MOSIX Scalable Cluster Computing for Linux

Linux clusters: Beowulf, Mosix e oltre

Roberto Innocente

Argomenti

- Introduzione
- Calcolo parallelo
  - Architetture parallele, tassonomia dei clusters
  - Paradigmi per la programmazione parallela
- Beowulf
- MOSIX
- Metacomputing
- Open Source Numerico/Scientifico

Introduzione/1

I “clusters” come collezione di computers interconnessi che co-operano esistono da molto tempo.

Tuttavia il “cluster computing” e’ esploso solo negli anni ’90 quando furono disponibili:

- microprocessori ad alte prestazioni
- reti ad alta velocita’
- tools standard per la programmazione distribuita

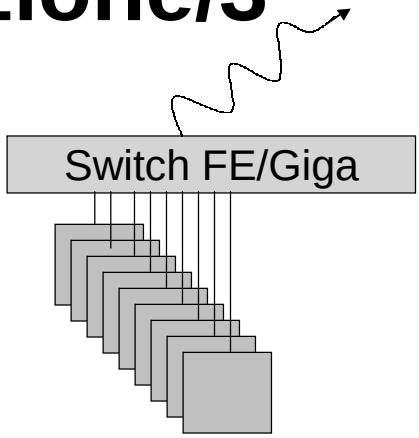
Introduzione/2

I clusters Linux sono stati usati per problemi che per dimensione dei dati da trattare o tempo di CPU necessario non erano trattabili da un singolo calcolatore (***Parallel Cluster****: Beowulf*)

e/o per smaltire un gran numero di job in maniera trasparente per l’utente(***High Throughput Cluster****: Mosix*).

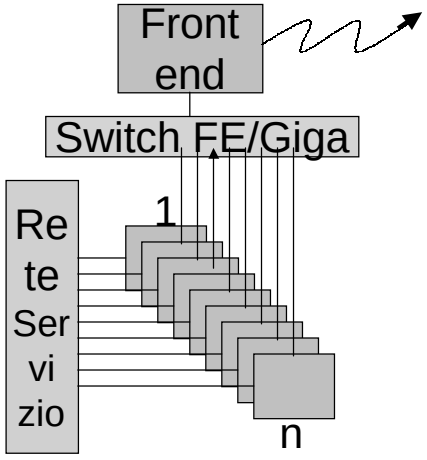
Introduzione/3

Il tipo piu' semplice di cluster si basa semplicemente su un alias DNS, e sulla condivisione dell'ambiente tramite strumenti standard quali NIS(YP) e NFS o AFS.



Introduzione/4

Un altro tipo di cluster linux e' costituito da un nodo accessibile via rete(FrontEnd) che usualmente fa anche da file server ed una serie di nodi in comunicazione tra loro tramite una rete di servizio (eg. molti Parallel clusters)



Calcolo parallelo /1

Sin dagli inizi del calcolo elettronico si e' pensato all'uso di piu' calcolatori collegati che lavorino allo stesso problema per risolverlo in minor tempo.

Molti problemi ammettono una semplice parallelizzazione tramite la partizione del **dominio** da trattare (**domain decomposition**): ad es. integrazione numerica, ricerche in un database.

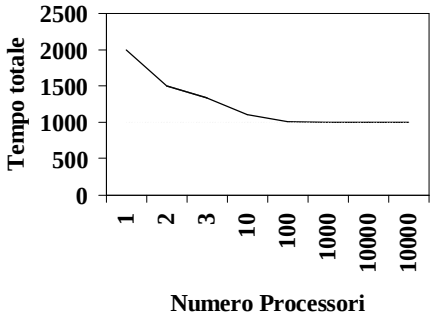
Calcolo parallelo /2

Un famoso articolo di *G.Amdahl* (padre dell'IBM 360)della fine degli anni '60 tuttavia faceva notare che il tempo totale di esecuzione di un programma(***t***) poteva essere suddiviso in un tempo di esecuzione necessariamente sequenziale(***s***) ed un tempo di esecuzione parallelizzabile(***p***) sicche' dati ***n*** calcolatori il tempo totale risulterebbe non minore di

t = s + p/n (legge di Amdahl)

Calcolo parallelo /3

Legge di Amdahl



$$\lim_{n \to \inf} s + P/n - s$$

Calcolo parallelo /4

Questo argomento ha influito in passato sulla valutazione delle potenzialita' del calcolo parallelo (soprattutto sul MPP : ***Massively Parallel Processing***).

