



 POLITECNICO DI MILANO

# Impianti Informatici

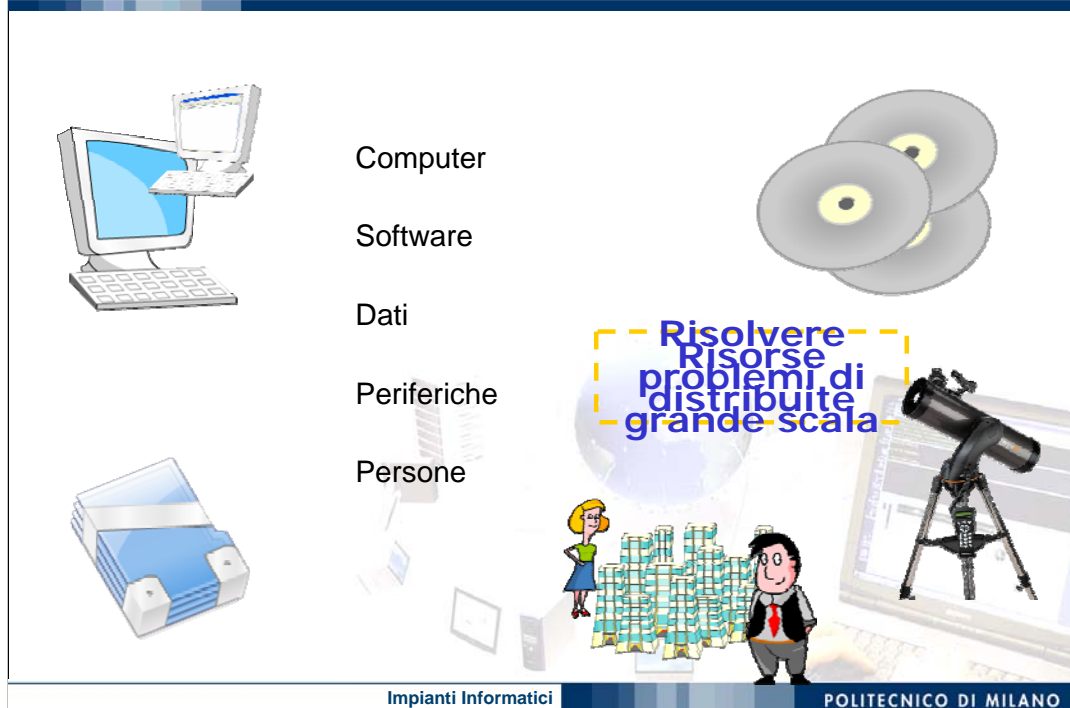


Grid computing e architetture parallele



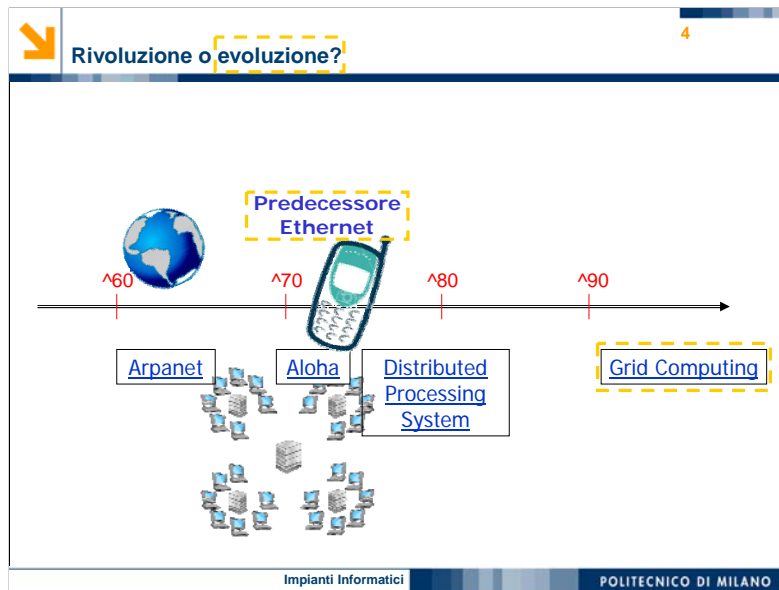
Il Grid Computing rappresenta il futuro nel mondo dell'Information Technology

1. e del Web. Non esiste una definizione di Grid Computing, tanto più che il termine è spesso usato con accezioni diverse in differenti contesti.
2. Possiamo vedere il Grid come un paradigma, o un'infrastruttura che abiliti
3. la condivisione,
4. la selezione
5. e l'aggregazione
6. di risorse distribuite geograficamente.



### L'obiettivo del Grid Computing

1. è quello di rendere fattibile la risoluzione di problemi complessi di grande scala.
2. A tal proposito, le risorse che vengono condivise possono essere di svariato tipo:
3. PC, workstation, cluster di computer o supercomputer, laptop o notebooks, dispositivi mobili e PDA
4. Oppure applicativi particolari,
5. O ancora cataloghi di dati o database, ad esempio l'accesso al genoma umano
6. O Dispositivi e strumenti speciali, come ad esempio radio telescopi o sensor network
7. Infine, altro tassello fondamentale della grid, è la sua capacità di far collaborare ed interagire fra loro persone e organizzazioni.



I concetti che stanno alla base del Grid Computing non rappresentano nulla di rivoluzionario, la novità è nel modo in cui vengono utilizzati e gli scopi per cui vengono integrati fra loro.

