Overview of
MOSIX2 for Clusters and Multi-Clusters

Prof. Amnon Barak
Department of Computer Science
The Hebrew University

http:// www . MOSIX . Org

January 2009

Background

Clusters, multi-clusters (intra-organizational Grids) and the emerging cloud tech are popular platforms for running demanding applications, e.g. HPC

Typically, users need to run multiple jobs with minimal burden how the resources are managed

- **Prefer not to:**
  - **Modify applications**
  - **Copy files or login to different nodes**
  - **Lose jobs when some nodes are disconnected**
- **Users don't know (and doesn't care):**
  - **What is the configuration, the status and the locations of the nodes**
  - **Availability of resources, e.g. CPU speed and load, free memory, etc.**

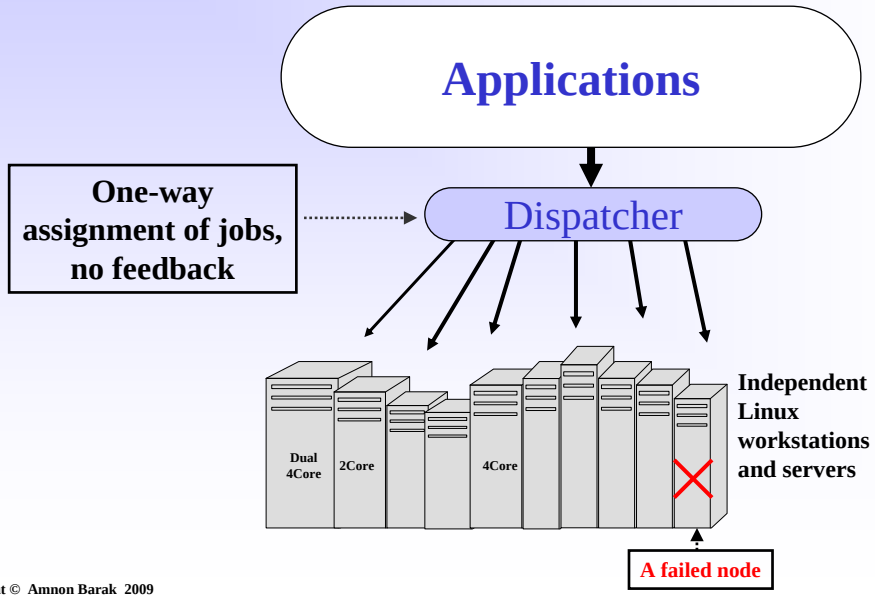
Traditional management packages

Most cluster management packages are batch dispatchers that place the burden of management on the users

For example, these packages:

- **Use static assignment of jobs to nodes**
  - **May lose jobs when nodes are disconnected**
- **Not transparent to applications**
  - **May require to link application with special libraries**
  - **View the cluster as a set of independent nodes**
    - **One user per node, cluster partition for multi-users**

Traditional management packages



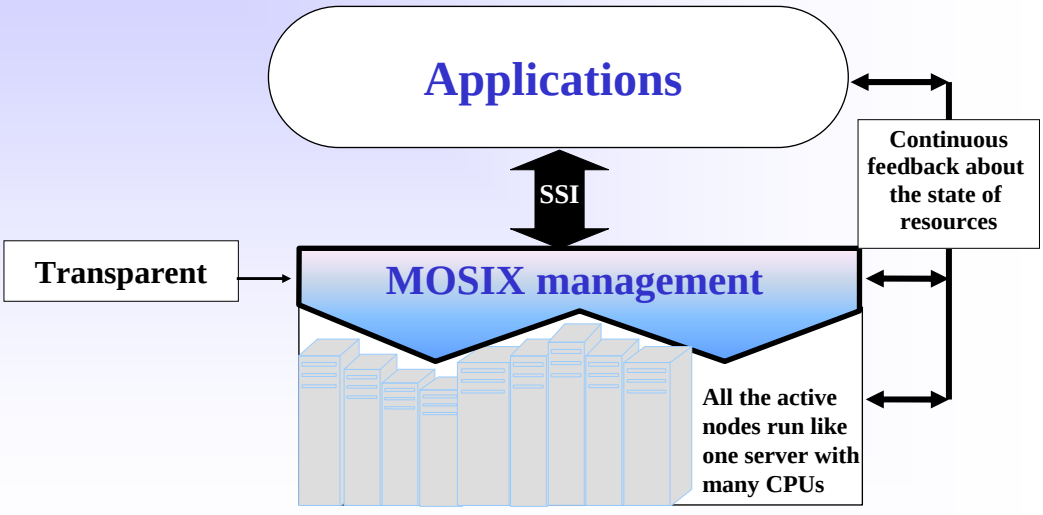
What is MOSIX (Multi-computer OS)

