High Throughput Computing (HTC)

Miron Livny

Computer Sciences Department
University of Wisconsin — Madison
Madison, Wisconsin
{miron@cs.wisc.edu}

High Throughput Computing (HTC) on Distributively Owned Resources

Performance vs. Throughput

- ***High Performance***

  Very large amounts of processing capacity for relatively short periods of time - Floating Point Operations per second.

- ***High Throughput***

  Large amounts of processing capacity for very long periods of time - Floating Point Operations per week.

Distributed

Ownership

Powerful computing resources are owned by individuals, groups, departments, and institutes.

- Huge increase in the processing capacity owned by the scientific community; (more power and more machines)
- Much smaller increase in the capacity accessible by a scientist; (privatization of processing capacity)

Wait while Idle

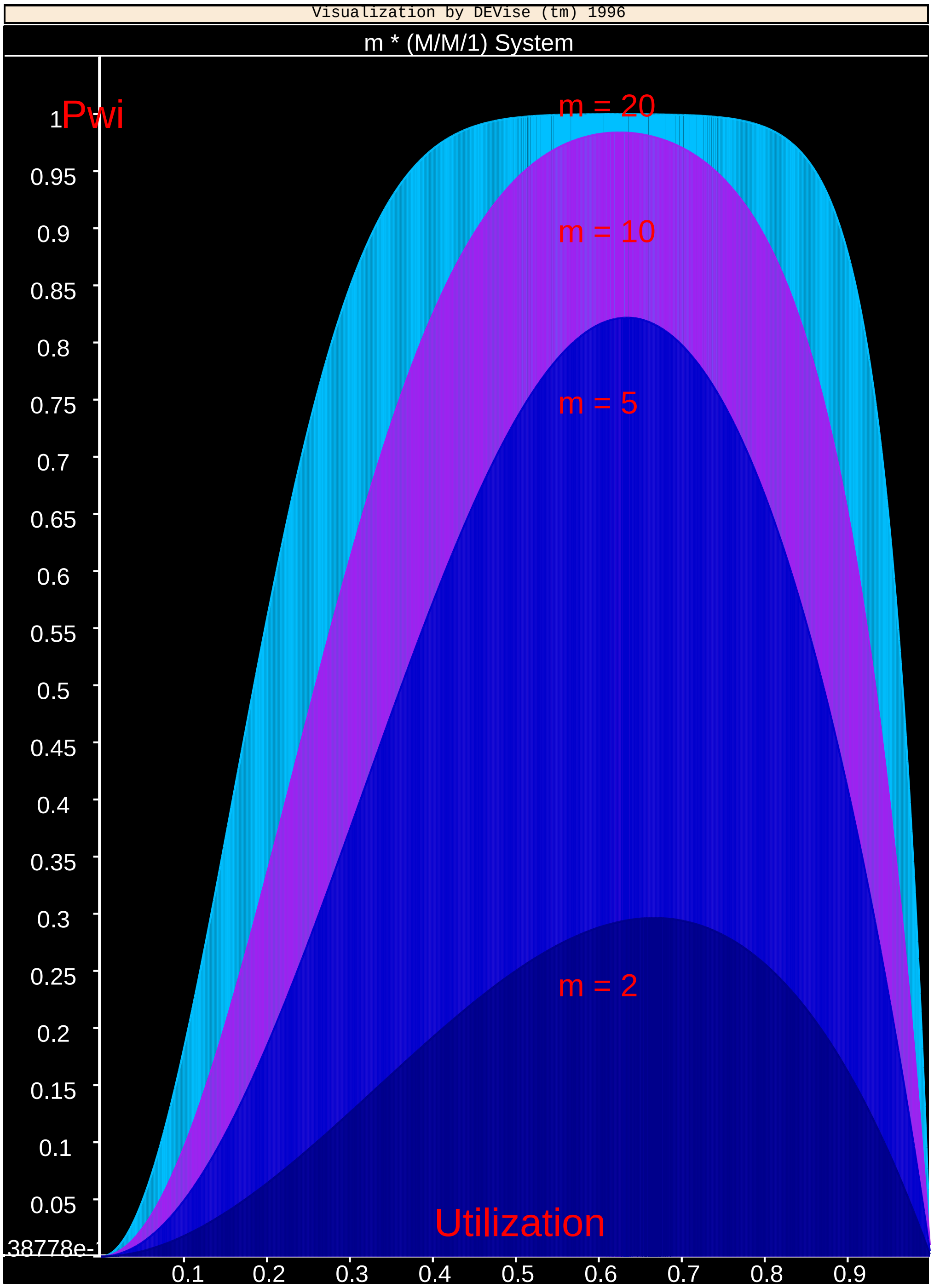
When a Poisson stream of customers with exponentially distributed service demands arrives at an 80% utilized server, (M/M/1 queuing system) a customer has to wait on the average 4 units of time per unit of service.