1. Il Grid Computing si appoggia sostanzialmente sui protocolli e le tecnologie utilizzate sul web,
2. ed è quindi naturale che il progetto ARPANET sia una sua solida fundamenta.
3. Altrettanto significativo è un progetto sviluppato qualche anno dopo, l'Aloha, anch'esso sviluppato grazie ai fondi dell'ARPA. La rete ALOHA, creata nel 1970, permetteva, in modo simile all'ARPANET,
4. l'accesso agli stessi sistemi da parte di persone poste in differenti località. La differenza tra i due progetti, è che mentre l'arpanet usava linee telefoniche dedicate,
5. aloha usa le onde radio come mezzo trasmissivo. L'importanza di quest'ultimo era quindi il fatto che usava un mezzo condiviso per l'invio dei dati, il che implica modelli di comunicazione più recenti, come CSMA/CD. Mentre in ARPANET ogni nodo poteva solo parlare con un nodo all'altro estremo della rete, in ALOHA ognuno usava la stessa frequenza, necessitando così di meccanismi di controllo.
6. La situazione era quindi molto simile a quella delle moderne Ethernet e delle reti Wi-Fi.
7. Già 30 anni fa si parlava di Sistemi di calcolo Distribuito, ovvero di sistemi che si avvalgono di più computer o processori per eseguire un'applicazione.
8. E' verso la metà degli anni 90 che inizia ad emergere il termine Grid Computing,
9. come evoluzione della computazione distribuita.

➤
**Sistemi di calcolo distribuito**
5

Availability e Reliability  
 Prestazioni  
 Modularità e facilità di espansione  
 Condivisione delle risorse  
 Automatica ripartizione del carico  
 Buone prestazioni anche in caso di overload

Impianti Informatici
POLITECNICO DI MILANO

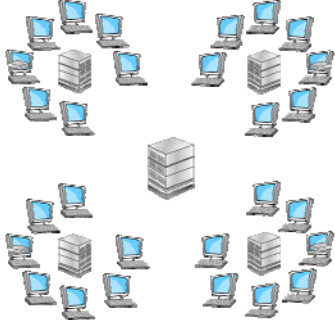
I sistemi di calcolo distribuito, consentono di realizzare impianti che siano

1. Altamente affidabili e disponibili; infatti la ripartizione del calcolo tra multipli nodi e risorse permette di disporre di capacità di elaborazione anche qualora un elemento della rete dovesse venire a mancare.
2. Le prestazioni di tali architetture sono inoltre elevate,
3. e soprattutto facilmente incrementabili per la loro natura modulare; i sistemi sono quindi facilmente estendibili tanto nelle capacità quanto nelle funzionalità.
4. Essi permettono inoltre la condivisione automatica delle risorse
5. e la ripartizione del carico,
6. garantendo buoni tempi di risposta anche in caso di temporanei sovraccarichi del sistema.

➤

**Sistemi di calcolo distribuito: peculiarità**
6

Molteplicità di risorse (repliche per affidabilità/prestazioni)  
 Interconnessioni (disaccoppiamento tra componenti)  
 Unità di controllo  
 Trasparenza  
 Autonomia dei componenti



Impianti Informatici
POLITECNICO DI MILANO

1. Il sistema dovrebbe fornire un numero di risorse assegnabili a seconda della domanda di servizio. Più elevato è il grado di replicazione delle risorse, maggiore è la capacità del sistema di mantenere un alto livello di affidabilità e prestazioni.
2. Un sistema distribuito dovrebbe includere una sottorete di comunicazione che interconnetta gli elementi del sistema e che usi protocolli che disaccoppino il più possibile le due parti comunicanti.
3. Tutti i componenti devono operare per perseguire un obiettivo comune, coordinati da un'unità di controllo.
4. L'insieme di risorse che costituisce il sistema deve apparire all'utente come una singola macchina virtuale; quando un utente richiede un servizio, non deve preoccuparsi della locazione fisica, o del carico attuale delle risorse.
5. I componenti del sistema, sia logici che fisici, dovrebbero essere autonomi, pur interagendo fra loro in modo da raggiungere l'obiettivo comune, e aderendo a insiemi di politiche comuni.

➤
**Sistemi di calcolo distribuito: problematiche**
7

Eterogeneità

Addressing degli oggetti

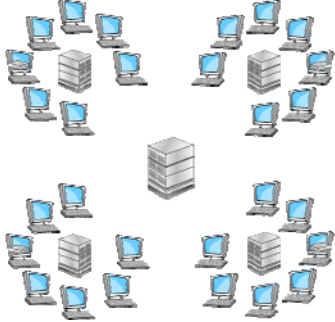
Nomenclatura

Ownership

Accesso concorrente

Coerenza delle cache

Parallelizzazione del problema  
(scomposizione del dominio)



Impianti Informatici
POLITECNICO DI MILANO

Operare in un sistema distribuito comporta alcune problematiche non riscontrabili in comuni sistemi, quali


- l'eterogeneità dell'impianto, composto da risorse hardware e software differenti.
- Collegato a questo aspetto vi è l'indirizzamento degli oggetti distribuiti
- e il relativo spazio dei nomi,
- oltre alle questioni inerenti la proprietà degli elementi distribuiti.
- Sorgono problemi dovuti all'accesso concorrente alle risorse, e
- alla gestione delle cache,
- Inoltre non è sempre facile scomporre il dominio e parallelizzare le applicazioni



La motivazione principale che ha portato alla nascita e diffusione dei sistemi a computazione distribuita

1. è la richiesta di potenza di calcolo sempre crescente. È infatti richiesta una maggiore capacità di elaborazioni per
2. eseguire simulazioni,
3. ad esempio di dinamica molecolare, riproduzioni di crash test
4. oppure previsioni del tempo,
5. o, ancora, applicazione di filtri ed effetti speciali a film o spot televisivi
6. Una maggiore potenza di calcolo permette inoltre di evitare la fase di prototipazione,
7. consentendo in questo modo la riduzione dei costi e del time-to-market, garantendo una maggiore competitività sul mercato





## Parallel Computing

9

---

Speed-up

- indica l'incremento di prestazioni al crescere del numero di processori

$$S_n = T_1/T_n$$

Efficienza

- esprime la capacità di sfruttare la potenza di calcolo

$$E_p = \frac{S_p}{p}$$

Impianti Informatici
POLITECNICO DI MILANO

Alla base dei sistemi di calcolo distribuito, vi è la computazione parallela, ovvero la scomposizione di un problema in più sottoproblemi eseguibili in modo parallelo.