In realta' si e' visto che questa *importante* conclusione e' mitigata dal fatto che all'aumentare delle dimensioni dei problemi si incrementa quasi esclusivamente la parte **parallelizzabile** ! (***legge di Gustafon***)

Architetture parallele/1

Classificazione di Flynn (1966):

- SISD (Single Instruction Single Data Stream)
- SIMD(Single Instruction Multiple Data: *la pronuncia corrente e' simdii*)
- MISD(Multiple Instruction Single Data)
- MIMD(Multiple Instruction Multiple Data: *la pronuncia corrente e' mimdii*)

Architetture parallele/2

Data Streams \ Instruction Streams	1	>1
1	SISD computer tradizionali	MISD
>1	SIMD array processors	MIMD multicomputers/ multiprocessors

Architetture parallele/3

- Stallings 1993:
  - SIMD-SM (shared memory),
  - MIMD-SM (shared mem: eg. SMP)
  - MIMD-DSM(distributed shared memory: Cray T3E)
  - MIMD-DM(distributed memory: IBM SP3,Beowulf)
- Tanenbaum: classificazione MIMD
  - loosely coupled: **multicomputer** (distributed/private memory)=distributed processing
  - tightly coupled: **multiprocessor** (shared mem)=parallel computing

Architetture parallele/4

SIMD / MIMD ?

- Negli anni ’80 si sono costruiti alcuni calcolatori SIMD con migliaia di processori (le prime Connection Machine ne sono un esempio di un certo successo)
- Si vide tuttavia che tali macchine erano poco flessibili ed avevano una soddisfacente efficienza solo su un set di problemi relativamente limitato (le Connection Machine CM5 furono poi delle macchine MIMD)

Architetture parallele/5

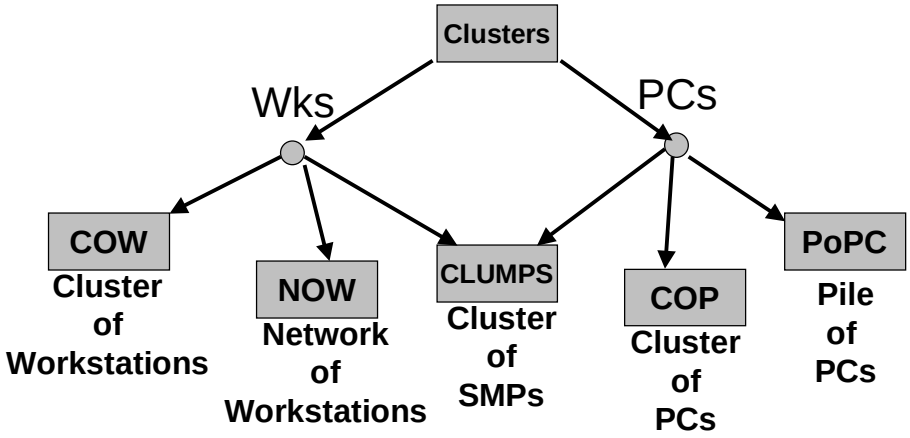
SM / DSM / DM ?

- Le architetture a memoria condivisa(SM: *shared memory*) non scalano per il bottleneck rappresentato dalla banda verso la memoria (al massimo una decina di nodi)
- Le architetture a memoria distribuita, ma globalmente indirizzabile (*DSM: distributed shared memory Cray T3e, SGI Origin 2000)* risultano poco efficienti se programmate non tenendo conto della non uniformita' della memoria (**NUMA** : non uniform memory access)

Architetture parallele/6

L'IBM SPx (SP1,SP2,SP3..) e' stata la macchina commerciale MIMD composta da nodi con h/w non esoterico che ha dimostrato la scalabilita' (migliaia di nodi) delle performance dei clusters Unix.