Wait while Idle (cont.)

When two such streams arrive each at an independent 80% utilized server (2*M/M/1 queuing system) almost 25% of the time a customer is waiting for service at one server while the other server is idle.

- For five streams and servers (5*M/M/1) it is 67%
- For ten streams and servers (10*M/M/1) it is 89%

Probaility of Wait while Idle



Who are HTC Customers?

- ***Inter-Job Parallelism***

  Tens to thousands of independent jobs.

  - Simulations (*Discrete Event*, *Monte Carlo*, *Continuous*)
  - Training of Neural Networks
  - Image Processing
  - Combinatorical problems

BATCH CUSTOMERS (cont.)

- *Intra-Job Parallelism*

  Each job consists of tens of processes and uses one of the many runtime support system to synchronize them. For example:

  - **PVM - uses messages for synchronization and exchange of data. Based on a dynamic model of parallel computing. Supports a heterogeneous environment.**

THE CHALLENGE

How to turn large existing distributively owned collections of *workstations*, *farms*, *multiprocessors*, and *supercomputers* into a High Throughput Computing Environment?

OBSTACLES

Ownership Distribution

Physical Distribution

Size and Uncertainties

Who are the players?

- Resource Owners
- System Administrators
- Application Developer
- Customers

Owner Prospective

- Owners support, key to high throughput computing
- Full control on when private resources are used by foreign users
- Full control on who can access private resources
- Full control on the impact of foreign usage on private quality of service

Owner Prospective (cont.)

- Easy to isolate
- Easy to monitor and understand what is going on
- Protection of private data/files

Contract between Owner and System Manager

Administrator Prospective

- No changes to existing software
- Easy to configure to meet local requirements
- Up to date information on what is going on in the system
- Easy to install and maintain

Developer Prospective

- Help with managing opportunistic resources
- Flexible and integrated Resource Management services
- Monitoring tools
- Preserve personal computation environment
- Few (if any) changes to application

Customer Prospective

- Steady source of large amounts of cycles via a single interface
- Reliable service
- Fair and known resource allocation policy

Customer Prospective(Cont.)

- Information on what is going on
- Control on costs
- Will be there tomorrow
- The key is, High Throughput