Non tutti i problemi sono parallelizzabili, data la loro natura spesso sequenziale. Due parametri tipicamente utilizzati per valutare la capacità di un applicativo di sfruttare un maggior numero di processori sono:

1. Lo speed-up e
2. L'efficienza
3. Lo speed-up indica l'incremento delle prestazioni al crescere della potenza di calcolo impiegata, in particolare del numero di processori.
4. Viene definito come il rapporto tra
5. il tempo per eseguire l'applicativo su un singolo processore
6. e quello per eseguirlo su p processori
7. L'efficienza è invece un indice che esprime la capacità di sfruttare la potenza di calcolo,
8. e si definisce come il rapporto tra
9. lo speed-up con p processori,
10. e il numero di processori

Legge di Amdahl (1967)

10

Tempo di esecuzione della frazione seriale  $T_s$       Tempo di esecuzione della frazione parallela  $T_p$

$$T_l = T_s + T_p$$

$$T_n = T_s + T_p/n$$

Frazione seriale  $\rightarrow f_s = T_s / (T_s + T_p)$

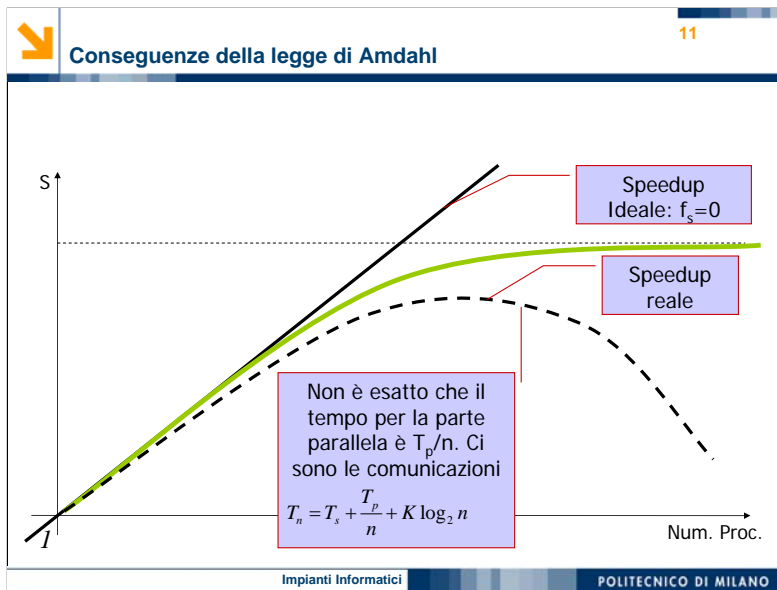
$$S(n) = \frac{n}{n \cdot f_s + (1 - f_s)}$$

Impianti Informatici
POLITECNICO DI MILANO

La legge di Amdahl, formulata nel 1967, permette di valutare l'impatto della parte non parallelizzabile degli applicativi.

Si supponga di poter scomporre il tempo di esecuzione di un'applicazione

1. nelle due componenti seriale e parallela,
2. il tempo di esecuzione su singolo processore sarà dato dalla loro somma.
3. Se ora si dovessero utilizzare più processori,
4. la parte seriale non gioverebbe in alcun modo del potenziamento,
5. mentre la parte parallela scalerebbe in maniera proporzionale
6. Definendo la frazione seriale come il rapporto tra il tempo di esecuzione della parte seriale del programma e il tempo di esecuzione totale su singolo processore
7. La legge di Amdahl esprime lo speed-up in funzione del
8. numero di processori,
9. e della componente seriale dell'applicazione.



La principale implicazione della legge di Amdahl è che non è possibile raggiungere

1. uno speed-up ideale, ovvero lineare rispetto all'aumentare del numero di processori, perché questo richiederebbe una frazione seriale nulla
2. Secondo amdahl lo speed-up è quindi sempre inferiore a quello ideale,
3. Nella realtà lo speed-up è in genere inferiore a quello teorico, dovuto al fatto che non è vero che la parte parallela scala in maniera lineare,
4. ma occorre tener conto delle comunicazioni tra i vari componenti, crescenti all'aumentare del numero di processori.



## Legge di Amdahl: esempio

12

Un programma che viene eseguito in 100 s su un processore singolo, ha una frazione sequenziale dell'1%. In quanto tempo viene eseguito su 10 processori?

$$S = 10 / (.1 + .99) = 10/1.09 = 9.174$$

$$T(n) = T(1)/S = 100/9.174 = 10.9 \text{ s}$$

$$E = 9.174/10 = 91.7\%$$

E su 100 processori?

$$S = 100/(1 + .99) = 100/1.99 = 50.25$$

$$T(N) = T(1)/S = 100/50.25 = 1.99 \text{ s}$$

$$E = 50.25/100 = 50\%$$



La dimensione del problema, in genere, cresce all’aumentare del numero di processori.

$T(k, n)$  = tempo di esecuzione di un sistema con  $n$  processori per risolvere un problema di complessità  $k$ .

Speedup scalato:

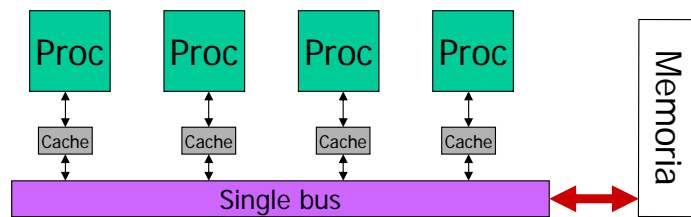
$$S(n) = \frac{T(1, k(n))}{T(n, k(n))}$$



Si possono classificare le macchine parallele in base al modo in cui è organizzata la memoria:

- A **memoria distribuita** (es. Intel Paragon, IBM SPx, cluster)
- A **memoria condivisa** (es. SGI Power Challenge, Cray T3D)

Entrambe i tipi di architetture possono essere usati con i "modelli software" SIMD e MIMD (si veda più avanti).



Non esistono percorsi preferenziali di accesso alla memoria. Vengono dette SMP (Symmetric MultiProcessors). Sono possibili 2 implementazioni:

- UMA (Uniform Memory Access)
- NUMA (Non Uniform Memory Access)

Il collo di bottiglia è il BUS: non si possono mettere “troppi” processori (non più di 16)

Esiste problema di coerenza delle cache

Le macchine UMA vengono anche dette “share everything”.