Tassonomia dei clusters



Programmazione distribuita/1

- Active messages :
  - il messaggio attiva nel processo ricevente una funzione che ne processa il contenuto : l'attivazione e' asincrona, non vi e' alcuna chiamata receive
- Message Passing:
  - PVM (Parallel Virtual Machine : *Beguelin,Dongarra et al.)* semantica stabilita dall'unica implementazione (http://www.epm.ornl.gov/pvm/pvm_home.html)
  - MPI (Message Passing Interface) *uno standard elaborato da vendor, universita', centri di ricerca* : semantica stabilita da un corposo standard (http://www-unix.mcs.anl.gov/mpi/mpi-standard/mpi-report-1.1/mpi-report.html)

Programmazione distribuita/2

Active Messages e' un paradigma molto flessibile, ma la necessita' di gestire eventi asincroni e la difficolta' di debuggare i programmi ne hanno precluso una larga adozione.

All'inizio degli anni '90 PVM ha avuto una larga diffusione, ma la mancanza di una definizione formale e la difficolta' dei principali vendors di portarlo in maniera efficiente sulle loro architetture, hanno alla fine decretato il largo successo di MPI.

MPI/1

Le chiamate fondamentali di MPI sono:

- Inizializzazione:
  - `MPI_Init(int argc,char* argv[ ])`
  - `MPI_Comm_size(MPI_COMM_WORLD,&size)` : quanti processori ci sono ?
  - `MPI_Comm_Rank(MPI_COMM_WORLD,&rank)`: qual'e' il numero del mio processore ?

MPI/2

- Scambio :
  - `MPI_Send(void* buf,int count,MPI_Datatype dt,int dest,int tag,MPI_Comm comm)`
  - `MPI_Recv(void* buf, int count,MPI_Datattype dt,int source,int tag MPI_Comm comm,MPI_Status* stat)`
- Chiusura:
  - `MPI_Finalize()`

MPI: implementazioni

Esistono 2 implementazioni open source ampiamente diffuse che vengono normalmente usate su Linux:

- MPICH sviluppato ad Argonne (http://www-unix.mcs.anl.gov/mpi/mpich)
- LAM sviluppato da Ohio Supercomputer Center (http://www.lam-mpi.org/)

Modelli programmazione/1

SPMD (Single Program Multiple Data) :

un solo programma con costrutti di controllo che scelgono parti differenti del programma a seconda del nodo su cui viene lanciato

MPMD (Multiple Program Multiple Data) :

i singoli nodi eseguono programmi differenti (tipicamente master/slave)

Modelli programmazione/2

Il modello SPMD e' quello che di gran lunga si e' dimostrato piu' facilmente manutenibile. Es.:

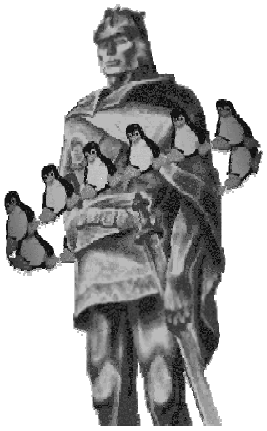
```
if (thisnode == 0)
   MPI_Send(txmbuff,.,1,.);
elseif (thisnode == 1)
   MPI_Recv(rcvbuff,.,0,.);
```

MPI hello.c

```
int me,nprocs; char hello[40],char recbuf[40];
MPI_Init(argc,argv);
MPI_Comm_size(MPI_COMM_WORLD,&nprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&me);
sprintf(hello,"i'm %d of %d\n",me,nprocs);
if (me == 0){
    for(from=1;from<nprocs;from++){
      MPI_Recv(recbuf,40,MPI_BYTE,from,...);
      puts(recbuf);
    }
} else {
    MPI_Send(hello,40,MPI_BYTE,0,...);
}
MPI_Finalize();
```


Beowulf /1

Beowulf e' l'eroe di un romanzo epico che e' tra i piu' antichi documenti in lingua inglese pervenutici(XI^{o} sec). Egli libera i danesi dal mostro marino **Grendel** che li opprimeva.



Beowulf /2

- Il progetto **Beowulf** inizia alla Nasa (Goddard Space Flight Center) nel 1994.
- Si propone di liberare scienziati e tecnici dagli enormi sforzi finanziari e tecnologici che richiedono i **supercalcolatori**(Grendel).
- *Becker ,Sterling* e *Savarese* sono tra i principali investigatori della possibilita' di usare in loro sostituzione degli economici PC(Beowulf).