Mechanisms needed to support HTC

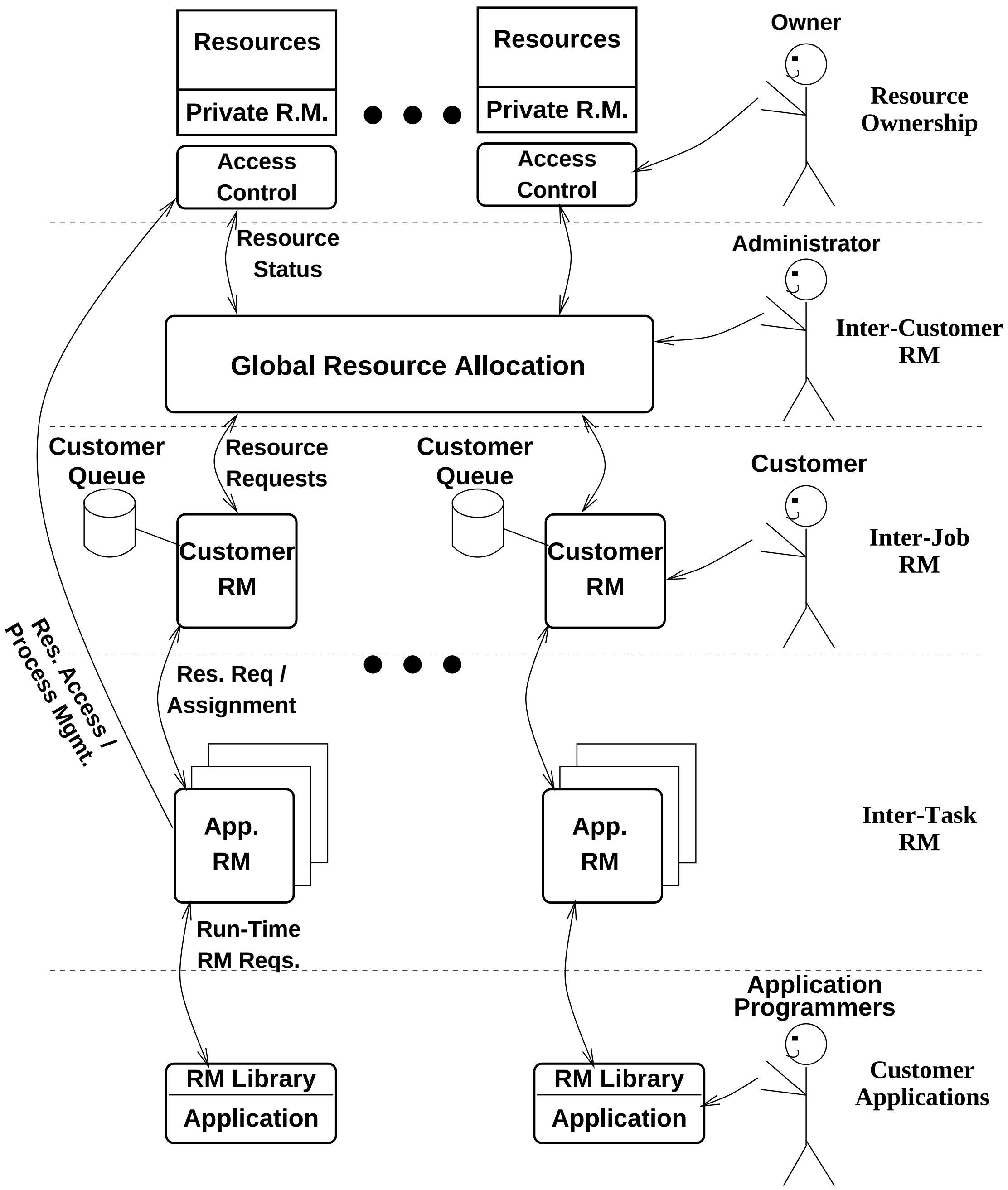
- Semantic free and flexible Matching services between Requests and Resources - Classified Adds
- Preemptive and resume scheduling - Checkpointing
- Remote access to data - Remote system calls
- Asynchronous Resource Management API - CARMI

Approach to HTC Resource Management

To meet all these requirements and expectations, a Layered approach to Resource Management Services is needed. Layers provide

- Modularity
- Extensibility
- Portability
- Adaptability

Layers of HTC RM



The Condor High Throughput System

- UNIX operating system
- Parallel Virtual Machine (PVM) parallel programming environment
- tcl/tk for visual interfaces
- Operational for a decade

