Presentation overview

- Cluster Computing
- Motivation / other systems
- What is MOSIX ?
- Resource sharing in MOSIX
- Preemptive process migration
- Implementation and API
- Conclusions and future work

2

Scalable Cluster Computing with MOSIX for Linux

Ariel Rosenblatt
arielr@cs.huji.ac.il
http://www.mosix.cs.huji.ac.il

Basic concepts - HW

- UP - Uni-Processor
- SMP - Symmetric Multi Processing
- CC - Cluster Computing

4

Parallel computing

- **The need** - demanding applications:
  - simulations, signal/image processing, cryptography, monte-carlo, wheather forecast, alien life (SETI)
- **The technology** -
  - divide the computation to sub units
  - use multiple CPUs concurrently

3

Cluster Computing technology

- **Computing Clusters** (CC) are made of standard PC's, fast LANs and standard software
- **CC advantages over other systems:**
  - Affordability (commodity hardware)
  - Size scalability
  - Performance scalability
  - High availability

6

Software tools

- **Shared Memory**
  - threads / pthreads
  - classic IPC, memory mapping
- **Message Passing**
  - sockets
  - remote shell + pipe
  - PVM - parallel virtual machine
  - MPI - message passing interface

5

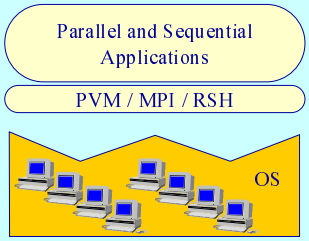
CC's are flexible

CC's can support:

- **Multi-user** & **time-sharing** environments
- Different **CPU speeds** & **memory sizes**
- **Varying resource** availability
- **Many kinds** of **unpredictable** processes

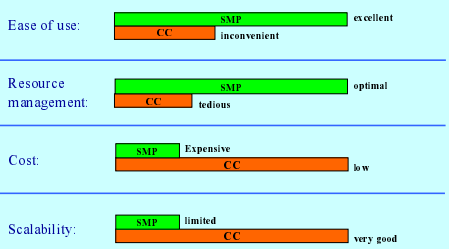
8

A typical environment



7

On {CC,SMP}'s...



10

CC's are not SMP's

- **SMP:** provides **optimal** resource management
  - All the CPUs work in **harmony**
- **CC:** insufficient **"bonds"** between nodes
  - Many OS services are locally confined
  - Each node operated independently
  - Remote services explicitly requested

9

Example: Beowulf

- A network driver (developed at NASA) that provides channel-bonding to increase bandwidth
- Packaged in a Linux distribution with compilers, PVM and MPI.
- Easy installation of a standalone cluster
- Mostly, User Level implementation

12

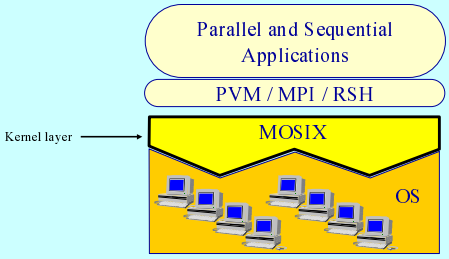
MOSIX: toward SMP services on a CC

- *Pools together* the *Cluster-wide* resources, for:
  - Better overall **resource utilization**
  - **improved performance** by adaptiveness
  - **Ease of use** - almost like SMP
  - **Scalable configuration**

Result: ***HPC at PC prices !***

11

The architecture



14

The Multicomputer OS for UNIX

- A kernel layer that provides the applications with the illusion of working on a single system
- Transparent to the application
- User interface remains unchanged
- Remote operations & resources are used like local ones

13

User-level link

16

The missing link

15

Process migration can be used for...

- **Load balancing:** use available CPU cycles to improve the performance
- **Memory sharing:** use idle memory to prevent disk paging
- **I/O optimization:** move I/O processes to target nodes
- **IPC optimization:** allow communicating processes to benefit from shared memory
- **Network RAM:** span big processes across nodes, bring the code to the data

18

The MOSIX technology

Preemptive process migration that can migrate any process, anywhere, anytime

- Supervised by adaptive algorithms that respond to global resource availability
- Transparently - for ease of use and coherence
- Supports scalable configurations

17

Information dissemination

Information dissemination: distribute enough information for the resource sharing algorithms

Each node -

- Calculate resource indices (cpu load, memory load, io activity, free slots in tables)
- Periodically sends its resources indices to a randomly chosen subset of the cluster
- Maintains a small sliding window to the world

