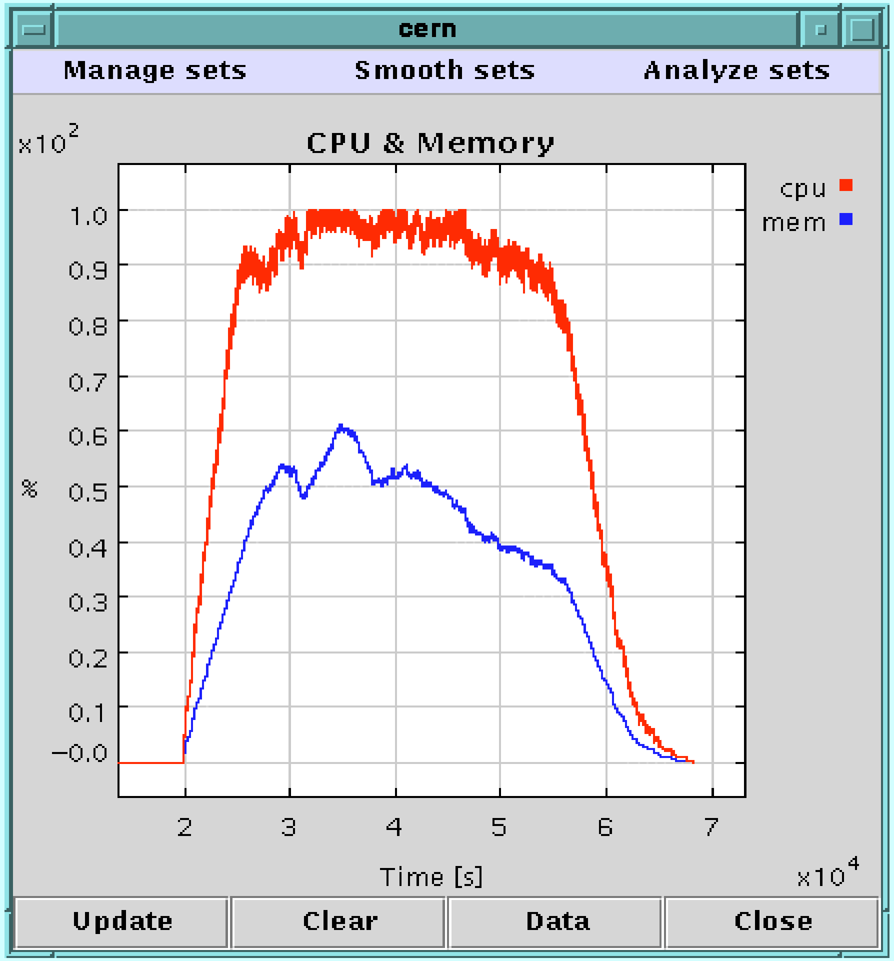
“CERN”

Statistics @cern

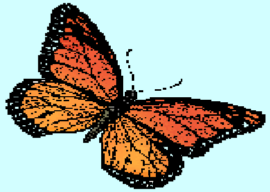
Parameter	Value
Estimated Price [k$]	5,136.00
Nr. of Jobs Processed	1000
Nr. of Jobs Aborted	0
CPU usage– Integrated mean [%] ...	49.433
Total CPU used [SI95*s]	324.420 * 10^6
DataBase servers write	0.00 [MB]
DataBase servers read	941.411 [GB]
Processed Events	4.0E7
Processing Rate [events/s]	609.497
Global Read Rate from DB [MB/s]	12.885
Global Write Rate to DB [MB/s]	0.00



