OpenMOSIX: High performance Linux farm

Rosario Esposito [Rosario.Esposito@na.infn.it]
INFN-Napoli

Cosa è OpenMOSIX ?

OpenMOSIX è un è un'estensione (patch) del kernel di Linux che aggiunge funzionalità avanzate di clustering tra macchine di architettura x86.

I processi in esecuzione vengono assegnati e riassegnati dinamicamente ai nodi del cluster in modo da ottenere in ogni momento il massimo vantaggio dalle risorse disponibili

Storia del progetto OpenMOSIX

- MOSIX (Multicomputer OS for UnIX) nasce nei primi anni 80 su PDP-11/70 all'università di Gerusalemme
- Prima implementazione su BSD/pdp
- Implementazione su VAX 11/780
- Implementazione su Motorola / bus VME (contratto con la Israeli Defence Forces)
- 1994 versione per BSDi
- 1997 GNU + Linux
- Ha contribuito a dozzine di patch al kernel standard di Linux
- Novembre 2001 Split MOSIX / OpenMOSIX

Caratteristiche di OpenMOSIX (1/9)

Network transparency

I nodi del cluster, che comunicano attraverso una LAN efficiente, si comportano come una singola macchina SMP.

- Le applicazioni non devono essere modificate o linkate a una particolare libreria
- L'utente ha l'impressione di interagire con un solo computer
- La scalabilità è lineare rispetto al numero di nodi del cluster

Caratteristiche di OpenMOSIX
(2/9)

Preemptive process migration

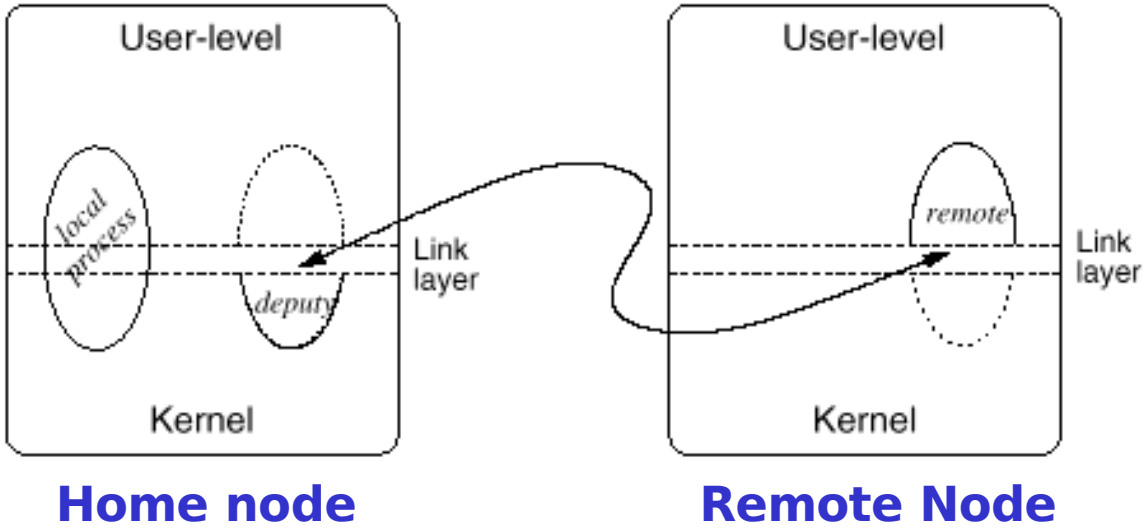
Ogni processo utente, in qualsiasi momento e in maniera trasparente può migrare verso un nodo disponibile.

I processi sono costituiti da 2 parti:

- system context (deputy) che è legato al nodo in cui è stato creato e non può migrare
- user context (remote) che può migrare su qualsiasi nodo del cluster

Caratteristiche di OpenMOSIX
(3/9)

Preemptive process migration



Caratteristiche di OpenMOSIX
(4/9)

Preemptive process migration: limiti

Alcuni processi non possono essere divisi in deputy e remote. La loro esecuzione è confinata al nodo in cui sono stati creati.