An operating system-like management system for distributed-memory architectures, e.g. cluster and multi-clusters

Main feature: Single-Systems Image (SSI)

- **Users can login on any node and need not know where their programs run**
- **Automatic resource discovery**
  - **Continuous monitoring of the state of the resources**
- **Dynamic workload distribution by process migration**
  - **Automatic load-balancing**
  - **Automatic migration from slower to faster nodes and from nodes that run out of free memory**

MOSIX is a unifying management layer



MOSIX Version 1

Can manage a single cluster

- **Main features:**
  - **Provides a SSI by process migration**
  - **Supports scalable file systems**
- **9 major releases developed for Unix, BSD, BDSI, Linux-2.2 and Linux-2.4**
- **Production installations since 1989**
- **Based on Linux since 1998**

MOSIX Version 2 (MOSIX2)

Can manage clusters and multi-clusters

- **Developed for Linux-2.6**
- **Geared for High Performance Computing (HPC), especially for application with moderate amounts of I/O**
- **Main features:**
  - **Provides a SSI by process migration**
  - **Process migration within a cluster and among different clusters**
  - **Secure run time environment (sandbox) for guest processes**
  - **Live queuing - queued jobs preserve their full generic Linux environment**
  - **Supports batch jobs, checkpoint and recovery**

Running applications in a MOSIX cluster

MOSIX recognizes 2 type of processes:

- **Linux processes - are not affected by MOSIX**
  - **Usually administrative tasks that are not suitable for migration**
  - **Processes that use features not supported by MOSIX, e.g. threads**
- **MOSIX processes -usually applications that can benefit from migration**
  - **All such processes are created by the ``mosrun''command**
  - **They are started from standard Linux executables, but run in an environment that allows each process to migrate from one node to another**
  - **Each MOSIX process has a unique home-node, which is usually the node in which the process was created**
- **Linux processes created by the ``mosrun -E'' command can still benefit from "MOSIX", e.g be assigned to the least loaded nodes**

Examples: running interactive jobs

Possible ways to run *myprog*:

> ***myprog*** **- run as a Linux process on the local node**

> **mosrun** ***myprog*** **- run as a MOSIX process in the local cluster**

> **mosrun -b** ***myprog*** **- assign the process to the least loaded node**

> **mosrun -b –m700** ***myprog*** **- assign the process only to a nodes with 700MB of free memory**

> **mosrun –E -b –m700** ***myprog*** **- run as a native Linux job**

> **mosrun –M -b –m700** ***myprog*** **- run a MOSIX job whose home node can be any node in the local cluster**

Running batch jobs

- To run 2000 instances of *myprog* on a multi-cluster

> mosrun –G –b –m700 –q –S64 myfile

- -G assign the job to a node in another cluster
- -S64 run up to 64 jobs at a time from the queue
- *myfile* a file with a list of 2000 jobs