Jobs in the System



CPU & Memory Usage

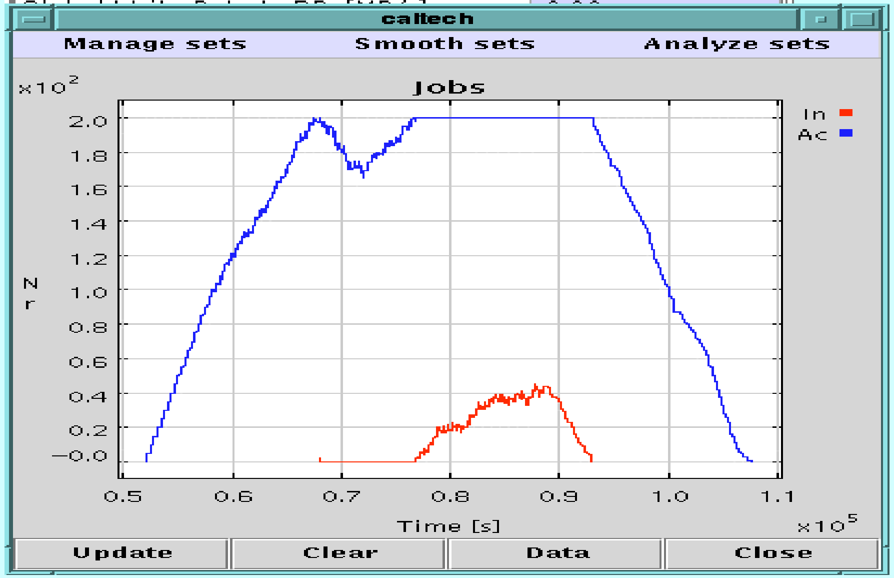


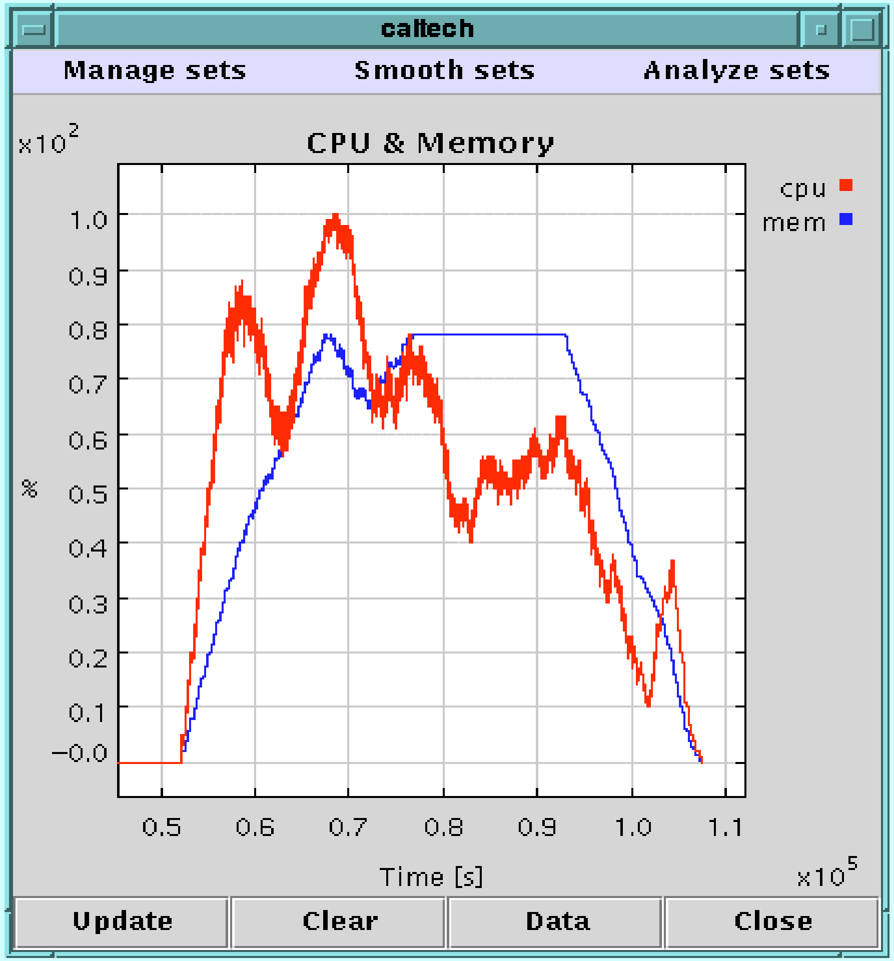
Physics Analysis Example

"CALTECH"