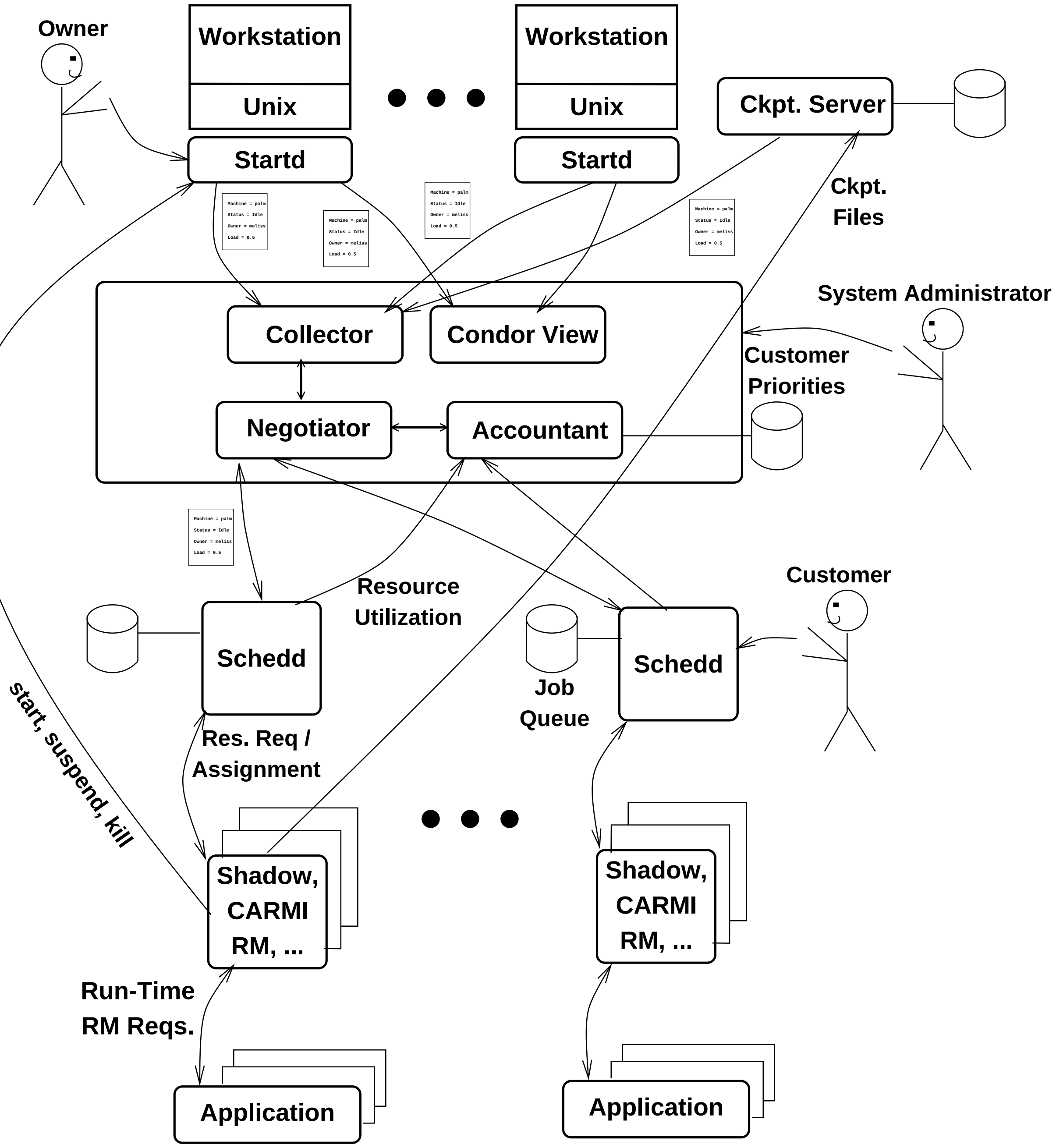
RM layers in Condor

- Owner - A process that advertises the resources and starts foreign jobs on private resources. Interface to OS.
- Matchmaker - Collection of processes that bring requests and resources together and provide accounting and storage (checkpoint files) services.

RM layers in Condor (Cont.)