NUMA è un metodo per configurare 1 cluster di CPU (tipicamente 4 CPU): implementano una cache L2 cioè una memoria condivisa solo fra le 4 CPU del gruppo.



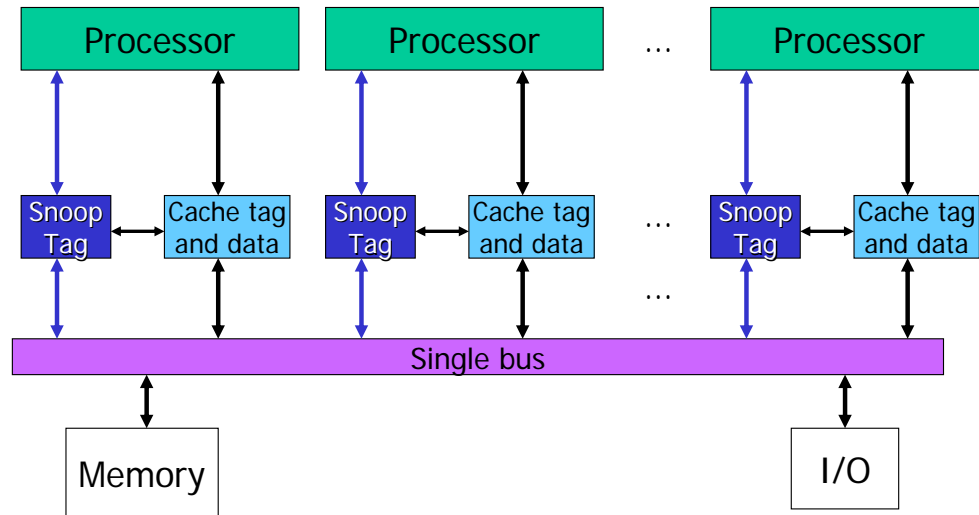
La possibilità che ogni processore lavori su dati differenti comporta problemi di coerenza delle cache (di ogni processore)

L'algoritmo più utilizzato per gestire la cache coherency è detto **snooping algorithm** (per sistemi a singolo bus)

Lecture multiple dello stesso dato non creano problemi

- è importante, invece, garantire la coerenza in fase di scrittura







Quando un processore aggiorna un dato shared, per prima cosa deve invalidare tutte le copie presenti nelle cache. Esistono due metodi:

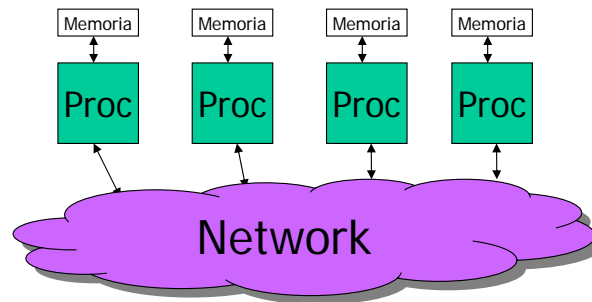
- **Write Invalidate:** prima di aggiornare il dato (nella cache locale) viene mandato un segnale di invalidate e NON si aggiorna la memoria
  - Se altri processori hanno bisogno del dato, lo si scrive in memoria e gli altri andranno a leggere il nuovo dato
- **Write Update:** invece di invalidare le cache, si aggiornano direttamente mandando il nuovo dato su bus
  - chi possiede quello vecchio, lo aggiorna. Anche chiamato write-broadcast
  - E' molto band consuming

(Write back/ write through)

Snooping Alg. Usato dal 1980.

W. Invalidate è simile a W. BACK per la riduzione della banda.

W. UPDATE è simile a W. THROUGH.



Ogni processore ha il suo spazio di indirizzamento privato

L'utente deve gestire in maniera esplicita la distribuzione dei dati e la loro allocazione



## Tipologie di elaborazione parallela (Flynn's Taxonomy)

20

[SISD]: elaborazione sequenziale

SISD

SIMD

(MISD)

MIMD

**SIMD:** *Single Instruction, Multiple Data.*

- Calcolatori vettoriali: una sola Control Unit, un solo clock (Array processors, simile a DSP)

**MIMD:** *Multiple Instructions, Multiple Data.*

- Ogni processore esegue codice indipendente dagli altri
- **SPMD:** *Single Program, Multiple Data.* Stile di programmazione delle macchine MIMD: ogni processore ha i "suoi" dati ma tutti eseguono lo stesso programma



I seguenti esempi si riferiscono a macchine a memoria condivisa (es. SMP)

Sono presentati in un linguaggio, *inesistente*, C-like

Il problema, elementare, che si vuole risolvere è la somma di  $N$  interi con *nproc* processori



```
#define N 100000
int a[N];
int i, s; //i: contatore, s: somma
...
void main(){
    s=0;
    for (i=0; i<N; i++)
        s = s + a[i];
    ...
}
```



```
#define N 100000
#define M (N/nproc)

share int a[n];
share int par_sum[nproc];
int i, s;
...
void main(){
...
    s=0;
    for (i = M*IDproc; i < M*(IDproc+1); i++)
        s = s+a[i];
    par_sum[IDproc]=s;
```

contiene le somme  
parziali di ciascun  
processore

ogni processore  
ha una copia di  
queste variabili

**Come sommo i risultati parziali ?**



**Soluzione intuitiva** (poco efficiente): il processore 0 (master) esegue tutte le somme parziali

```
synch();  
  
if (IDproc==0){  
    for (i=1; i<nproc; i++)  
        s=s+par_sum[i];  
}
```

Per evitare che cominci a sommare i parziali prima che gli altri abbiano finito di calcolarli

L'uso di dati condivisi crea problemi di prestazioni (OH maggiori)