Statistics @caltech

Parameter	Value
Estimated Price [k$]	7,269.40
Nr. of Jobs Processed	500
Nr. of Jobs Aborted	0
CPU usage– Integrated mean [%]	28.579
Total CPU used [SI95*s]	153.717 * 10^6
DataBase servers write	0.00 [MB]
DataBase servers read	260.373 [GB]
Processed Events	2.0E7
Processing Rate [events/s]	185.923
Global Read Rate from DB [MB/s]	2.420





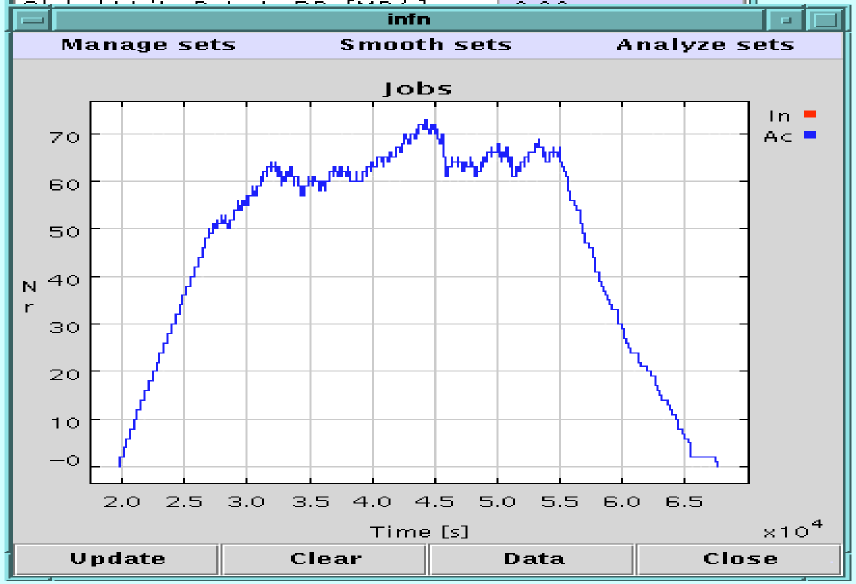
Jobs in the System **CPU & Memory Usage**

Physics Analysis Example

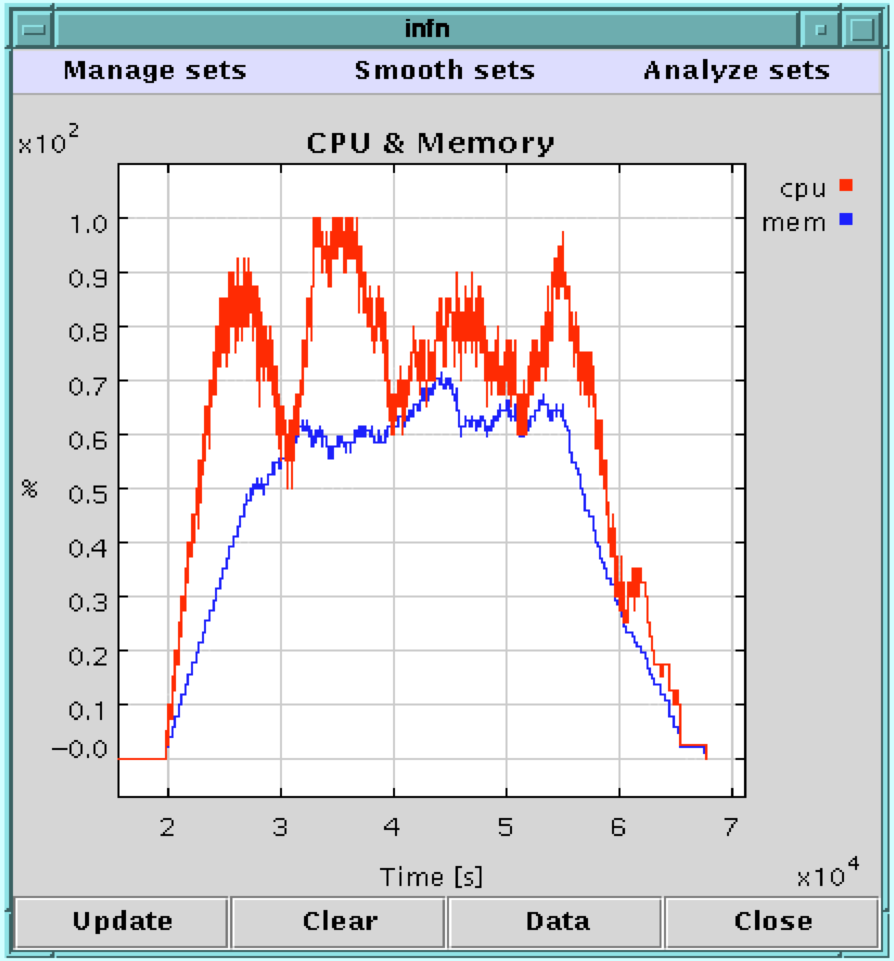
"INFN"