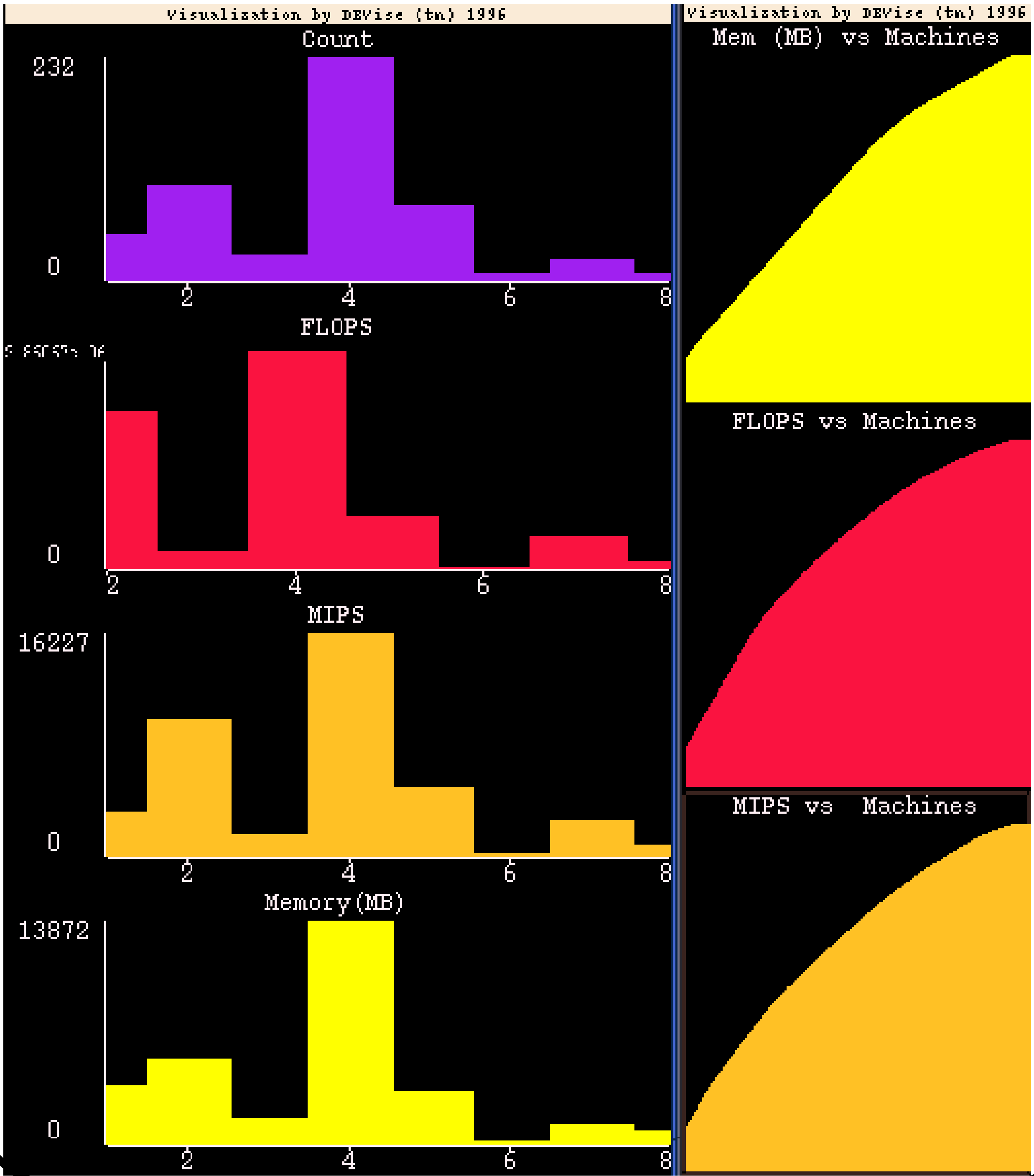
- Customer - A process that manages the job queue of the customer, and advertises requests.
- Application - Moves computation and data to allocated resources, manages allocated resources, provides interface between application and customer layer.
- High Level Application - Higher level resource management services.