20

Global resource sharing

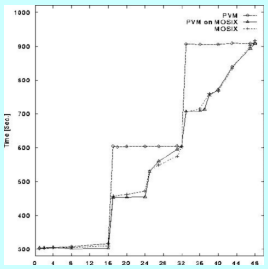
- Operate in a dynamic environment (on line)
- Support general and unpredictable applications
- Base decision on partial knowledge
- Required distributed control
- Regard scalability properties

19

PVM vs. MOSIX

Static allocation vs. process migration

16 nodes
1-48 processes



22

Load balancing

- Load matching: reduce variance between pairs of nodes by migrating processes from over-loaded to less-loaded nodes
- Tunable to CPU and LAN speeds
  CPU speed measured at boot time, LAN speed by "tune"
- Competitive, adaptive allocation
  Outperforms any static allocation
- Near optimal sustained performance

21

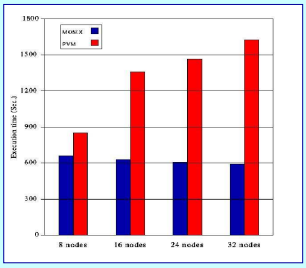
PVM vs. MOSIX

Static allocation vs. process migration

Fixed number of processes per node

Average process size: 8MB

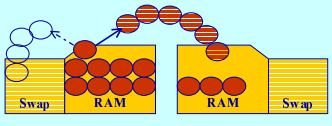
Note the performance scalability !



24

Memory ushering

- Algorithm: migrate processes from a node that exhausted its memory, to prevent paging
- Overrides the load balancing activities
- Useful in clusters with non-uniform memory or applications with varying sizes



23

HUJI's scalable configuration

26

Scalability issues

- Symmetry - no client/server
- Autonomy - decentralized control
- Probabilistic algorithms, partial knowledge
- Migration overhead independent of cluster size

⇒ Same overhead for 16 nodes and 256 nodes !

- Vs. SMP - contention for locks grows

25

System image model

- Each process has a Unique Home Node (UHN)
- Home machine model - migrated processes always seem to run at the home node
  Example: 'ps' show all the processes, even if they run elsewhere
- Processes use local resources (at the remote site) and interact with the UHN to access their environment

28

Porting MOSIX to Linux

Phase 1: On-line kernel debugger

Phase 2: Transparent operation of split processes
Communication layer & interaction protocol

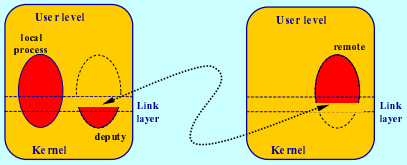
Phase 3: Process migration - *migdaemon*

Phase 4: Information - *infodaemon*
Load balancing, memory ushering

Phase 5: Interface via /proc, user utilities

27

Deputy / Remote interaction



- Local processes are unaffected
- Deputy/Remote connected over an exclusive link
  - Synchronous mode: system calls
  - Asynchronous mode: signals, MOSIX events

30

Splitting the process

- The Linux process is split:
  Process context (code, stack, data) - site independent
  System context (environment) - site dependent
- Process context called ***Remote*** - may migrate
- System context called ***Deputy*** - confined to the UHN
  - The *Deputy* encapsulates the process when running in kernel

29

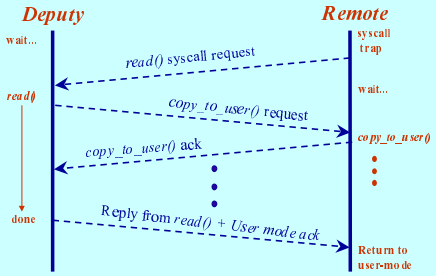
Logical process state

- A process may need to run while logically stopped or asleep, to handle:
  - Migration requests, or process-data requests
  - Configuration changes

  ⇒ sleepwalker processes - run these while asleep...
- Maintain two process states:
  - Immediate state - as seen by the kernel scheduler
  - Logical state - as seen by other processes

32

Example: *read()* system-call



31

The API

- The /proc interface includes:
  - Synch/asynch migration requests
  - Locking/unlocking against automatic migrations
  - System setup, administration
  - Information about other nodes
  - *remote* (guest) processes

34

The MOSIX API

User interface via /proc file-system:

- /proc/mosix
  - /proc/mosix/admin
  - /proc/mosix/decay
  - /proc/mosix/info
  - /proc/mosix/nodes/nnnn/...
  - /proc/mosix/remote/pppp/...