Statistics @ infn

Parameter	Value
Estimated Price [k$]	5,909.80
Nr. of Jobs Processed	200
Nr. of Jobs Aborted	0
CPU usage- Integrated mean [%] ...	45.476
Total CPU used [SI95*s]	61.485 * 10^6
DataBase servers write	0.00 [MB]
DataBase servers read	88.178 [GB]
Processed Events	8000000.0
Processing Rate [events/s]	118.339
Global Read Rate from DB [MB/s]	1.304



Jobs in the System

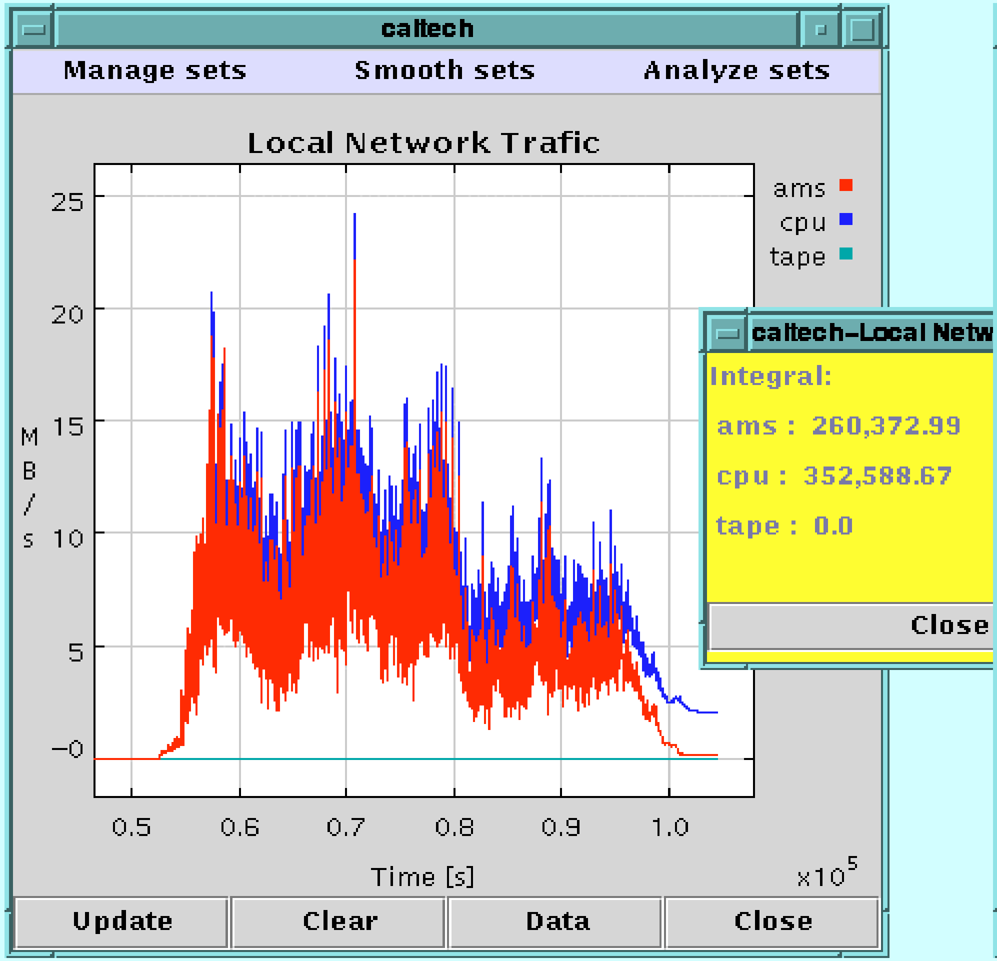


CPU & Memory Usage

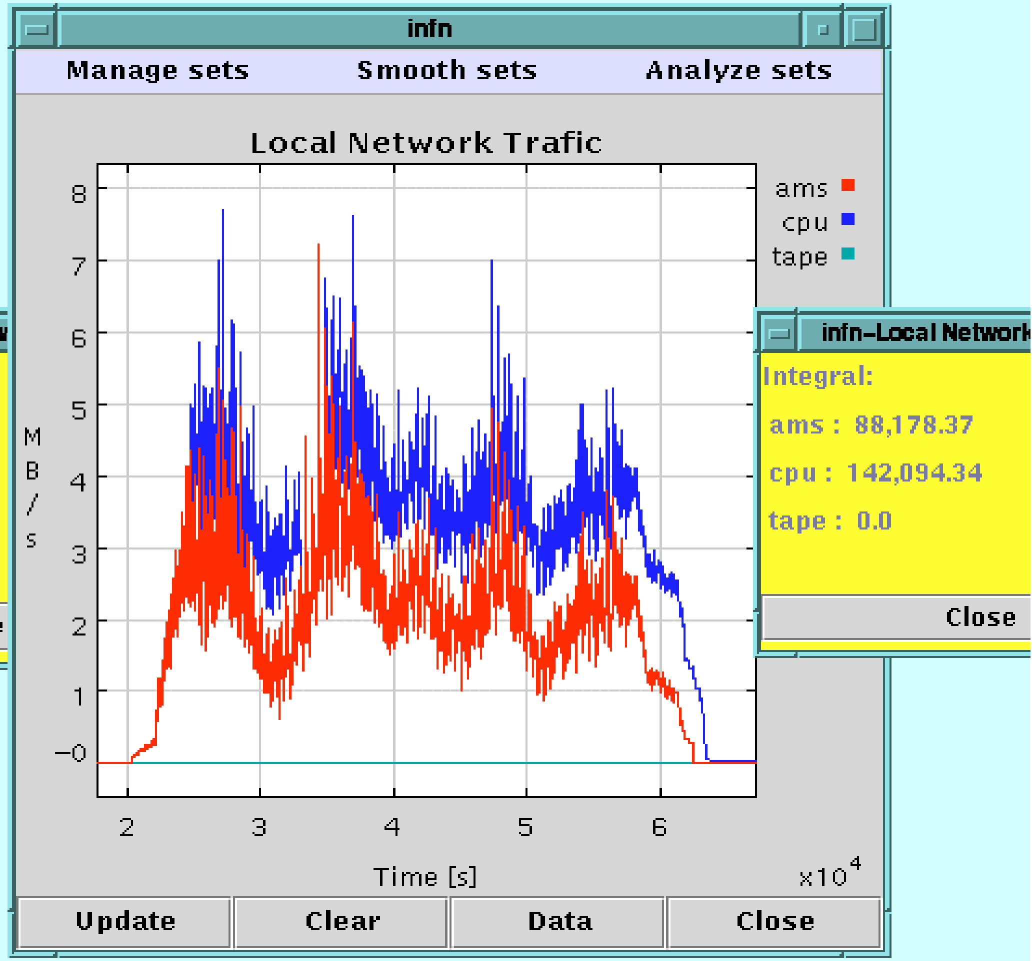
Physics Analysis Example

Local Data Traffic

"CALTECH"



