The MOSIX Scalable Cluster Computing for Linux

Prof. Amnon Barak

Computer Science

Hebrew University

http://www. mosix.org

Presentation overview

- Part I : Why computing clusters (slide 3 - 7)
- Part II : What is MOSIX (8 - 20)
- Part III: Parallel file systems (21 - 26)
- Part IV: Tools (27 - 28)
- Part V : Experience /parallel applications (29 - 32)
- Part VI: Summary/current/future projects (33 - 36)

Color code: Highlight GOOD properties (blue)
Not so good properties (Red)

Computing Clusters are not just for High Performance Computing (HPC)

- Many organizations need powerful systems to run "demanding applications" and for high availability. Some examples:
  - Internet: web servers, ISP, ASP, GRID computing
  - Telephony: scalable switches, billing systems
  - Image/signal proc: DVD, rendering, movies
  - Databases: OLTP, decision support, data warehousing
  - Commercial applications, financial modeling
  - RT, FT, Intensive I/O jobs, large compilations, . . .

Outcome

- Low cost Computing Clusters are replacing traditional super-computers and mainframes, because they can provide good solutions for many demanding applications
- Computing clusters are especially suitable for small and medium organizations that can not afford expensive systems
  - Made of commodity components
  - Gradual growth - tuned to budget and needs

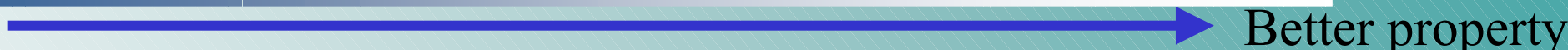
Advantage and disadvantage of Clusters

- Advantage: the ability to do parallel processing
  - Increased performance and higher availability
- Disadvantage: more complex to use and maintain
- Question: is an SMP (NUMA) a good alternative ?
- Answer: by classification of parallel systems using 4 criteria:
  - Ease (complexity) of use, overall performance (resource utilization), cost (affordability), scalability

2 alternatives: SMP vs. CC

- SMP (NUMA): easy to use, all the processors works in harmony, efficient multiprocessing and IPC, supports shared memory, good resource allocation, transparent to application, expensive when scaled up
- Clusters: low-cost, unlimited scalability, more difficult to use: insufficient "bonds" between nodes - each node works independently, many OS services are locally confined, limited provisions for remote services, inefficient resource utilization, may require modifications to applications