Condor Layers

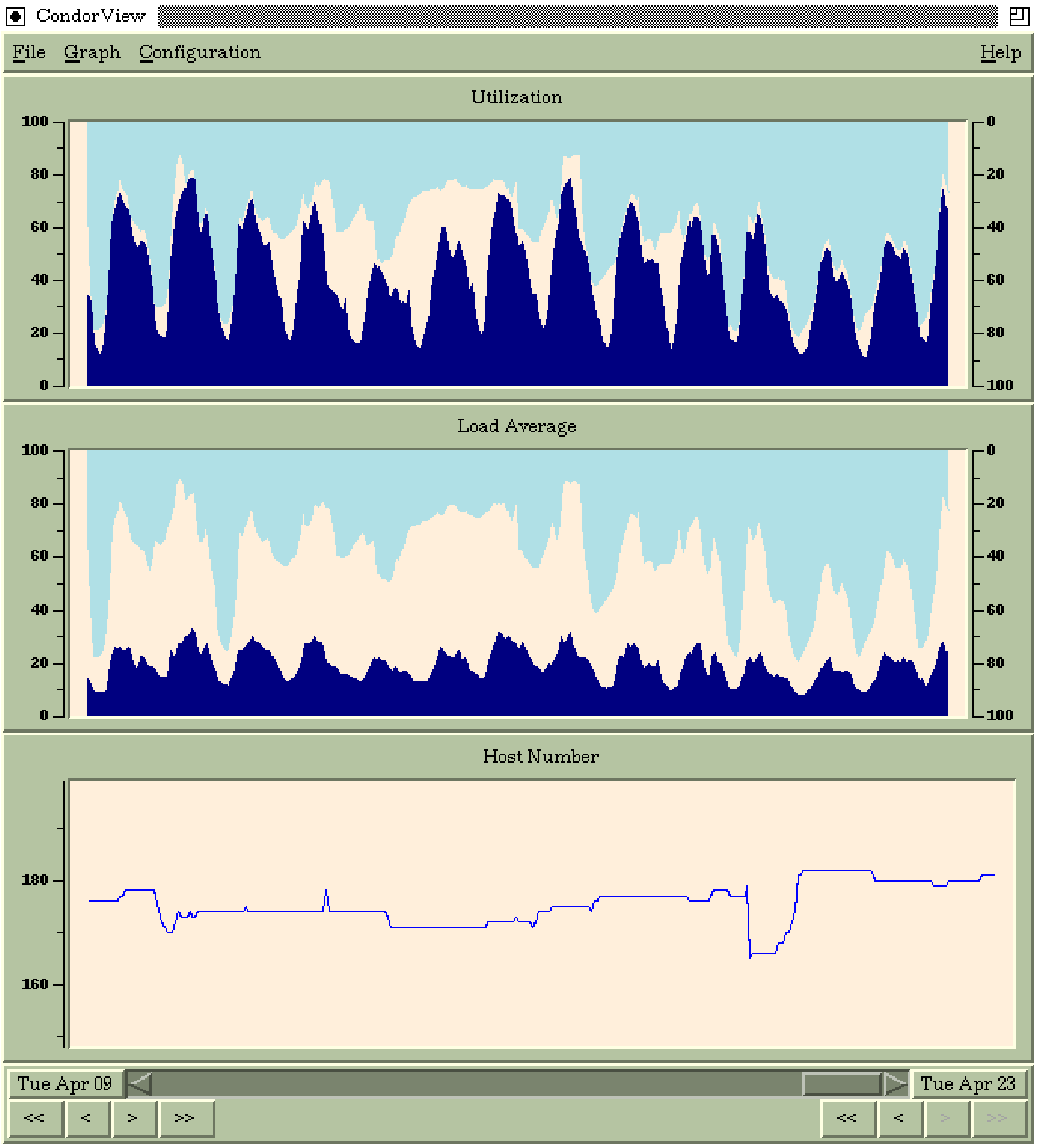


Samples from our Condor Pools

Our Condor Resources



Condor View

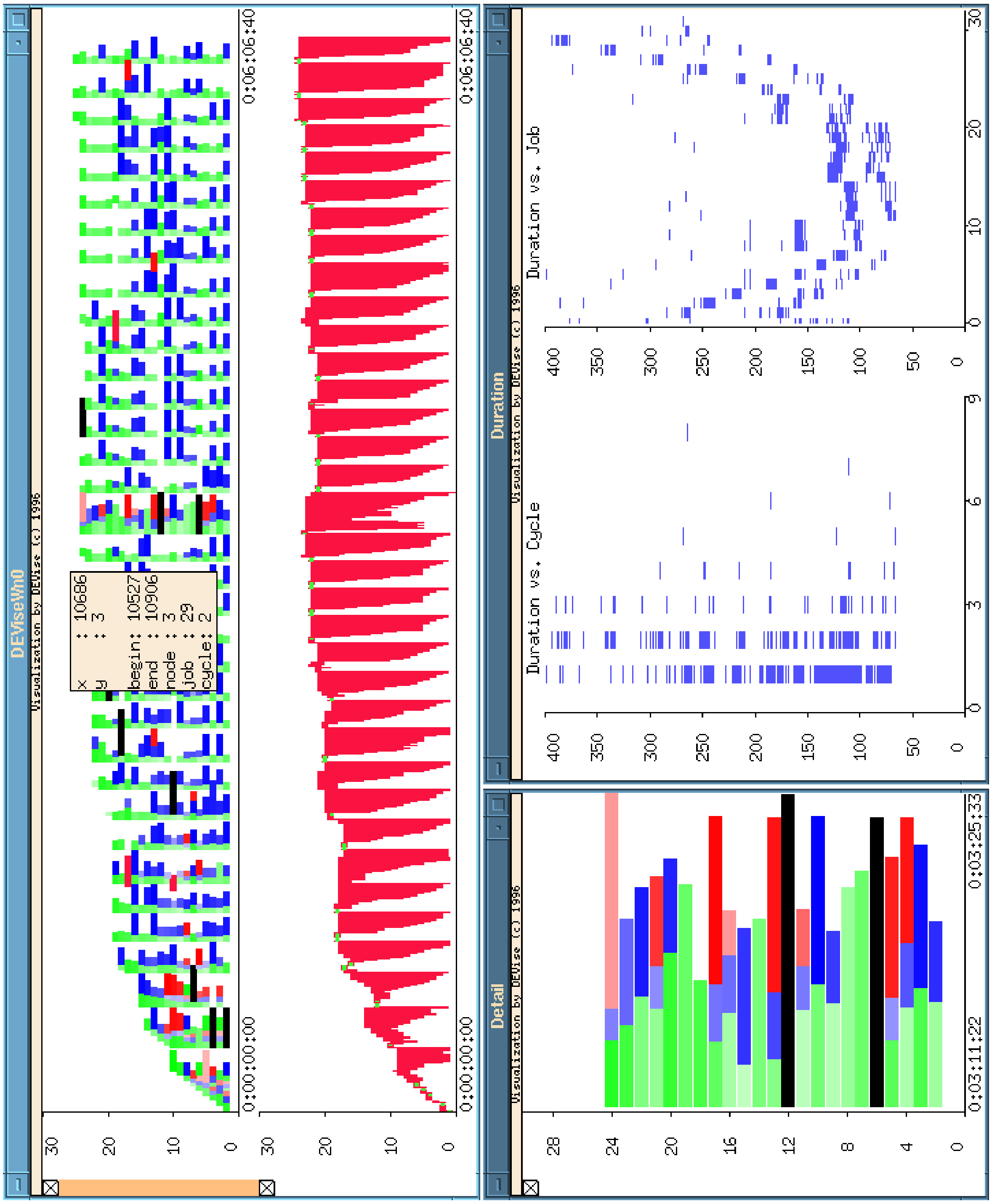

CondorView
File
Graph
Configuration
Help
Utilization
100
80
60
40
20
0
0
20
40
60
80
100
Load Average
100
80
60
40
20
0
0
20
40
60
80
100
Host Number
180
160
Tue Apr 09
Tue Apr 23
<<
<
>
>>

Material Sciences Application

```
potential = start
FOR iteration = 1 to 36
    FOR job = 1 to 31
   totalEnergy =+ Energy(job, potential)
    END
    potential = FunctionOf(totalEnergy)
END
```

Executed on Condor using PVM and WoDi Two traces - execution trace and performance trace