Beowulf /3

- 1º Beowulf(1994): 16 Intel 486 @ 66Mhz, interconnessi da Ethernet a 10Mbps, Linux 0.99pl14 !
- 2º Beowulf(1995): 16 Pentium @ 100 Mhz, interconessi da Fast Ethernet(100Mbps)
- 3º Beowulf: ...

Il sito del progetto Beowulf e' : http://www.beowulf.org/

Beowulf /4

Questo lavoro che per molti puo' risultare esoterico ha avuto importanti ricadute.

Molti dei driver Linux per schede Ethernet derivano dal lavoro di *Donald Becker* a questo progetto.

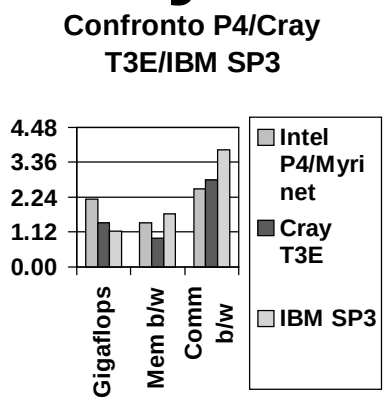
Deriva da questo progetto l'implementazione su Linux del *channel bonding* : la possibilita' di affasciare piu' schede di rete in un unico dispositivo di rete virtuale.

Beowulf /5

In definitiva:

- un Beowulf e' un cluster in cui sono installati i tools e librerie necessarie alla esecuzione di programmi paralleli
- un Beowulf usa componenti largamente diffusi, pezzi disponibili attraverso la grande distribuzione : **M²COTS (Mass-Market Commodity Off The Shelf)**

Confronto P4 / Cray T3E/IBM SP3



- Pentium 4@2Ghz :oltre 2 Gflop/s in moltiplicazione di matrici usando le istruzioni SSE2 (http://hpc.sissa.it/p4/)
- Pentium 4@2Ghz con memoria RIMM: banda cpu/memoria dell'ordine di 1.5 GB/s
- Myrinet 3: banda 240 MB/s, latenza 4usec

Mosix /1

- Nato nel 1985 come S.O.: Multicomputer OS (MOS), *Amnon Barak (HUJI)*
- Dal 1990 diventa un insieme di patches per BSD: MOSix
- Dal 1995 diventa un insieme di patches per Linux

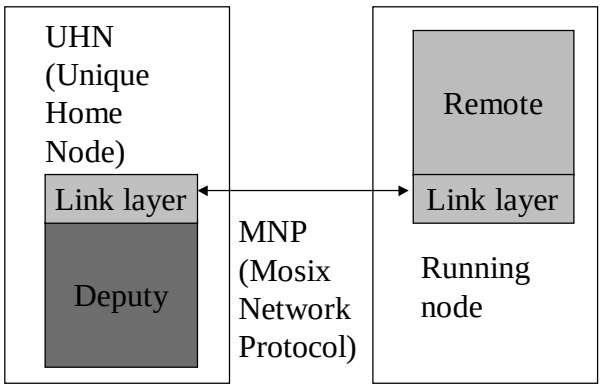
Mosix /2

- Permette la **migrazione trasparente** dei processi da un nodo del cluster ad un altro
- Usa la migrazione trasparente dei processi per **bilanciare automaticamente il carico** tra i nodi del cluster
- Permette quindi di realizzare clusters **High Throughput**: capaci di completare un alto numero di lavori (jobs)

Mosix /3

La migrazione e' trasparente poiche' il kernel intercetta tutte le chiamate di sistema del processo migrato (**remote**) e tramite un proprio protocollo (**MNP: Mosix network protocol**) le fa eseguire dallo scheletro del processo originale (**deputy**) rimasto sul nodo di origine (**Unique Home Node**).