How does it work

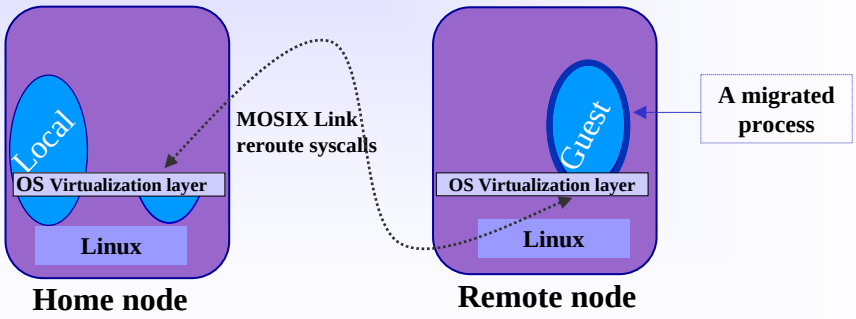
- Automatic resource discovery by a "gossip algorithm"
  - Provides each node with the latest info about the cluster/multi-cluster resources (e.g free nodes)
  - All the nodes disseminate information about relevant resources: speed, load, memory, local/remote I/O, IPC
  - Info exchanged in a random fashion - to support scalable configurations and overcome failures
  - Useful for high volume transaction processing
    - Example: a compilation farm - assign the next compilation to least loaded node

Dynamic workload distribution

- A set of algorithms that match between required and available resources
  - Geared to maximize the performance
  - Initial allocation of processes to the best available nodes in the user's private cluster
    - Not to nodes outside the private cluster
  - Multi-cluster-wide process migration
    - Automatic load-balancing
    - Automatic migration from slower to faster nodes
    - Authorized processes move to idle nodes in other clusters
- Outcome: users need not know the current state of the cluster and the multi-cluster resources

Core technologies

- Process migration – move the process context to a remote node
- OS virtualization layer allow migrated processes to run in remote nodes, away from their creation (home) nodes



The OS virtualization layer

- Provides the necessary support for migrated processes
  - By intercepting and forwarding most system-calls to the home node
- Result: migrated processes seem to be running in their respective home nodes
  - The user's home-node environment is preserved
  - No need to change applications, copy files or login to remote nodes or to link applications with any library
  - Migrated processes run in a sandbox

Outcome: users get the illusion of running on one node

- Drawback: increased communication and virtualization overheads
  - Reasonable vs. added cluster/multi-cluster services (see next slide)

Reasonable overhead:

Linux vs. migrated MOSIX process times (Sec.), 1Gbit-Ethernet

Application	RC	SW	JEL	BLAT
Local - Linux process	**723.4**	**627.9**	**601.2**	**611.6**
Total I/O (MB)	0	90	206	476
Migrated process- same cluster	**725.7**	**637.1**	**608.2**	**620.1**
slowdown	0.32%	1.47%	1.16%	1.39%
Migrated process across	**727.0**	**639.5**	**608.3**	**621.8**
1Km campus slowdown	0.5%	1.85%	1.18%	1.67%

Sample applications:

RC = CPU-bound job
JEL = electron motion
SW = proteins sequences
BLAT = protein alignments

Multi-cluster management

- Main new features of MOSIX2 for Linux-2.6
  - Process migration among different clusters
  - Priorities among different clusters
  - Live queuing
  - Supports disruptive configurations
  - Supports batch jobs, checkpoint and recovery

Administrating a multi-cluster

A federation of x86 (both 32-bit and 64-bit) clusters, servers and workstations whose owners wish to cooperate from time to time

- Collectively administrated
  - Each owner maintains its private cluster
    - Determine the priorities vs. other clusters
  - Clusters can join or leave the multi-cluster at any time
  - Dynamic partition of nodes to private virtual clusters
  - Users of a group access the multi-cluster via their private clusters and workstations

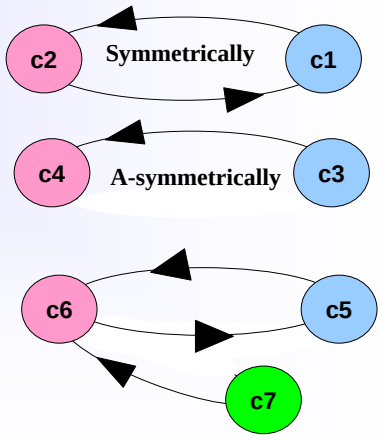
Outcome: each cluster and the multi-cluster performs like a single computer with multiple processors