SMP vs. Clusters



Ease of use

SMP/NUMA — excellent

Cluster — inconvenient

Utilization

SMP / NUMA — excellent

Cluster — tedious

Cost

SMP / NUMA — high

Cluster — excellent

Scalability

SMP / NUMA — limited

Cluster — unlimited

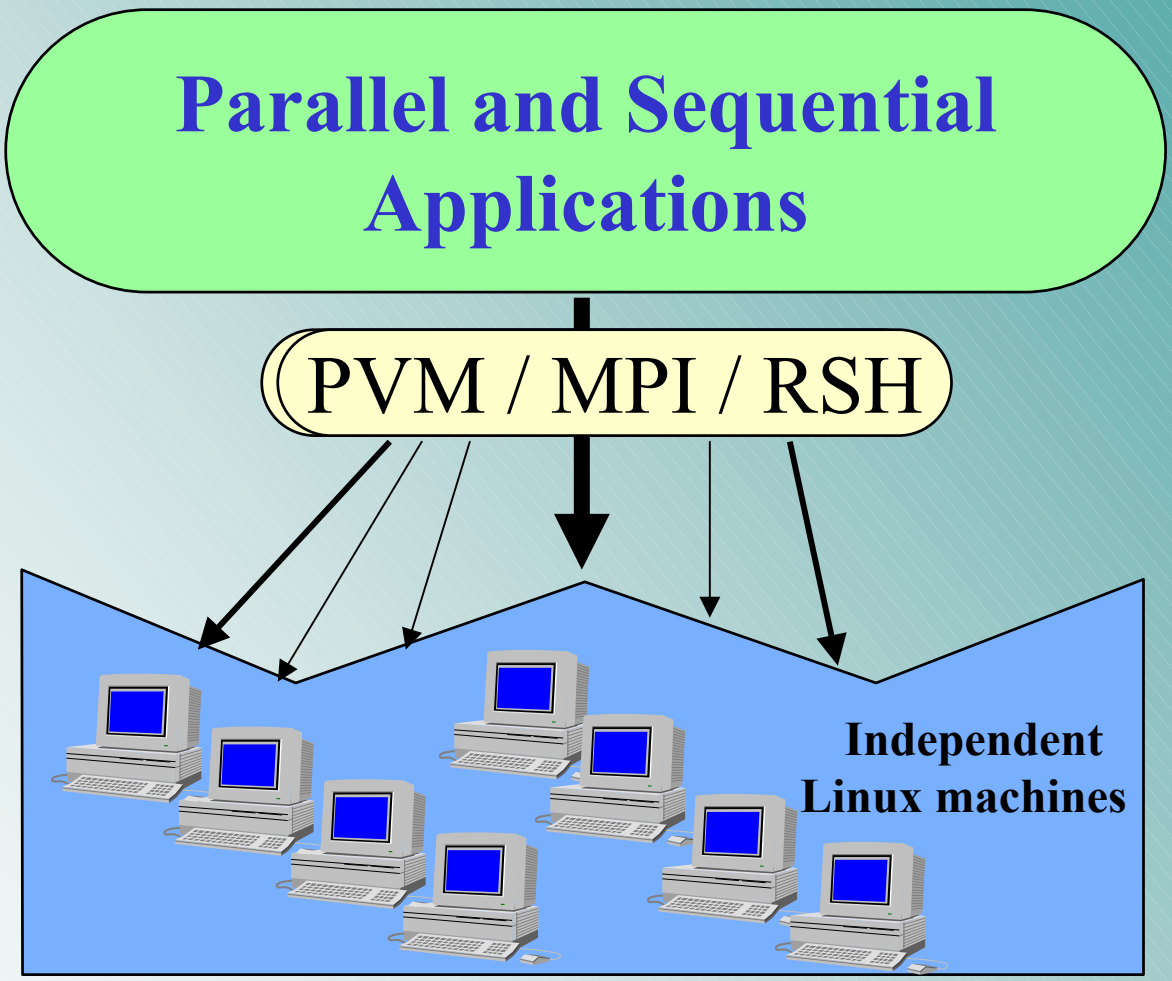
Goal of MOSIX

Make cluster computing as efficient and easy to use as an SMP

Without MOSIX: user level control

Not transparent to applications

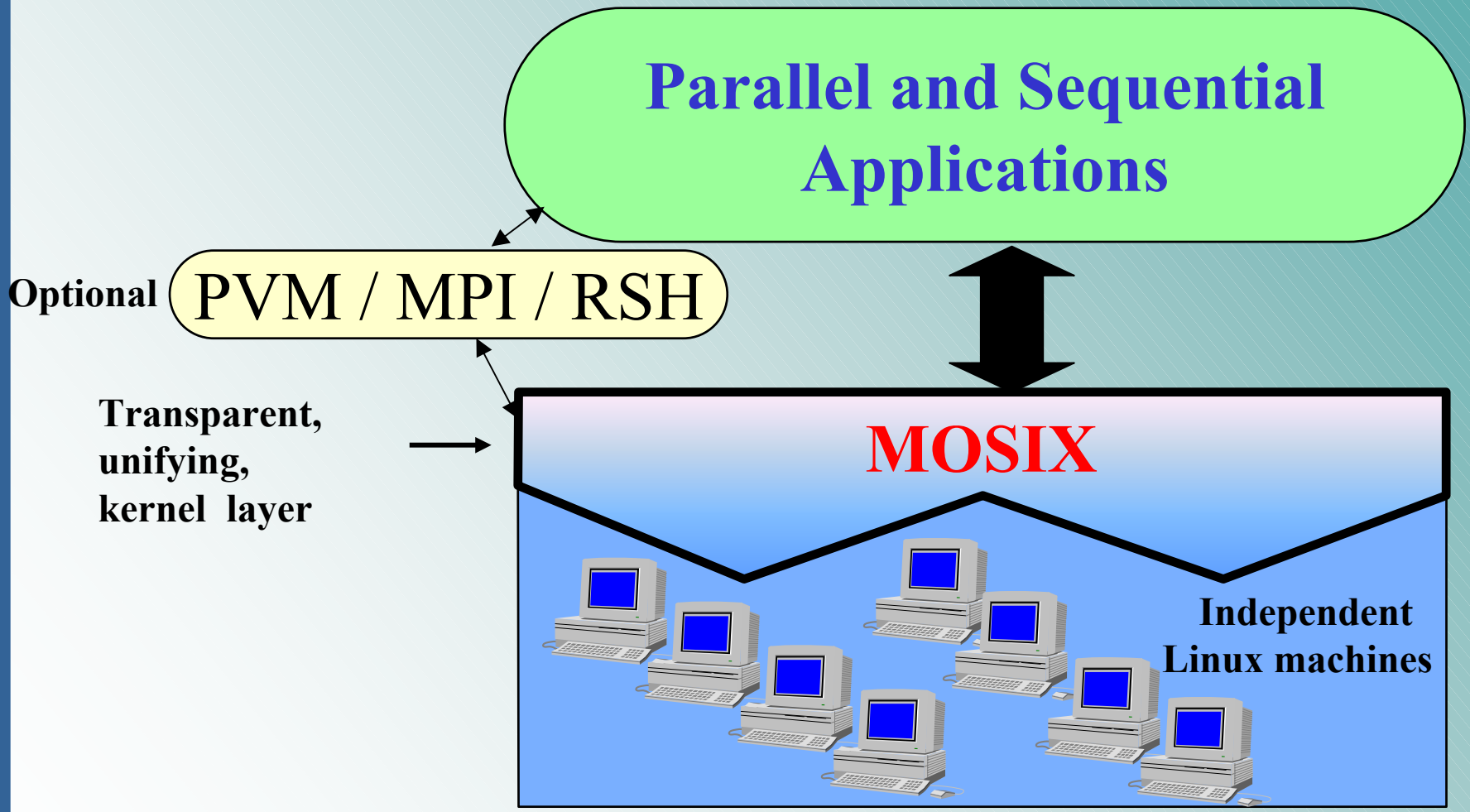
Rigid management lower performance



What is MOSIX

- A kernel layer that pool together the cluster-wide resources to provide user's processes with the illusion of running on one big machine
- Model: Single System (like an SMP)
  - Ease of use - transparent to applications