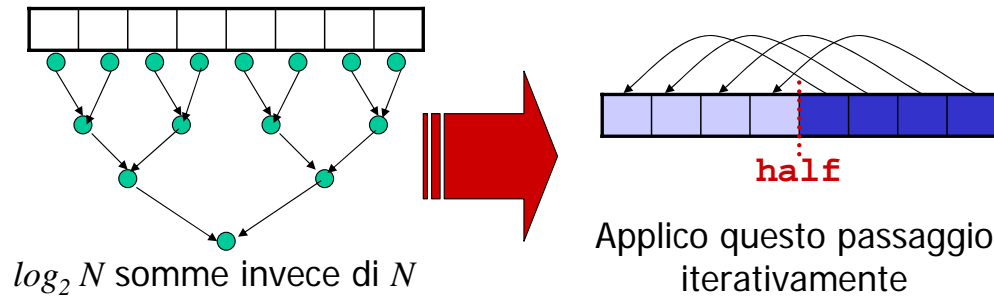
→ Vedi PARALLELIZING PROFILE.

```
|  
|_  
||  
||-----+  
||-----|→
```





**Soluzione ottima.** Si parallelizza anche la somma parziale. Questo approccio funziona in modo ottimo se il numero di elementi da sommare è una potenza di 2





```
int half= nproc/2;
while(half>0){
    synch();
    if (IDproc<half)
        sum[IDproc]=sum[IDproc] +
        sum[IDproc + half];
    half=half/2;
}
```

devo sincronizzare  
ogni ramo



### HPF (High Performance Fortran)

- Prima versione rilasciata nel 1994
- HPF è Fortran 90 + lo statement FORALL e le direttive di decomposizione dei dati
- Approccio SPMD. Linguaggio di alto livello, no message passing.
- Esempio di forall:
  - `forall(i=1:m, j=1:n) a(i,j) = b(i,j)*j + (i-1)*m`
  - Corrisponde a cicli `do` annidati. Il compilatore li parallelizza automaticamente
  - Esistono direttive per gestire situazioni di dipendenza degli indici dai cicli

### OpenMP

- Insieme di direttive per standardizzare la programmazione di macchine a memoria condivisa (fine e coarse grained parallelism): C/C++, fortran



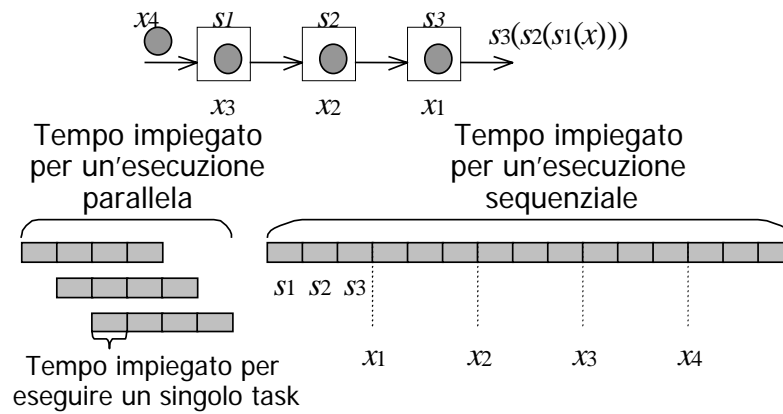
**Skeleton:** insieme di strutture/modelli tipici che si possono riscontrare nelle applicazioni parallele

Esempi di skeleton sono:

- **Pipeline:** si ottiene scomponendo una funzione in sotto funzioni “consecutive” implementabili in maniera indipendente (es.  $f(g(x))$  con  $x$  vettore)
- **Farm:** frazionamento del problema in sottoproblemi che vengono affidati da un master a dei worker
- **Loop:** parallelizzazione dei cicli
- **Sequenziale,**
- ...



## la pipeline



Esempio di  
Codice:

```
pipe nome_modulo in(inlist) out(outlist)
  < chiamata dello stadio 1 >
  ...
  < chiamata dello stadio n >
end pipe
```



**PVM** (Parallel Virtual Machine) – package sw con VM

- Fornisce il supporto per un OS distribuito
- Controllo della generazione di processi da parte dell'utente
- Molto utilizzato ancora oggi
- Costituito da un insieme di librerie (da includere nel codice dell'applicazione) e da un supporto run-time (demoni).

**MPI** (Message Passing Interface): approfondito nel seguito...

I demoni PVM implementano una VM che fornisce supporto ad un OS distribuito.

MPI non definisce una VM. Usa molti meccanismi (a seconda dell'implementazione) per attivare proc. remoti.



È un insieme di specifiche (API):

- Modello message-passing
- Non è un set di specifiche per i compilatori
- Non è un prodotto

Per computer paralleli, cluster e reti eterogenee

Full-featured

Progettato per consentire lo sviluppo di librerie di software parallelo

Progettato per fornire accesso a caratteristiche avanzate delle macchine parallele per:

- Utenti finali
- Sviluppatori di librerie
- Sviluppatori di programmi



Specifiche rilasciate nel 1994 (MPI 1.0) dal MPI Forum costituito da un'ampia rappresentanza di produttori hw, sw e sviluppatori

Indipendente dalla piattaforma hw/sw su cui si appoggia

Linguaggi ospiti: Fortran, C, C++, ...

Esistono implementazioni open source e proprietarie

Versione oggi utilizzata: MPI 2.0

MPI è principalmente per SPMD/MIMD. (HPF è un esempio di interfaccia SIMD)





MPI è solo un insieme di API. Quindi l'implementazione sulla specifica architettura deve essere lasciata a software di livello più basso: **i device**

I device implementano le specifiche MPI sulle varie architetture (NEC, Cray, cluster, ...)

Spesso sono realizzati da sistemi di comunicazione proprietari precedenti ad MPI

Per i cluster il device più utilizzato è p4 (MPICH)



Possibilità di definire sottogruppi di processori (**communicators**)

Programmi portabili (API standard)

Comunicazioni Point-to-point (bloccanti e non) e collettive (**broadcast** e **reduce**) per sincronizzazione

Caratteristiche non previste e non supportate dalle specifiche MPI:

- process management
- remote memory transfers
- active messages
- threads
- virtual shared memory



Necessità di realizzare un programma parallelo portabile

Sviluppo di una libreria parallela

Problema irregolare o *dynamic data*

Relazioni che non si adattano ad un modello *data parallel*

...e quando NON usare MPI:

- Si può usare HPF o una versione parallela di Fortran 90
- Non serve il parallelismo
- Si possono usare librerie (magari scritte in MPI)



$$\int_0^1 \frac{4}{1+x^2} dx = 4 \cdot \arctan(x) \Big|_0^1 = \pi$$

Per eseguire l'integrale, l'intervallo 0-1 viene diviso in  $n$  parti.