In particolare NON possono migrare i processi che fanno uso di:

- Writeable Shared Memory
- Sockets
- Real-Time scheduling

Caratteristiche di OpenMOSIX (5/9)

Dynamic load balancing

- Viene realizzato con la migrazione dei processi verso i nodi più “scarichi”
- Tiene conto delle caratteristiche dei processi (CPU intensive, RAM intensive, I/O etc…) del numero di nodi disponibili e delle loro velocità
- La politica è asimmetrica e decentralizzata; Tutti i nodi usano lo stesso algoritmo di load-balancing e la riduzione del carico è effettuata indipendentemente da ogni coppia di nodi

Caratteristiche di OpenMOSIX (6/9)

Memory ushering

- I processi vengono migrati verso i nodi che hanno una maggiore memoria disponibile per prevenire il trashing dovuto a continue operazioni di swap su disco.
- L'algoritmo di memory ushering ha priorità rispetto a quello di load-balancing
- L'assegnazione di un processo ad un nodo viene effettuata sulla base di un modello matematico che minimizza i costi associati all'ultilizzo delle risorse (memoria, cpu, network-bandwidth…)

Caratteristiche di OpenMOSIX (7/9)

Efficient kernel communication

L'overhead dovuto alle comunicazioni interne del kernel (tra un processo che è stato migrato e il suo nodo di origine) è ridotto al minimo grazie ad un protocollo efficiente, con bassa latenza di startup e un throughput elevato.

Caratteristiche di OpenMOSIX (8/9)

Algoritmi probabilistici di disseminazione delle informazioni

- Ad ogni intervallo di tempo (1 sec) ogni nodo colleziona e dissemina informazioni sul suo stato (CPU, memoria, n. di processi…) verso un subset di altri nodi scelti in modo casuale
- Ogni nodo ha un'elevata conoscenza delle risorse disponibili sugli altri nodi
- I nodi sono in grado di accorgersi di variazioni nella configurazione del cluster

Caratteristiche di OpenMOSIX (9/9)

Controllo decentralizzato e autonomia dei nodi

- Ogni nodo è in grado di prendere autonomamente decisioni sul controllo dei processi
- Non esistono relazioni “master-slave” tra i nodi
- I nodi possono “entrare” o “uscire” dal cluster con la minima perdita di consistenza

OpenMOSIX & PVM/MPI

- PVM/MPI: L'allocazione dei processi è statica. Alcuni nodi possono rimanere idle mentre altri sono sovraccarichi
- OpenMOSIX: I processi migrano da un nodo all'altro garantendo una distribuzione uniforme del carico e una scalabilità lineare
- Applicazioni che utilizzano PVM/MPI possono girare su un cluster OpenMOSIX senza problemi, traendone i vantaggi dovuti alla migrazione dei processi.

MFS: MOSIX File System (1/4)

- OpenMOSIX è molto efficiente nell'eseguire in maniera distribuita processi CPU-bound
- Per gestire in maniera efficiente anche i casi di processi I/O-bound si possono utilizzare dei filesystem con supporto DFSA (Direct File System Access)

MFS: MOSIX File System (2/4)

DFSA (Direct File System Access)

- Le operazioni di I/O vengono effettuate da un processo in locale, sul nodo in cui è in esecuzione e NON via rete, per evitare un overhead
- Il file system deve avere:
  - Uno stesso mount point su tutti i nodi
  - File consistency
  - Time-stamp consistency
- Attualmente pochissimi file system supportano DFSA

MFS: MOSIX File System (3/4)

- OpenMOSIX è dotato di un proprio filesystem: MFS (MOSIX File System), conforme alle specifiche DFSA
- L'algoritmo di load-balancing di OpenMOSIX consente ai processi di migrare verso i nodi su cui devono fare operazioni di I/O
- Con NFS i dati seguono le applicazioni mentre con MFS le applicazioni migrano verso i dati !

MFS: MOSIX File System (4/4)

- MFS è un file system simile a /proc
- Fornisce ad ogni nodo una visione completa di tutti i file system montati fisicamente su tutti i nodi
- Garantisce una consistenza della cache su uno stesso file acceduto da più nodi (la cache è mantenuta sul nodo in cui risiede fisicamente il file)

API di OpenMOSIX

- Nessuna nuova system-call
- Tutte le informazioni vengono gestite attraverso il file system /proc

/proc/hpc
/proc/hpc/admin Amministrazione
/proc/hpc/info Informazioni sul cluster
/proc/hpc/nodes/nnnn/ Informazioni per nodo
/proc/hpc/remote/pppp/ Informazioni sui processi migrati

OpenMOSIX: Installazione

- **Step 1**: Applicare la patch di OpenMOSIX al vanilla kernel di Linux
- **Step 2**: Compilare il nuovo kernel e installarlo sui nodi del cluster
- **Step 3**: Compilare e installare gli “userland-tools”, delle utility per il monitoraggio e il controllo manuale dei processi (mosmon, mosrun, mosctl…)

In alternativa è possibile usare la versione RPM, sia del kernel che degli userland-tools

OpenMOSIX: Configurazione

OpenMOSIX ha un unico file di configurazione (/etc/mosix.map) che deve essere uguale per tutti i nodi. In questo file ci sono scritti gli indirizzi IP dei nodi che compongono il cluster

OpenMOSIX: Monitoring

Il monitoring dei processi su un cluster OpenMOSIX è realizzato con delle utility che leggono le informazioni disponibili sotto /proc/hpc/info (utilizzo di cpu, memoria, velocità…)

- mosmon fa parte degli “userland-tools”
- mosixview con interfaccia grafica X, distribuita gratuitamente all’ indirizzo:

  http://www.mosixview.com

OpenMOSIX diskless farm (1/5)

Perché una farm con nodi diskless ?