    - no need to modify applications
  - Maximal overall performance - adaptive resource management, load-balancing by process migration
  - Overhead-free scalability (unlike SMP)

MOSIX is a unifying kernel layer



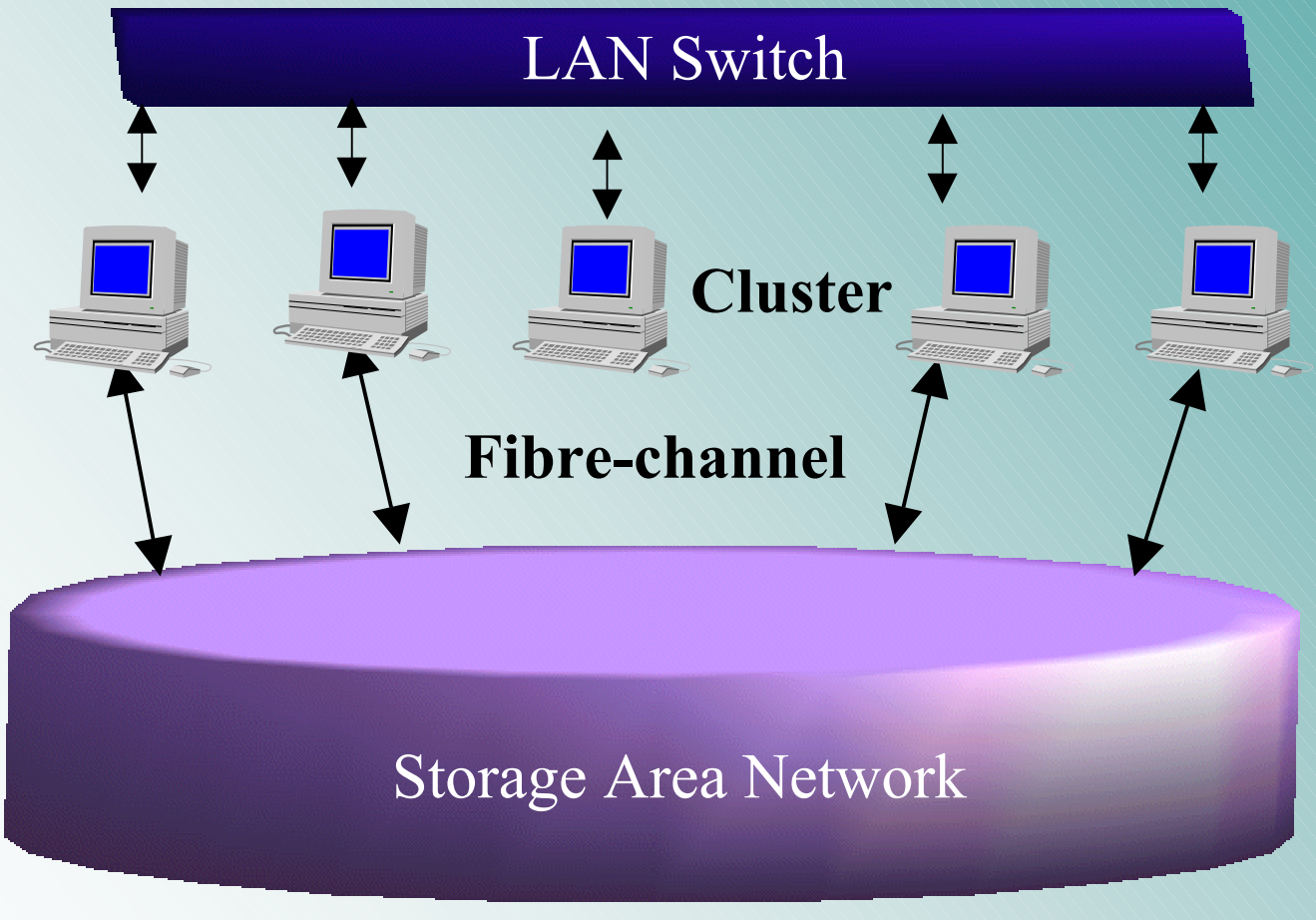
A two tier technology

1. **Information gathering and dissemination**
   - Provides each node with sufficient cluster information
   - Support scalable configurations
   - Fixed overhead: 16 or 6000 nodes
2. **Preemptive process migration**
   - Can migrate any process, anywhere, anytime
   - Transparent to applications - no change to user interface
   - Supervised by adaptive algorithms that respond to global resource availability