WoDi Execution

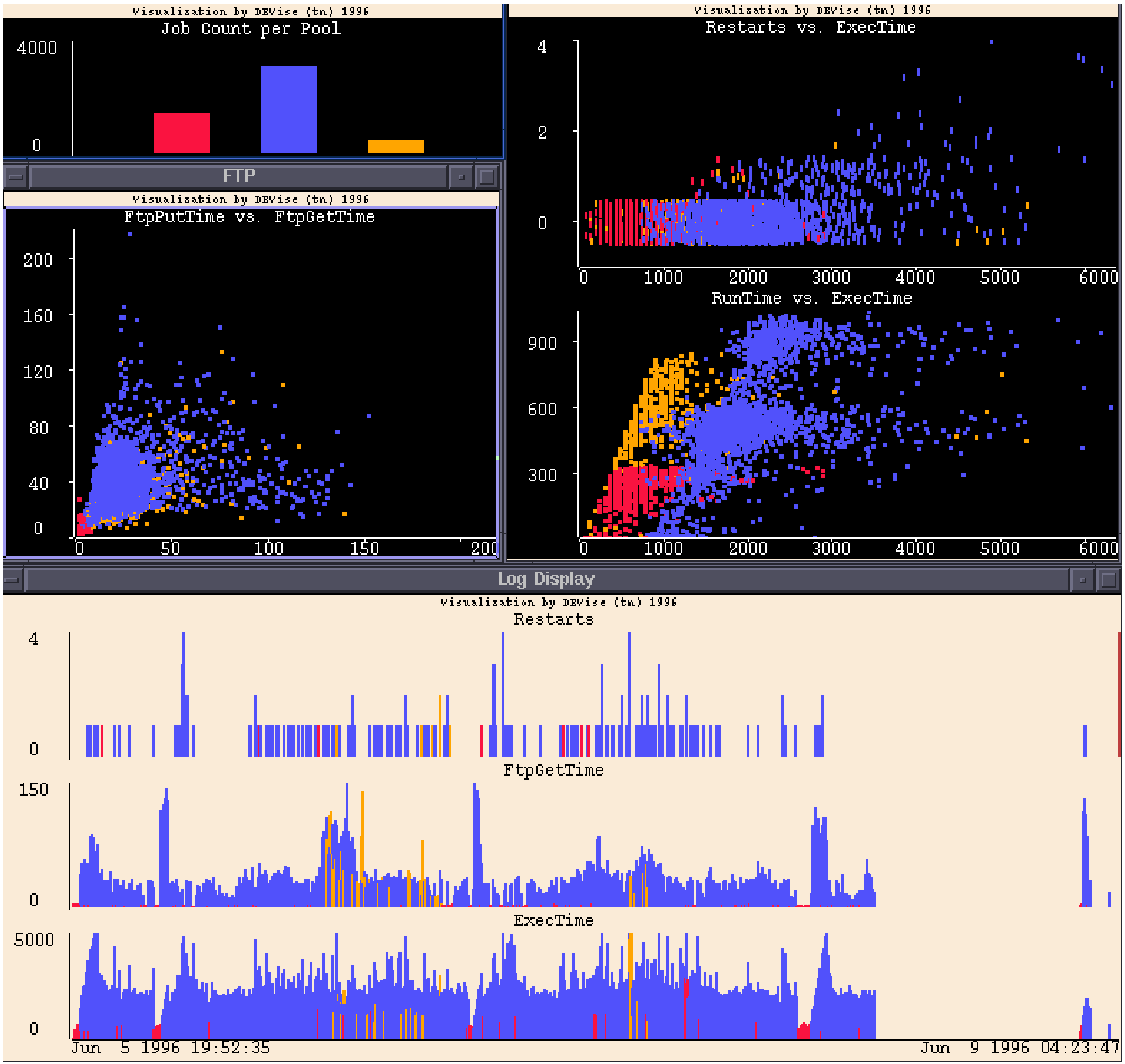


HTC on the Internet

Distributed Batch Controller (DBC) (joint effort with Ken Salam, supported by NASA)

- Controls several Condor pools
- Moves data to/from execution pool via the internet
- Used for the retrieval of atmospheric temperature and humidity profiles from the TOVS sensor system which has been on board of NOAA polar orbiting satellites since 1978.

5K DBC jobs


Visualization by DEVise (tm) 1996
Job Count per Pool
4000
0
FTP
Visualization by DEVise (tm) 1996
FtpPutTime vs. FtpGetTime
200
160
120
80
40
0
0
50
100
150
200
Visualization by DEVise (tm) 1996
Restarts vs. ExecTime
4
2
0
0
1000
2000
3000
4000
5000
6000
RunTime vs. ExecTime
900
600
300
0
1000
2000
3000
4000
5000
6000
Log Display
Visualization by DEVise (tm) 1996
Restarts
4
0
FtpGetTime
150
0
ExecTime
5000
0
Jun 5 1996 19:52:35
Jun 9 1996 04:23:47

Challenges Ahead

- Inter-pool Gateways - Condor Flocking
- New ports - Condor to NT!
- System configuration and monitoring tools
- I/O and storage scheduling
- Demand dependent resource allocation policies

The key to HTC is Effective Resource Management