Ogni processore calcola il suo "pezzo"

Più grande è  $n$  e maggiore è l'accuratezza: metodo pessimo ma utile a scopo didattico.



```
#include "mpi.h"
#include <stdio.h>
#include <math.h>

int main(int argc, char *argv[]){

    int done = 0, n, myid, numprocs, i;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
```




```
while (1){
    if (myid == 0){
        printf("Enter the number of intervals:
                (0 quits) ");
        scanf("%d",&n);
    }
    MPI_Bcast(&n, 1, MPI_INT, 0,
MPI_COMM_WORLD);
    if (n == 0)
        break;
    else{
```






### Esempio: pi-greco (3)

39

```
h = 1.0 / (double) n;
sum = 0.0;
for (i = myid + 1; i <= n; i += numprocs){
    x = h * ((double)i - 0.5);
    sum += (4.0 / (1.0 + x*x));
}
mypi = h * sum;
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
           MPI_COMM_WORLD);
if (myid == 0){
    printf("pi is approximately %.16f, Error is
           %.16f\n",pi, fabs(pi - PI25DT));
} \\else
} \\while
MPI_Finalize();
return 0;
} \\ main
```



# Impianti Informatici



Grid computing: infrastrutture

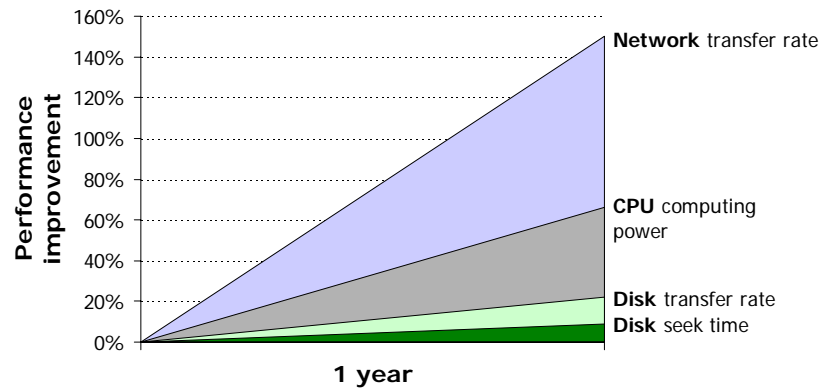


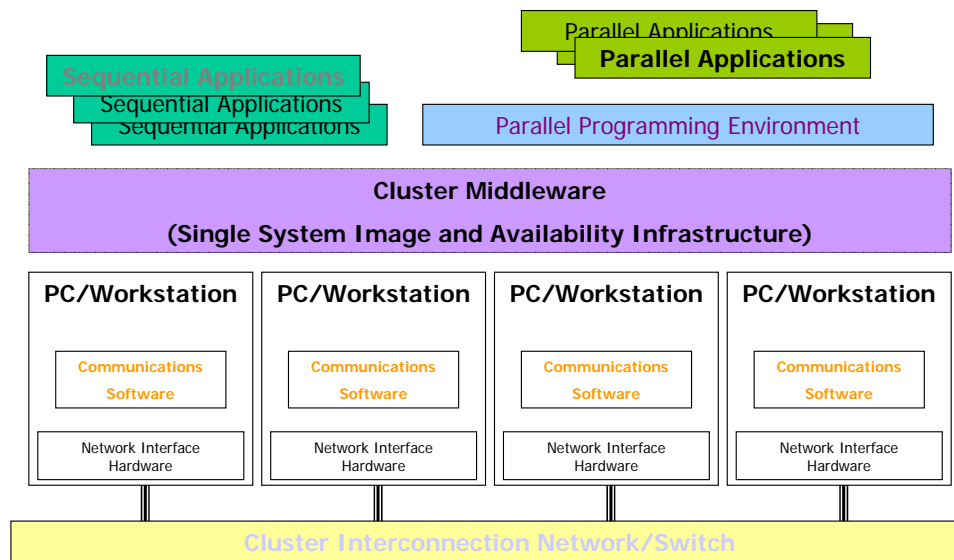


Potenza computazionale in continua crescita → supercomputer

Sempre maggiore banda (bandwidth) nei sistemi di comunicazione con interessanti opportunità di interconnettere nodi computazionali

**Cluster:** insieme di PC (o Workstations), interconnessi da una rete e collaboranti tra loro







### Parallel Programming Environments and Tools

- Threads (PCs, SMPs, NOW..)
  - POSIX Threads
  - Java Threads
- MPI
  - Linux, NT, almost all UNIX
- PVM
- Software DSMs (Distributed Shared Memory)
- Compilers
  - C/C++/Java
  - Fortran 77/90 - HPF
- RAD (Rapid Application Development tools)
  - GUI based tools for Parallel Program modeling
- Debuggers
- Performance Analysis Tools
- Visualization Tools



### Pros:

- High Performance (per node)
- Expandability and Scalability
- High Throughput
- High Availability
- low cost, easily scalable, “easy” setup

### Cons:

- performance are (generally) lower than those of a supercomputer
- network is (generally) the bottleneck



### Application Target

- High Performance (HP) Clusters
  - Grand Challenging Applications
- High Availability (HA) Clusters
  - Mission Critical Applications

### Node Ownership

- Dedicated Clusters
- Non-dedicated clusters
  - Adaptive parallel computing
  - Communal multiprocessing



### Node Hardware

- Clusters of PCs (CoPs)
  - Piles of PCs (PoPs)
- Clusters of Workstations (COWs)
- Clusters of SMPs (CLUMPs)

### Node Operating System

- Linux Clusters (e.g., Beowulf)
- Solaris Clusters (e.g., Berkeley NOW)
- NT Clusters (e.g., HPVM)
- AIX Clusters (e.g., IBM SP2)
- SCO/Compaq Clusters (Unixware)
- Digital VMS Clusters
- HP-UX clusters
- Microsoft Wolfpack clusters