Tier 1: Information exchange

- All the nodes gather and disseminate information about relevant resources: CPU speed, load, free memory, local/remote I/O, IPC
- Info exchanged in a random fashion (to support scalable configurations and overcome failures)
- Applicable to high volume transaction processing
  - Scalable web servers, telephony, billing
  - Scalable storage area cluster (SAN + Cluster) for intensive, parallel access to shared files

Storage Area Cluster



Problem: how to preserve file consistency

Example: parallel "make" at EMC^2

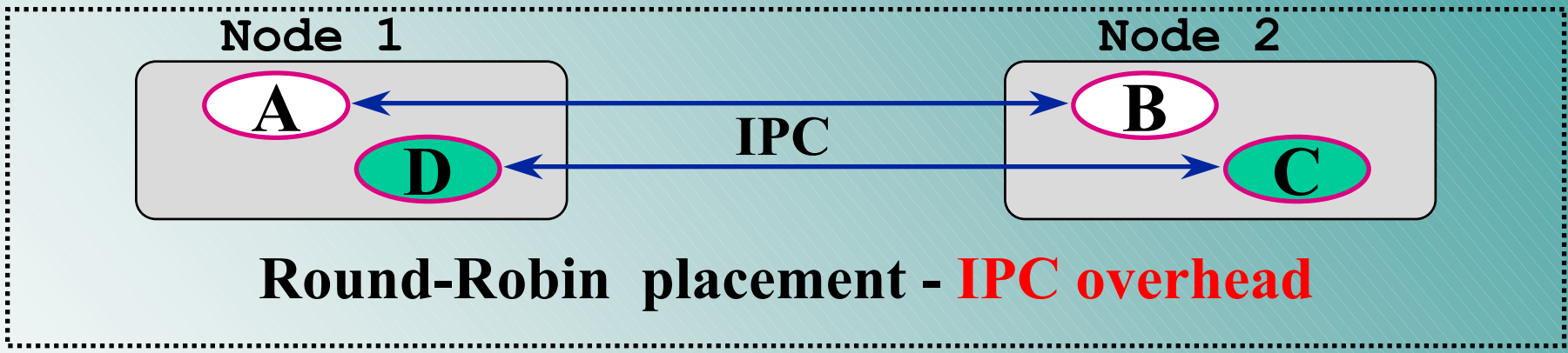
- Assign the next compilation to least loaded node
- A cluster of ~320 CPUs (4-way Xeon nodes)
- Runs 100-150 "build", with millions lines of code concurrently
  - Serial time ~30 hours, cluster time ~3 hours
  - Much better performance and unlimited scalability vs. a commercial package, for less cost

Tier 2: Process migration for

- **Load balancing:** to improve the overall performance
  - Responds to uneven load distribution
- **Memory ushering:** to prevent disk paging
  - Migrate processes when RAM exhausted
- **Parallel file operations**
  - Bring the process to the data