Why an " intra- organizational" Grid: due to trust

The priority scheme

- Cluster owners can assign priorities to processes from other clusters
  - Local and higher priority processes force out lower priority processes
- Pairs of clusters could be shared, symmetrically(C1-C2) or asymmetrically(C3-C4)
- A cluster could be shared (C6) among other clusters (C5, C7) or blocked for migration from other clusters (C7)
- Dynamic partitions of nodes to private virtual clusters

Outcome: flexible use of nodes in shared clusters



When priorities are needed

- Scenario 1: one cluster, some users run many jobs, depriving other users from their fair share
- Solution: partition the cluster to several sub-clusters and allow each user to login to only one sub-cluster
  - Users in each sub-cluster can still benefit from idle nodes in the other sub-clusters
  - Processes of local users (in each sub-cluster) has higher priority over guest processes from other sub-clusters
- Scenario 2: some users run long jobs while other user need to run (from time to time) short jobs
- Scenario 3: several groups using a shared cluster
  - Sysadmin can assign different priorities to each group

Scheduling and monitoring

- **Batch jobs – run as Linux processes in different nodes**
- **Checkpoint & recovery - time basis, manually or by the program**
- **Live queuing – queued jobs maintain an organic connection with their Unix environment**
  - **Queue management provides means for tracing jobs, changing priorities, order of execution, for running parallel e.g. MPI jobs**
  - **Queued jobs are released gradually in a manner that prevents flooding the local cluster or other clusters**
- **Built-in on-line monitor for the local cluster resources**
- **On-line web monitor of the multi-cluster and each cluster**
  - **http://www.mosix.org/webmon**

Example: queuing

- **With the -q flag, *mosrun* places the job in a queue**
- **Jobs from all the nodes in each cluster share one queue**
  - **Queue policy: first-come-first-serve, with several exceptions**
  - **Users can assign priorities to their jobs, using the *–q{pri}* option**
    - **The lower the value of *pre* the higher priority**
    - **The default priority is 50. It can be changed by the sysadmin**
    - **Running jobs with pri < 50 should be coordinated with the cluster's manager**
  - **Out-of-order and fair-share**
    - **These options allow to instantly start a fix number of jobs per user, overriding the queue**
- **Examples:**

  > mosrun –q –b –m1000 myprog **(queue a MOSIX program to run in the cluster)**

  > mosrun –q60 –G –b -J1 myprog **(queue a low priority job to run in a different cluster)**

  >mosrun –q30 –E –m500 myprog **(queue a high priority batch job)**

mosq – view and control the queue

- *mosq list* – list the jobs waiting in the queue
- *mosq listall* – list jobs already running from the queue and jobs waiting in the queue
- *Mosq delete {pid}* – delete a waiting job from the queue
- *Mosq run {pid}* – run a waiting process now
- *Mosq cngpri {newpri}{pid}* – *change the priority of a waiting job*
- *Mosq advance {pid}* – *move a waiting job to the head of its priority group within the queue*
- *Mosq retard {pid}* – *move a waiting job to the end of its priority group within the queue*

More options in the mosq manual

Disruptive configurations

When a cluster is disconnected:

- All guest processes move out
  - To available remote nodes or to the home cluster
- All migrated processes from that cluster move back
  - Returning processes are frozen (image stored) on disks
  - Frozen processes are reactivated gradually

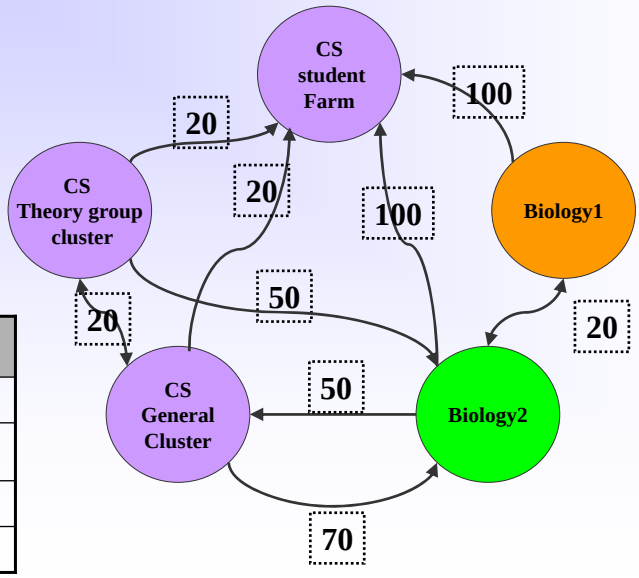
Outcome:

- Long running processes are preserved
- No overloading of nodes

Hebrew University multi-cluster campus Grid (HUGI)

- 15 production MOSIX clusters ~430 nodes, ~650 CPUs
  - In life-sciences, med-school, chemistry and computer science
  - Target: 2000 nodes (mostly Windows machines in student labs)
- Sample applications that our users are running:
  - Nano-technology
  - Molecular dynamics
  - Protein folding, Genomics (BLAT, SW)
  - Weather forecasting
  - Navier-Stokes equations and turbulence (CFD)
  - CPU simulator of new hardware design (SimpleScalar)

Priorities among HUGI clusters



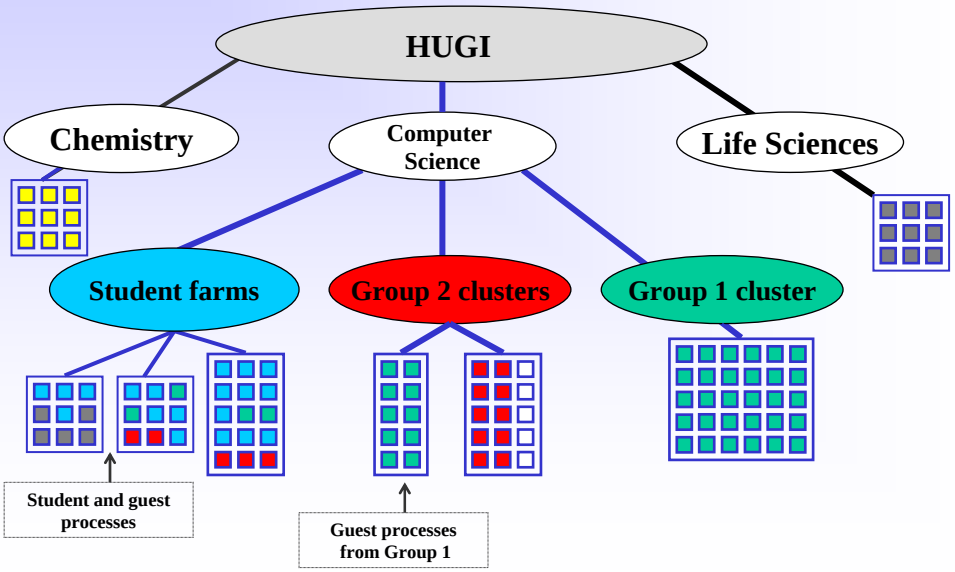
CS general Priority for Accepting processes

From cluster	Priority
Theory	20
Student Farm	Blocked
Biology1	Blocked
Biology2	50

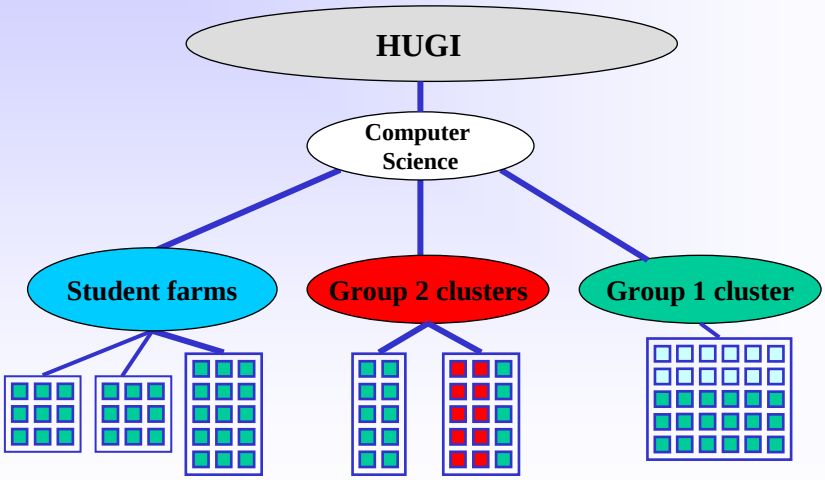
Biology 2 Priority for accepting processes