- Diminuire i costi
- Centralizzare la gestione del software
- Minimizzare il tempo di installazione, configurazione e amministrazione dei nodi della farm

OpenMOSIX diskless farm (2/5)

Setup della farm:

- Uno dei nodi (dotato di disco rigido) viene scelto come server
- Sul nodo server viene installata una distribuzione di Linux + OpenMOSIX
- I nodi diskless ottengono un indirizzo IP dal server (via BOOTP o DHCP), scaricano il kernel OpenMOSIX (via TFTP) e montano dal server un root file system via NFS

OpenMOSIX diskless farm (3/5)

Etherboot

- Soluzione efficiente per il boot da rete dei nodi diskless
- Gratuito http://www.etherboot.org
- Consente di creare e installare un piccolo bootstrap loader su un qualsiasi supporto di boot (floppy, hd, EPROM di una scheda di rete…)

OpenMOSIX diskless farm (4/5)

Ogni nodo diskless deve montare dal server la root del proprio file system via NFS

- **Problema**: Sul server deve essere mantenuta una root directory per ogni nodo.
- **Soluzione**: **ClusterNFS** http://clusternfs.sourceforge.net

OpenMOSIX diskless farm (5/5)

ClusterNFS è una versione modificata di NFS server

- Consente a tutti i nodi (compreso il server) di condividere lo stesso root file system
- Tutti i file sono condivisi per default
- Un file xxx comune a tutti i client (ma non al server) viene ridenominato xxx$$CLIENT$$
- Un file yyy specifico per un certo client foo.domain.tld con indirizzo ip A.B.C.D viene ridenominato yyy$$HOST=foo.domain.tld$$ oppure yyy$$IP=A.B.C.D$$

OpenMOSIX: altri tipi di installazione

- **ClumpOS**: Una mini-distribuzione di Linux con kernel OpenMOSIX che gira interamente in RAM e viene caricata al boot da CD

  http://clumpos.psoftware.org

- **K12LTSP**: soluzione OpenMOSIX per nodi diskless basata su Linux Terminal Server Project

  http://www.k12ltsp.org

Presente e Futuro di OpenMOSIX

E’ in corso lo sviluppo di:

- Migrating sockets
- Network RAM
- Distributed Shared Memory
- CheckPoint / Restart
- Queue Manager / Scheduler

Si prevede in futuro:

- Inclusione nel kernel 2.6 di Linux
- Riscrivere e ottimizzare MFS

QlusterOS e OpenMOSIX (1/2)

Caratteristiche di QlusterOS (http://qlusters.com):

- E' un prodotto commerciale basato in parte su OpenMOSIX
- Migrating sockets, Network RAM
- Cluster Configurator
- Qsense (detection dei nodi autmatica, senza /etc/mosix.map)
- Monitor (scritto in Flash)
- Queue manager, Launcher, Scheduler
- Job Description Language in XML

QlusterOS e OpenMOSIX (2/2)

- Nuovo Load Balancer
- Migrazione di applicazioni basate su thread
- Kernel Linux 2.4.18
- Più di 100 patch (RedHat Quality)
- Supporto alla migrazione su reti Myrinet e Dolphin
- Integrazione con GFS
- Supporto futuro per AFS

...

...

Conclusioni

- OpenMOSIX è attualmente una delle opzioni più avanzate per clustering HPC
- Nel mondo più di 5.000 nodi eseguono un kernel OpenMOSIX (tra cui INFN-Napoli, INFN-Lecce, Univ. di Pisa, SISSA...)
- OpenMOSIX produce un notevole aumento di performance in ambienti con processi paralleli e ottimizza l'uso delle risorse disponibili in un cluster
- OpenMOSIX non è al momento la scelta migliore in ambienti con applicazioni hardware-dependent e con forti requisiti di High Availability

Riferimenti

- Home page di OpenMOSIX
  http://www.openmosix.org

- Project Manager
  Dr. Moshe Bar [moshe@moelabs.com]