- Speed of migration (Fast Ethernet): **88.8 Mb/Sec**

Intelligent load-balancing



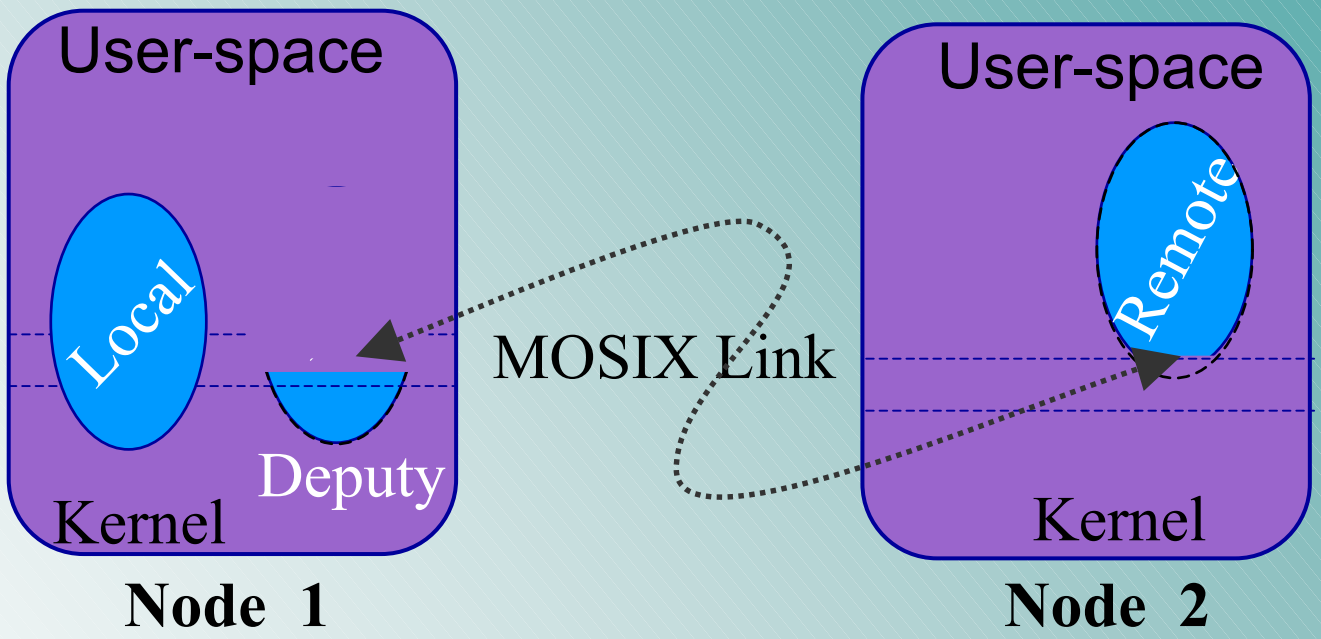
Round-Robin placement - IPC overhead

Optimal assignment - No IPC overhead

Single system image model

- Users can "login" to any node in the cluster. This node is the "home-node" for all the user's processes
  - Migrated processes always seem to run at the home node, e.g., "ps" show all your processes, even if they run elsewhere
- Migrated processes use local resources (at the remote nodes), while interact with the home-node to access their environment, e.g. perform I/O
- Drawback: extra overhead for file and I/O operations

Splitting the Linux process



- Process context (Remote) is site independent - may migrate
- System context (deputy) is site dependent - must stay at "home"
- Connected by an exclusive link for both synchronous (syscalls) and asynchronous (signals, MOSIX events)

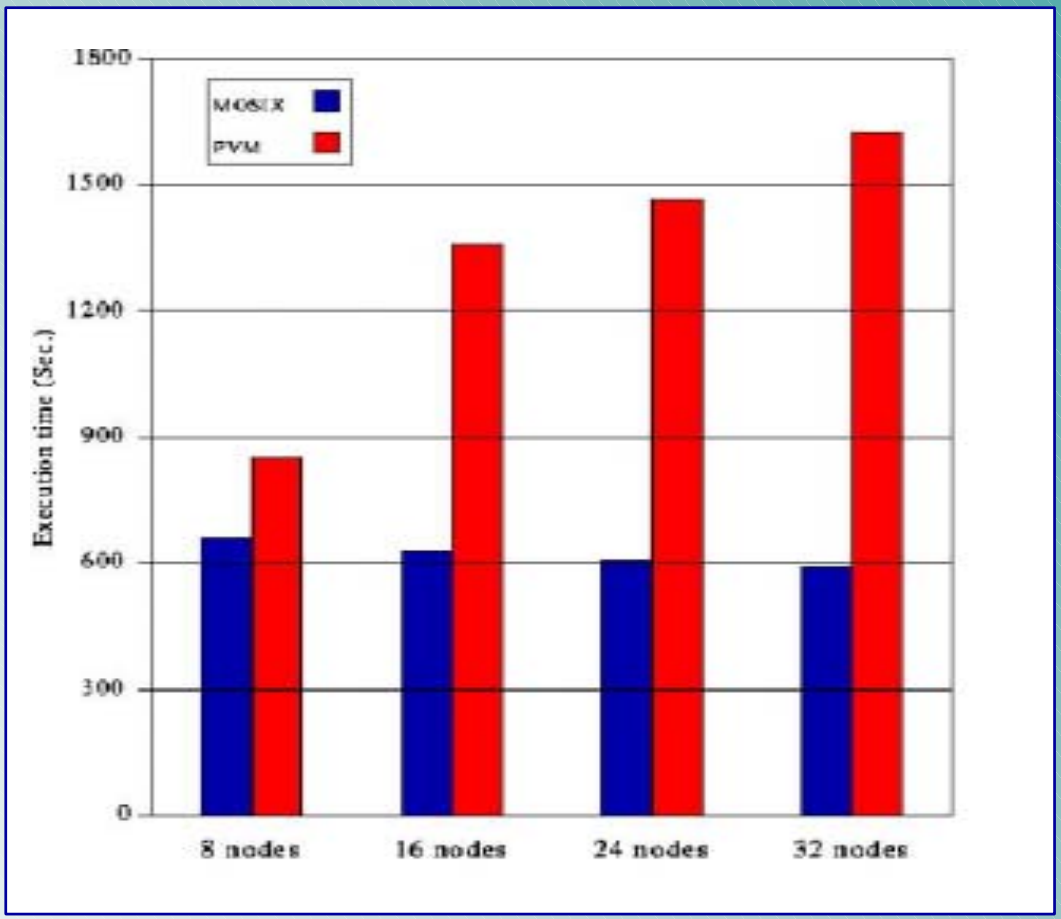
Example: MOSIX vs. PVM

Fixed number of processes per node

Random process size with average 8MB

MOSIX is scalable

PVM is not scalable



Direct File System Access (DFSA)