From cluster	Priority
Theory	50
Student Farm	Blocked
CS General	70
Biology1	20

Day use: idle shared nodes allocated to users



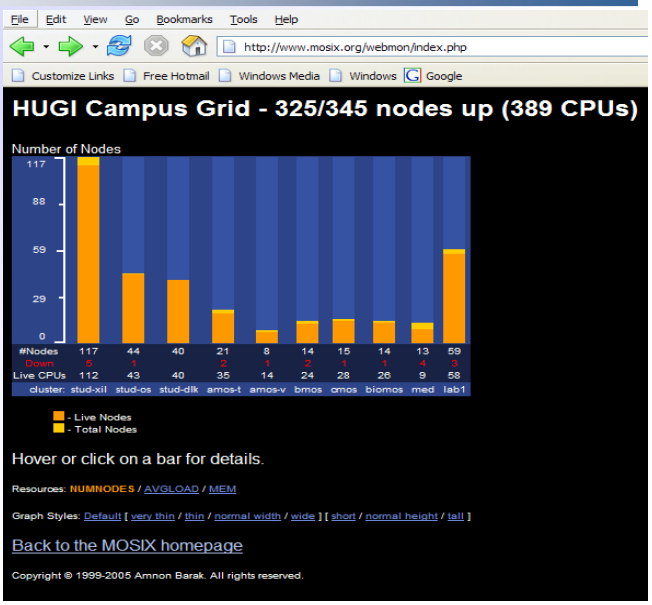
Night use: most nodes are allocated to one group



Web monitor: www.MOSIX.org/webmon

Display:

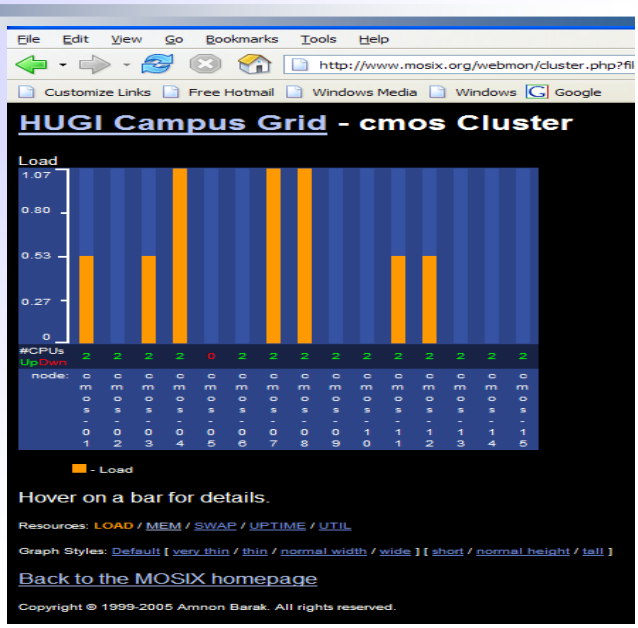
- Total number of nodes/CPUs
- Number of nodes in each cluster
- Average load



Zooming on each cluster

Display:

- Load
- Free/used memory
- Swap space
- Uptime
- Users



Conclusions

MOSIX2 is a comprehensive set of tools for automatic management of Linux clusters and multi-clusters

- **Self-management algorithms for dynamic allocation of system-wide resources**
  - **Cross clusters performance nearly identical to a cluster**
- **Many supporting tools for ease of use**
  - **Includes an installation script and manuals**
- **Can run in native mode or on top of virtual machine packages, e.g. VMware, Xen, MS Virtual Server, over an unmodified OS (LINUX, Windows, OS X)**

How to obtain a copy of MOSIX

- **A free, unlimited trial copy is provided to faculty, staff and researchers for use in academic, research and non-profit organizations**
- **A free, limited evaluation copy is provided for non-profit use**
- **Non-academics copies are available**

Details at **http://www.MOSIX.org**