Mosix /4



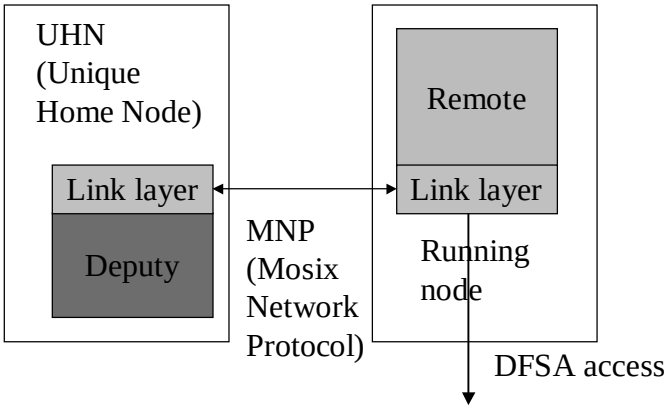
Mosix /5

Per permettere che anche processi con molto I/O possano sfruttare efficacemente la migrazione, Mosix ha aggiunto la possibilita' che per i filesystems **node-indipendent** e cioe' :

- **globali** : un file ha lo stesso nome su tutti i nodi
- **coerenti** : o vi e' una sola cache,oppure le caches sono sincronizzate

l'I/O avvenga direttamente sul nodo di esecuzione(**DFSA: Direct File System Access**)

Mosix /6



Mosix /7

Links utili:

- le slides di un breve tutorial sono disponibili su http://hpc.sissa.it/walrus/mosix01/
- un monitor in java per clusters Mosix e' disponibile su http://hpc.sissa.it/walrus/mjm/mjm.htm
- la home del progetto Mosix e' : http://www.mosix.org/

Metacomputing/1

Il termine si e' diffuso da un'articolo di L.Smarr (NCSA) del 1992 in cui con tale termine si denotava "***l'uso di potenti strumenti di calcolo disponibili all'utente in maniera trasparente per mezzo di un ambiente di rete***": un supercomputer virtuale creato per mezzo della rete.

Un metacomputer non e' un computer parallelo in quanto il suo pool di nodi e' dinamico ed e' costituito da macchine indipendenti.

Metacomputing/2

Inoltre un metacomputer non e' un cluster in quanto le capacita' dei collegamenti non sono omogenee potendo essere i nodi anche distribuiti su rete geografica (WAN).

Un analogia spesso usata e' quella della rete di distribuzione dell'energia elettrica (**electricity grid**). Questa e' tecnologicamente complicata (molti generatori, a migliaia di chilometri di distanza,...), pero' e' facilmente usabile anche da un bambino che ci attacca un piccolo elettrodomestico.

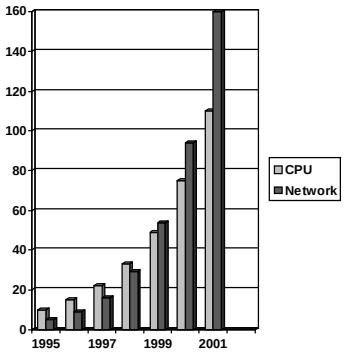
Metacomputing/3

Esempi :

- Condor (http://hpc.sissa.it/batch/index.html)
- SETI@HOME (http://www.seti.org)
- Entropia (purpose computing: Andrew Chien, UIUC) (http://www.entropia.com/)

Grid Computing/1

Questa visione che il calcolatore e' **la griglia dei calcolatori interconnessi in rete** nasce dalla previsione che il trend di crescita delle capacita' della rete sara' molto maggiore di quello di crescita delle capacita' delle CPU.

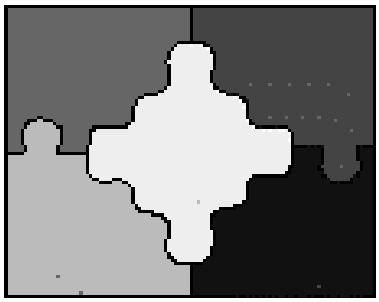


Grid Computing/2

- Le capacita' delle reti attuali (~1/2 Gb/s) sono dell'ordine delle capacita' dei bus
- In laboratorio si sono gia' da un po' realizzati prototipi di Ethernet a 10 Gb/s
- 10 Gb/s e' dell'ordine di PCI-X, che pero' ha latenze troppo alte
- Infiniband la rete *switched* che probabilmente sostituira' gli attuali bus sta comparendo con velocita' di 2.5/5 Gb/s