- I/O access through the home node incurs high overhead
- Direct file System Access (DFSA) allow processes to perform file operations (directly) in the current node - not via the home node
- Available operations: all common file and I/O system-calls on conforming file systems
- Conforming FS: GFS, MOSIX File System (MFS)

DFSA Requirements

- The FS (and symbolic-links) are identically mounted on the same-named mount-points (/mfs in all nodes)
- File consistency: completed operation in one node is seen in any other node
  - Required because a MOSIX process may perform consecutive syscalls from different nodes
- Time-stamp consistency: if file A is modified after B, A time stamp must be greater than B's time stamp

The MOSIX File System (MFS)

- Provides a unified view of all files and all mounted FSs on all nodes, as if they were within a single FS
- Makes all directories and regular files throughout a MOSIX cluster available from all the nodes
- Provides file consistency from different nodes by maintaining one cache at the server (disk) node
- Parallel file access by process migrate

Global File System (GFS) with DFSA

- Same as MFS, with local cache over the cluster using a unique locking mechanism
- GFS + process migration combine the advantages of load-balancing with direct disk access from any node - for parallel file operations
- GFS for Linux 2.2 includes support for DFSA
- Not written yet for Linux 2.4

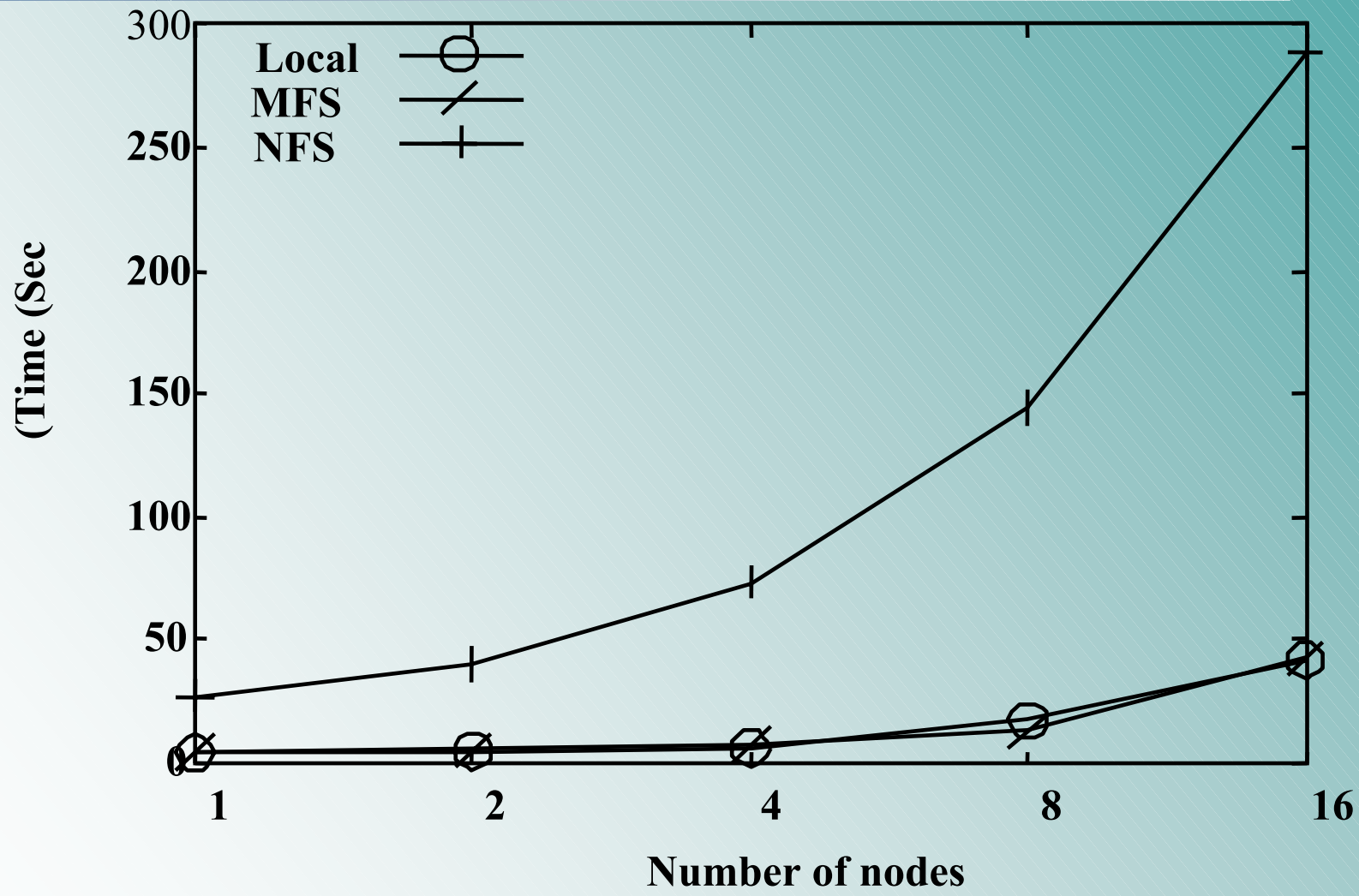
Postmark (heavy FS load)
Client - Server performance