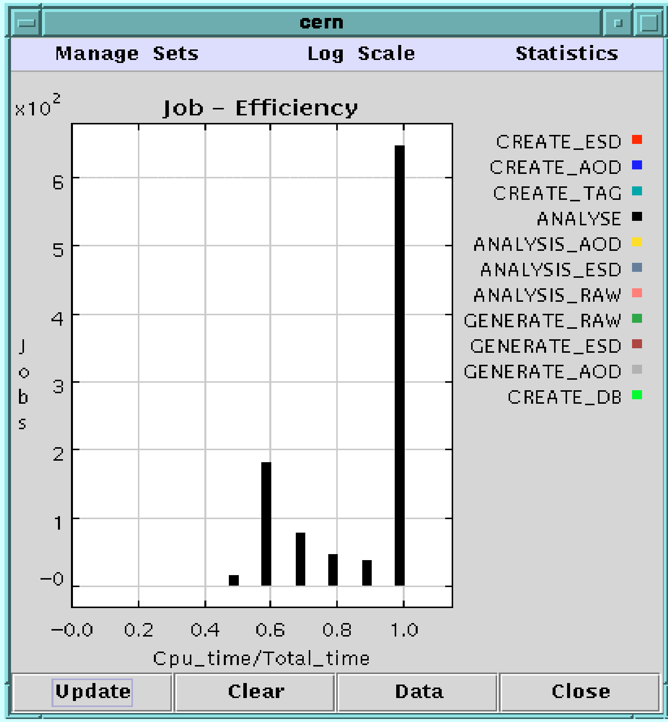
"INFN"



Physics Analysis Example

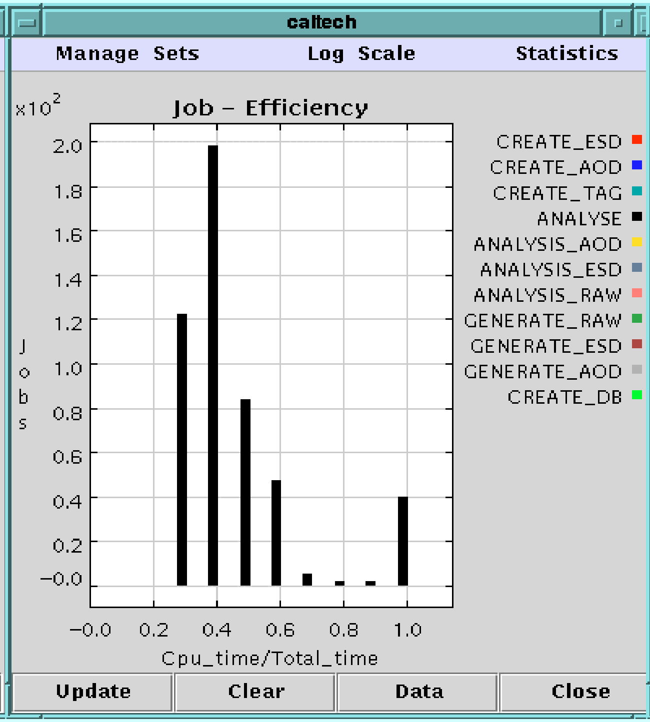
Job efficiency distribution

"CERN"