Grid Computing/3

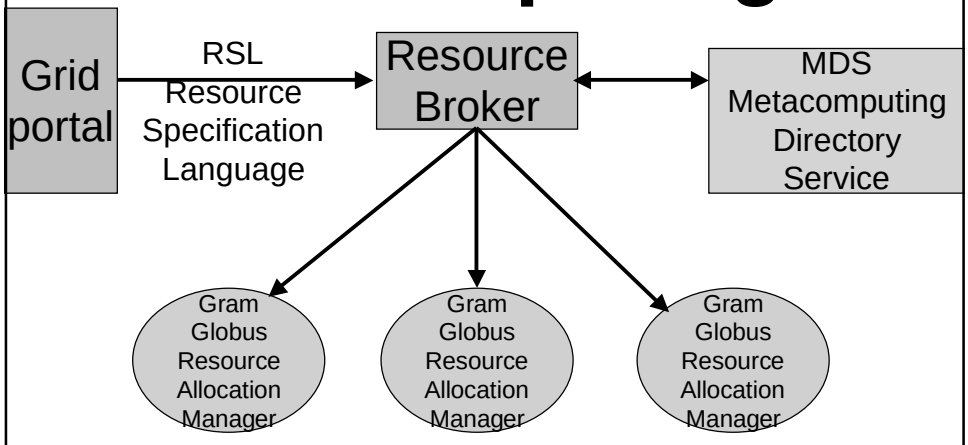
Grid Computing e' la "buzzword" piu' citata negli ultimi tempi a proposito di metacomputing. Si richiama a tutta una serie di tools ed iniziative che si ricollegano a I. Foster e C.Kesselman editors del libro: *Grid Computing: Blueprint for a New Computing Infrastructure*



Grid Computing/4

- Il toolkit sviluppato per il progetto Grid si chiama Globus (http://www.globus.org).
- Questo tool e' composto principalmente da middleware per una infrastruttura adatta all'utilizzo in maniera trasparente di un ambiente di calcolo distribuito.
- Anche questo toolkit viene sviluppato per la gran parte su Linux.

Grid Computing/5



Open Source Scientifico/1

Il software numerico/scientifico open source e' stato spesso di qualita' elevata e frequentemente migliore di quello presente nelle offerte commerciali.

Uno dei principali sostenitori e' Jack Dongarra (Oak Ridge National Lab e University of Texas)

Stadi:

- fino meta' anni 80: nastri
- meta' 80 – 90: netlib@research.att.com
- anni 90: http://www.netlib.org

Open Source Scientifico/2

Tra i piu' importanti esempi citiamo:

- Dongarra et al.:
  - ATLAS(BasicLinearAlgebra auto configurabile)
  - LAPACK (algebra lineare per matrici dense)
  - SCALAPACK (alg.lin. per calcolatori paralleli)
- QUADPACK (integrazione numerica)
- ODEPACK (eq.diff. ordinarie)
- FFTW (trasformate di Fourier)
- MINPACK (minimizzazione)

Open Source Scientifico/3

Iniziative:

- Scientific Web (http://www.scientificweb.com/)
- progetto Open Science: http://www.openscience.org/

Clusters Linux a Trieste/1

- 1998: 20 Pentium II @ 450 Mhz, interconnessi da FastEthernet (100Mbps) all'ICTP

(http://www.ictp.trieste.it/parallel/performance/)

- 2000: biprocessori Pentium III@550 Mhz(katmai), interconnessi Myrinet 2 e Gigabit Ethernet ICTP/SISSA (http://hpc.sissa.it/paper01/)

Clusters Linux a Trieste/2

- 2000: biprocessori Pentium III@933 Mhz(Coppermine), interconnessi Myrinet 3 ICTP/SISSA
- 2000: cluster Mosix con biprocessori Pentium III e interconnessione con 2 FastEthernet affasciate alla SISSA http://hpc.sissa.it/walrus/mosix01/
- 2001: biprocessori Pentium 4@1.7Ghz, interconnessi Myrinet 3 ICTP/SISSA

Bibliografia/1

- *How to build a Beowulf* Sterling,Becker
- *High Performance Clusters,* ed. Rajkumar Buyya
- *In search of Clusters,*Pfister
- *History of scientific computation,* ed. S.Nash ACM 1987
- *The Grid: Blueprint for a new computing infrastructure,* ed. Foster, Kesselman

Bibliografia/2

- *Using MPI: Portable Parallel Programming with the Message Passing Interface*, Gropp et al.
- *Parallel programming with MPI*, Morgan Kaufmann Publishers,Inc
- *High Performance Computing*, Dowd
- *MPI: The Complete Reference*, Snir