Access Method	*Data Transfer Block Size*						
	64B	512B	1KB	2KB	4KB	8KB	16KB
Local (in the server)	102.6	102.1	100.0	102.2	100.2	100.2	101.0
MFS	104.8	104.0	103.9	104.1	104.9	105.5	104.4
NFS	184.3	169.1	158.0	161.3	156.0	159.5	157.5

* All numbers in seconds

Parallel Read from a Quad server

(4K block size)

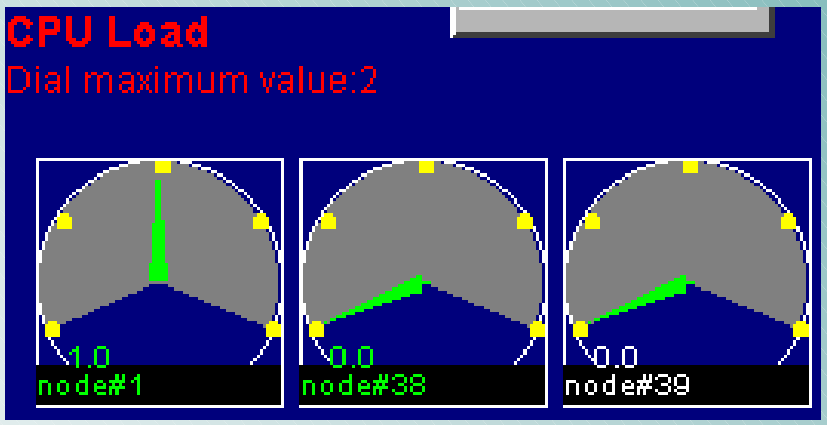


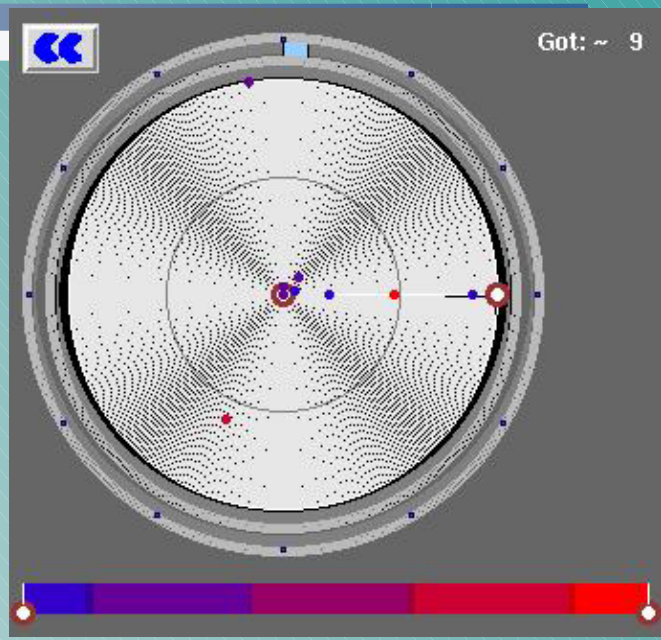
Tools

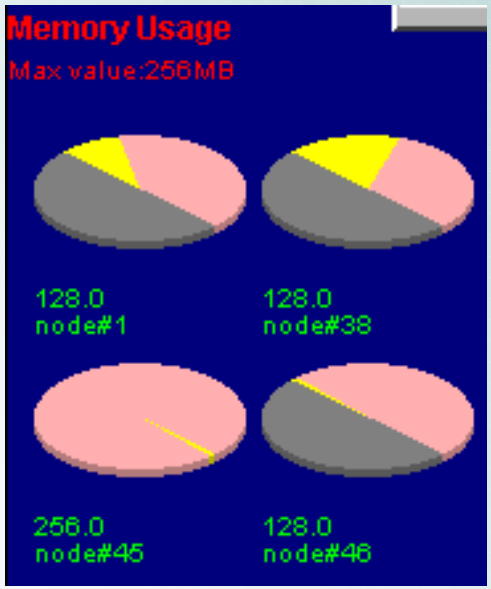
- **Linux kernel debugger** - on-line access to the kernel memory, processes properties, stack, etc.
- **Cluster installation, administration and partition**
- **Kernel monitor (mon)** - displays load, speed, total and used memory and CPU utilizatio
- **JAVA monitor** - display cluster properties
  - CPU load, speed, utilization, memory utilization, CPU temp., fan speed
  - Access via the Internet