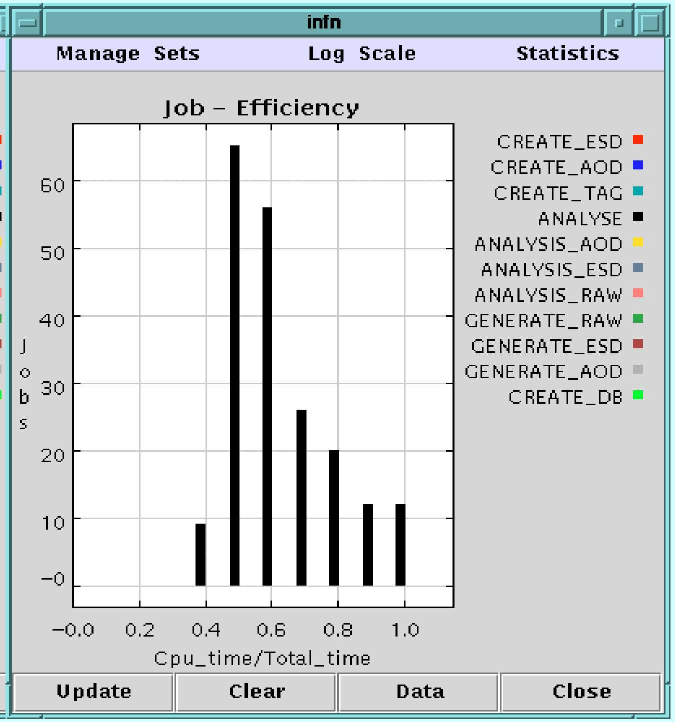
Mean 0.83

"CALTECH"

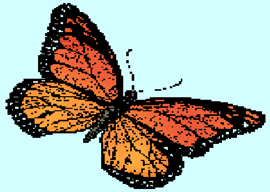


Mean 0.42

"INFN"