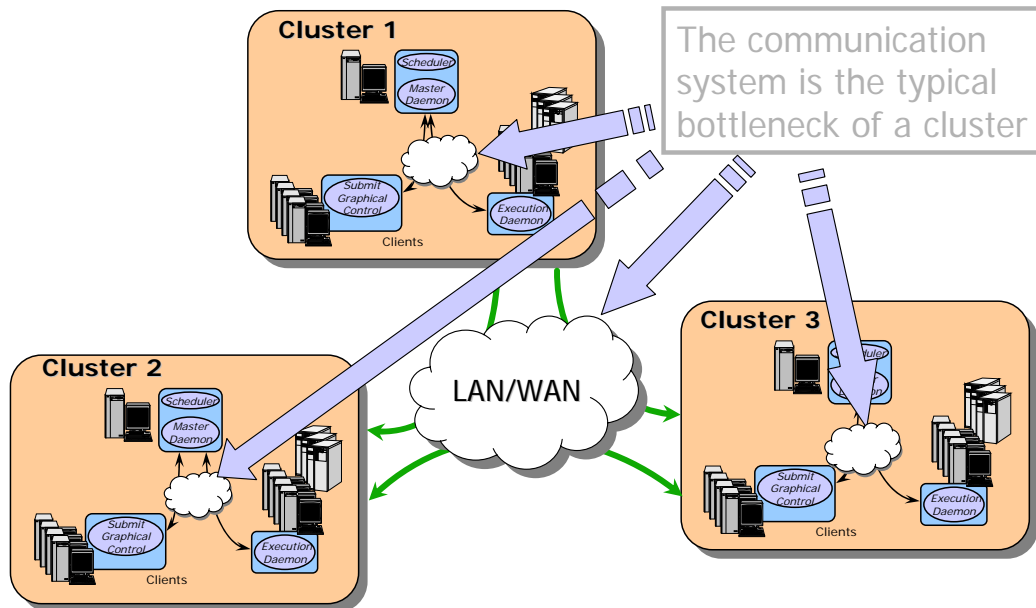


### Node Configuration

- Homogeneous Clusters
  - All nodes will have similar architectures and run the same OSs
- Heterogeneous Clusters
  - All nodes will have different architectures and run different OSs

### Levels of Clustering

- Group Clusters (#nodes: 2-99)
  - Nodes are connected by SAN like Myrinet
- Departmental Clusters (#nodes: 10s to 100s)
- Organizational Clusters (#nodes: many 100s)
- National Metacomputers (WAN/Internet-based)
- International Metacomputers (Internet-based, #nodes: 1000s to many millions)
  - Metacomputing / Grid Computing
  - Web-based Computing
  - Agent Based Computing



Scheduler

Master Daemon

Submit Graphical Control

Execution Daemon





### Network vs. computer performance

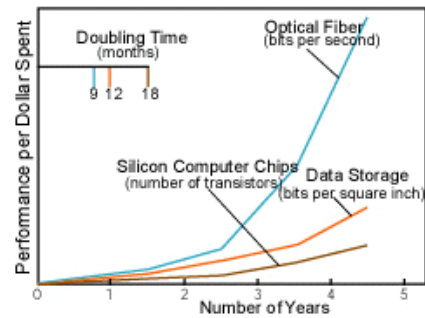
- La velocità dei computer raddoppia ogni 18 mesi
- La velocità della rete raddoppia ogni 9 mesi

### 1986 to 2000

- Computers: x 500
- Networks: x 340.000

### 2001 to 2010

- Computers: x 60
- Networks: x 4000



Moore's Law vs. storage improvements vs. optical improvements. Graph from *Scientific American* (Jan-2001) by Cleo Vilett, source Vined Khoslan, Kleiner, Caufield and Perkins.



“When the network is as fast  
as the computer's internal  
links, the machine  
disintegrates across the net  
into a set of special purpose  
appliances”

(George Gilder)



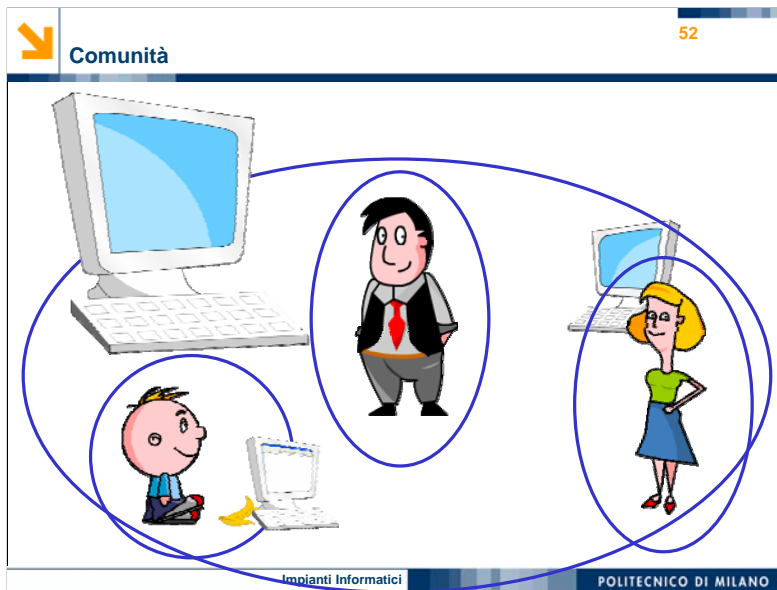
*Grid Computing* è un paradigma/infrastruttura che consente la condivisione, selezione e aggregazione di risorse distribuite geograficamente...