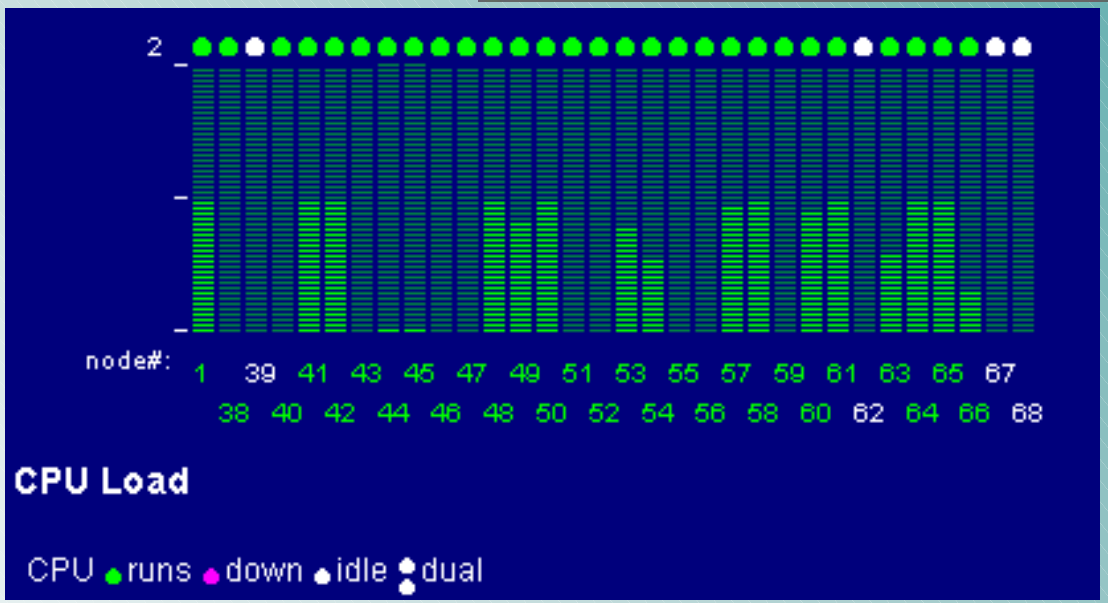
Java Monitor

* Leds
* Dials
* Radar
- Pie
- Bars
- Matrix
- graphs
- Fan
- Tables…









MOSIX is best for

- **Multi-user platforms - where many users share the cluster resources**
- **High performance demanding applications**
- **Parallel applications - HPC**
- **Non-uniform clusters - with different speed machines, different memory sizes, etc.**

API and implementation

- API: no new system-calls - all done via /proc
- MOSIX 0.9 for Linux 2.2 (since May 1999):
  - 80 new files (40K lines), 109 files modified (7K lines)
  - 3K lines in load-balancing algorithms
- MOSIX 1. for Linux 2.4 (since May 2001):
  - 45 new files (35K lines), 124 files modified (4K lines)
  - 48 user-level files (10K lines)

Our “scalable” configuration

Examples of HPC Applications

- Programming environments: PVM, MPI
- Examples:
  - Protein classification - 20 days 32 nodes
  - High energy molecular dynamics
  - Quantum dynamics simulations - 100 days non-stop
  - Weather forecasting (MM5)
  - Computational fluid dynamics (CFD)
  - Car crash simulations (parallel Autodyn)

Summary

- MOSIX brings a new dimension to Cluster and GRID Computing
  - Ease of use - like an SMP
  - Overhead-free scalability
  - Near optimal performance
  - May be used over LAN and WAN (GRID)
- Production quality -
  - Large production clusters installed

Ongoing projects

- DSM - strict consistency model
- High availability - recovery model
- Migratable sockets - for IPC optimization
- Scalable web server - cluster IP
- Network (cluster) RAM - bring process to data
- Parallel File System

Future projects*

- **Port MOSIX to:**
  - **New platforms, e.g. Itanium, PPC**
  - **Other OS, e.g. FreeBSD, Mac OS X**
- **Shared disk file system, e.g. GFS**
- **Parallel DBMS, e.g. billing systems**
- **Tune to specific parallel applications, e.g. rendering**

* subject to external funding

The MOSIX group

- Small R&D group - 7 people
- Over 20 years of experience in Unix kernel
  - Development of 7 generations of MOSIX
- We like to work with a large company on R&D of cluster technology - not only in Linux
- Possible directions: HPC, telephony, high volume transaction/web servers, rendering, movies, etc.

MOSIX

Thank you for your attention

Questions ?

For further details please contact:

amnon@cs.huji.ac.il