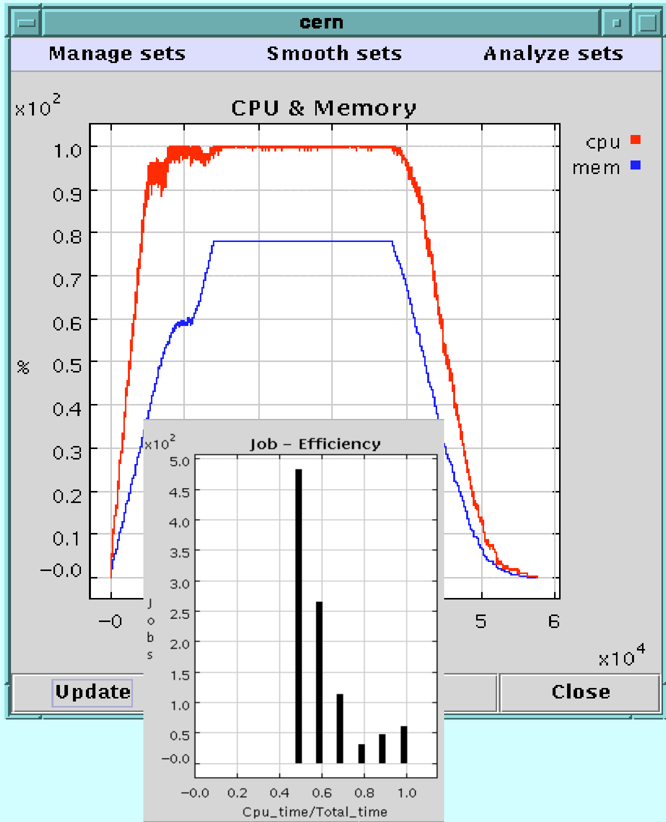
Mean 0.57



Resource Utilisation vs. Job's Response Time

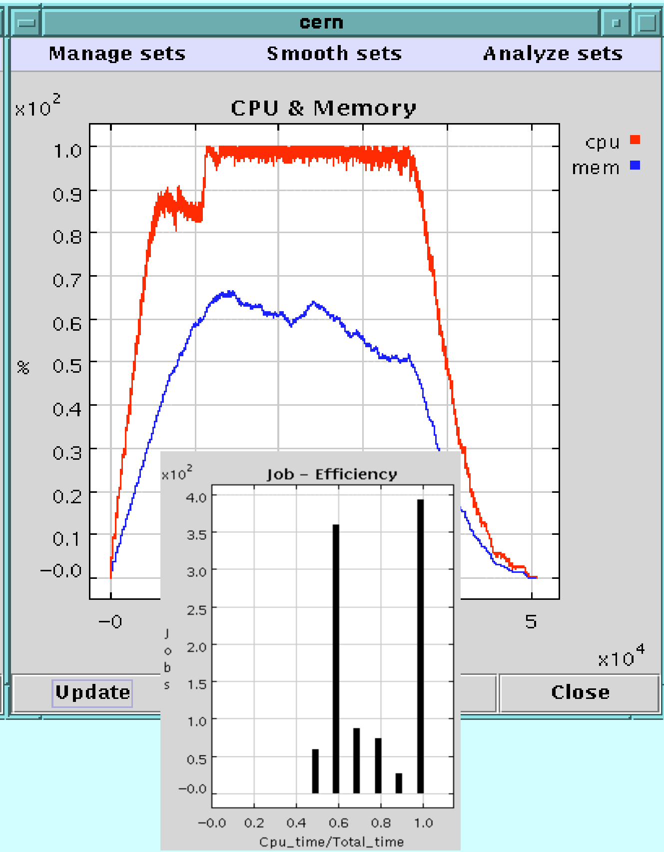
"CERN" - Physics Analysis Example

180 CPUs



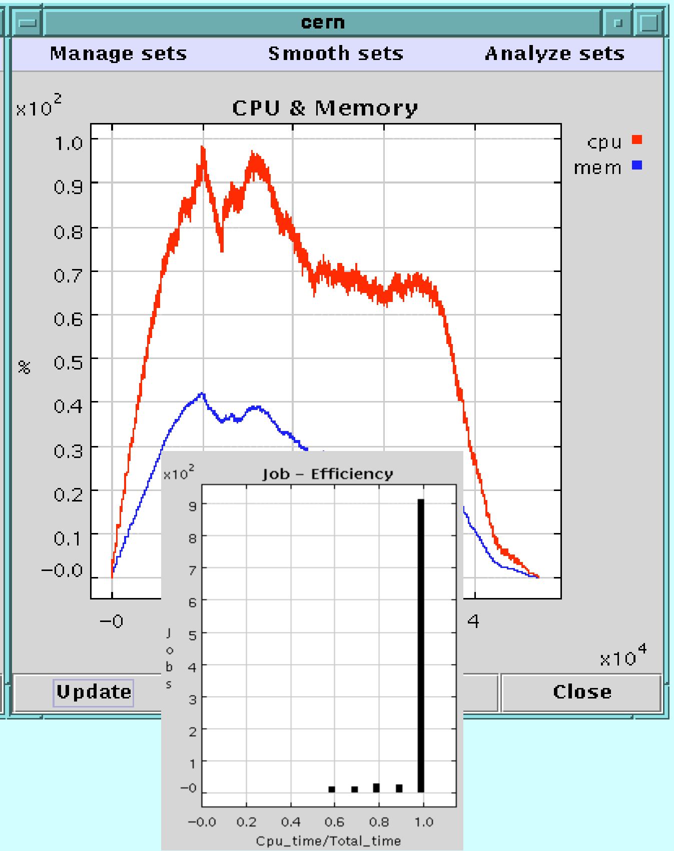
Mean 0.55

200 CPUs



Mean 0.72

250 CPUs



Mean 0.93

The Plan for Short Term Developments

- Implement the Data Base replication mechanism.
- Inter-site scheduling functions to allow job migrations.
- Improve the evaluation of the estimated cost function and define additional "cost functions" to be considered for further optimisation procedures.
- Implement the "desktop" data processing model.
- Improve the Mass Storage Model and procedures to optimise the data distribution.
- Improve the functionality of the GUIs.
- Add additional functions to allow saving and analysing the results and provide a systematic way to compare different models.
- Build a Model Repository and a Library for "dynamically loadable activities" and typical configuration files.

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.

❋ New dedicated measurements to evaluate realistic parameters for the simulation program are in progress.