33

MOSIX for Linux

- ~85 Linux kernel files modified
  New code is in: #ifdef CONFIG_MOSIX
- ~20,000 new source lines in: mos/... & include/mos/...
  2,000 lines in loadable load-balancing module
- User utilities:
  'mon' - monitoring resource usage
  'mosctl' - administrative tool
  'mosrun' - manually control programs execution
  'tune' - tune LAN & CPU parameters

36

API examples

- "echo 6 > /proc/$$/migrate"
- "cat /proc/$$/where"
- "echo 1 > /proc/mosix/admin/expel"
- "cat /proc/mosix/nodes/22/load"
- "cat /proc/mosix/remote/653/from"

- The "mos" library enables easy use of the API
- Automatic process migration can be disabled

  ⇒ API allows user programs to control migrations

35

Experience with MOSIX

- **Multi-user platform -**

  An 8 node cluster is used at HU for over 2 years, serving all CS student coursework over 2 years

  650 students !

- **High-performance -**

  Our 100 node cluster is used for executing large scientific/engineering applications

  100 days non-stop executions !

38

Summary

- MOSIX brings a new dimension to Cluster Computing with Linux:
  - Ease of use
  - Overhead-free scalability
  - Bridge towards SMP services on a CC

37

MOSIX is best for

- Multi-user platforms - many users share the resources
- High performance systems for demanding applications
- Scalable web servers e.g. on-line stock market analyzer
- Parallel processing - for supercomputing
- Non-uniform clusters - with different speeds/memories

40

MOSIX for Linux at HUJI

- **Large configuration**:
  - 50 Pentium-II 300 MHz
  - 34 Pentium-II 400 MHz (dual CPU)
  - Over 17 GB cluster wide RAM
  - Over 100 processors
- Myrinet (2.56 Gb/s) & Fast Ethernet LANs
- Runs **SuSE** and **RedHat**, based on **Kernel 2.2.12**
- Upgrade: HW with Intel, SW with Linux kernel

39

Parallel Applications on MOSIX

- Programming environments: PVM, MPI
- Demanding applications:
  - Protein classification
  - Molecular dynamics
  - Weather forecasting
  - Computational fluid dynamics
  - Car crash numerical simulations
  - Image/signal processing

42

On-going projects

- MOSIX development:
  - Migratable sockets
  - Parallel File System support - **DFSA**
  - Ports to other architectures (subject to external funding)
- Open projects:
  - User level utilities
  - Testing / adjusting to other Linux distributions

41

Molecular Dynamics

- Desorbing atom spatial distribution simulations
- Requires dynamic load balancing because:
  - Unpredictable, large number of cases
  - Time for each case varies
  - Unknown workload and memory sizes
- Highlight: was very inconvenient on the SP2
- Joint work with Ashkenazy and Kelzon (Physics, TAU)

44

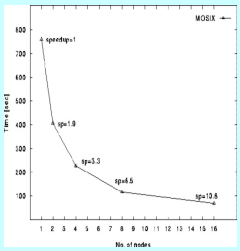
Protein Sequences

- 544,000 protein segments were clustered
- Clustering profile reveals new relationships between protein families
- Highlight: program was executed for 20 days on a 32-node MOSIX, without interference with other users (e.g. students)
- Joint work with Lilial (Biology, HU), Linial, Tishby & Yona (CS, HU)

43

Quantum Dynamics Simulations

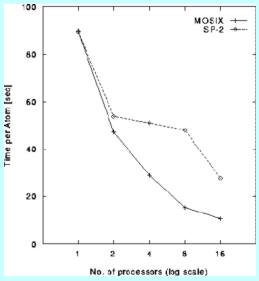
- Dynamic properties of large poly-atomic systems
- 3D quantum wave-functions of 3400 atoms
- Joint work with Fredj, Broude & Gerber (Chemistry, Irvine & HU)



46

High Energy Molecular Dynamics

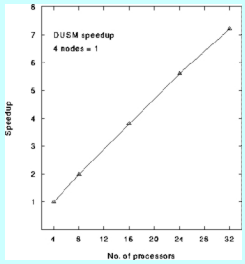
- Simulations of an energetic atom impacting a surface
- Many (>10e6) atoms and many time steps
- Non local potentials (like N-Body)



45

Drexel Univ. Shell Model

- Shell model calculations of the nucleus spectra and state transitions
- Memory consuming (large matrices):
- 6,700 x 6,700 ~120MB
- 50,800 x 50,800 ~4GB
- 200K x 200K ~80GB
- Joint work with Novoselsky (HU) & Vallieres (Drexel U)



47