- **Computers** – PCs, workstations, clusters, supercomputers, laptops, notebooks, mobile devices, PDA, etc.
- **Software** – e.g., ASPs renting expensive special purpose applications on demand
- **Catalogued data and databases** – e.g. transparent access to human genome database
- **Special devices/instruments** – e.g., radio telescope – SETI@Home searching for life in galaxy
- **People/collaborators**



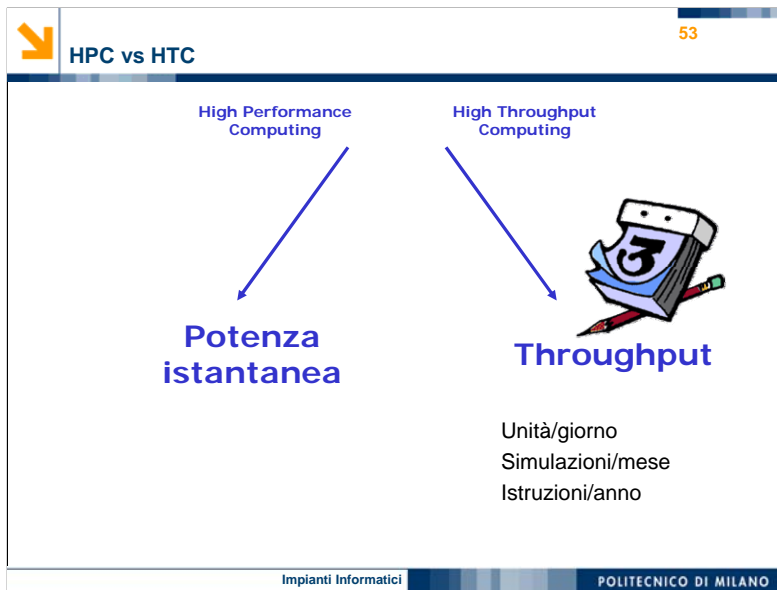
*[depending on their availability, capability, cost, and user QoS requirements]*

...per risolvere applicazioni/problemi di “grande scala” (large-scale problems).



Il punto di forza di una qualsiasi comunità è che

1. le capacità e le competenze dei diversi elementi
2. che ne fanno parte, vengono migliorate e potenziate
3. grazie alla collaborazione. Il grande progresso nell'area delle comunicazioni tra computer,
4. permette di integrare i sistemi stand-alone in comunità multi-computer, con i relativi vantaggi e possibilità.



Per moltissimi scienziati impegnati nel campo della sperimentazione, il progresso scientifico e la qualità della ricerca sono fortemente collegate

1. al throughput computazionale. Ovvero,
2. essi sono scarsamente interessati alla potenza di elaborazione istantanea,
3. detta High Performance Computing, ma quello che davvero importa è l'ammontare di computazione che il sistema può eseguire in un mese, oppure in un anno,
4. ovvero l'High Throughput Computing
5. Per le loro esigenze, la potenza di calcolo viene quindi misurata come
6. il numero di unità elaborate in un giorno,
7. le simulazioni eseguite in un mese,
8. oppure il numero di istruzioni compiute nel corso dell'anno.

↘ Anni 90: The Grid
54

Grid Computing

Infrastruttura di computazione distribuita

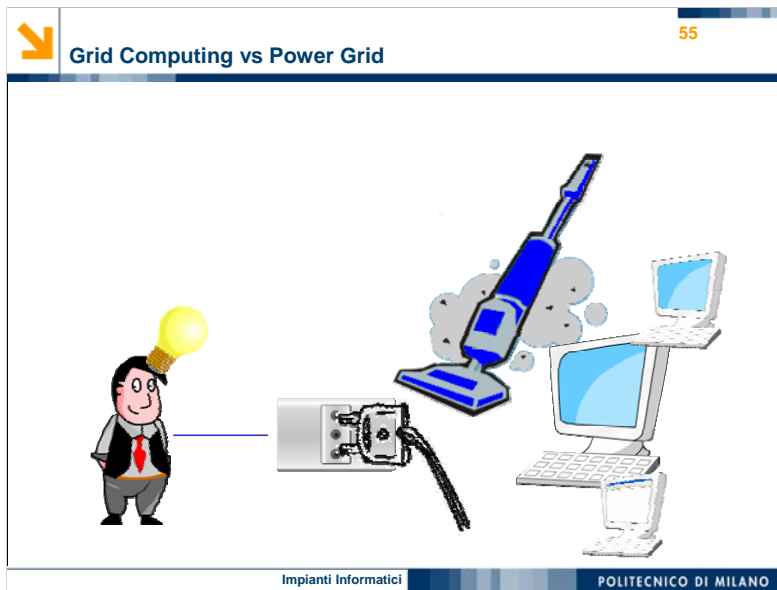
Supporto a problemi complessi

The diagram illustrates the concept of Grid Computing. On the left, a cartoon character in a suit stands with a glowing yellow lightbulb above his head, representing an idea or problem. Three lines radiate from the character towards a central network of three computer monitors. The monitors are arranged in a triangular pattern, with lines connecting them to each other, forming a mesh or 'grid' structure. This visualizes the distributed nature of the computing infrastructure.

Impianti Informatici
POLITECNICO DI MILANO

Si è iniziato a parlare di Griglia una decina di anni fa.

1. Il termine è stato coniato nella metà degli anni 90, per denotare
2. un'infrastruttura di computazione distribuita
3. per il supporto di problemi ingegneristici e scientifici di natura particolarmente complessa. Le promesse della griglia erano di
4. cambiare fundamentalmente il modo di pensare e usare la potenza di elaborazione. Secondo una visione piuttosto utopistica, ma che rende bene l'idea delle potenzialità del Grid, l'infrastruttura permetterebbe di connettere multiple griglie computazionali, sia a livello nazionale che regionale,
5. creando una sorgente universale di potenza computazionale in grado di supportare una classe nuova e molto esigente di applicazioni.



Le aspettative del grid computing sono numerose e non semplici da realizzare. L'ideale, seppur inverosimile, sarebbe di avere una grid computing

1. paragonabile alla griglia elettrica. Così' come
2. un elettrodomestico che necessita di energia viene attaccato alla presa di corrente,
3. analogamente una qualsiasi esigenza computazionale potrebbe essere soddisfatta semplicemente attaccandosi alla griglia. Non importa
4. da dove giunga la potenza di elaborazione, quello che conta e' il risultato, che puo' essere l'esecuzione di un applicativo, la richiesta di calcoli computazionalmente complessi, l'esigenza di determinate informazioni presenti in qualche posto nella griglia, magari a seguito di un processo di elaborazione.

➤
Obiettivi
56

Scalabilità


Stabilità

Cost-saving

- Performance
- Simulazioni
- Modelli complicati

Trasparenza

Accessibilità

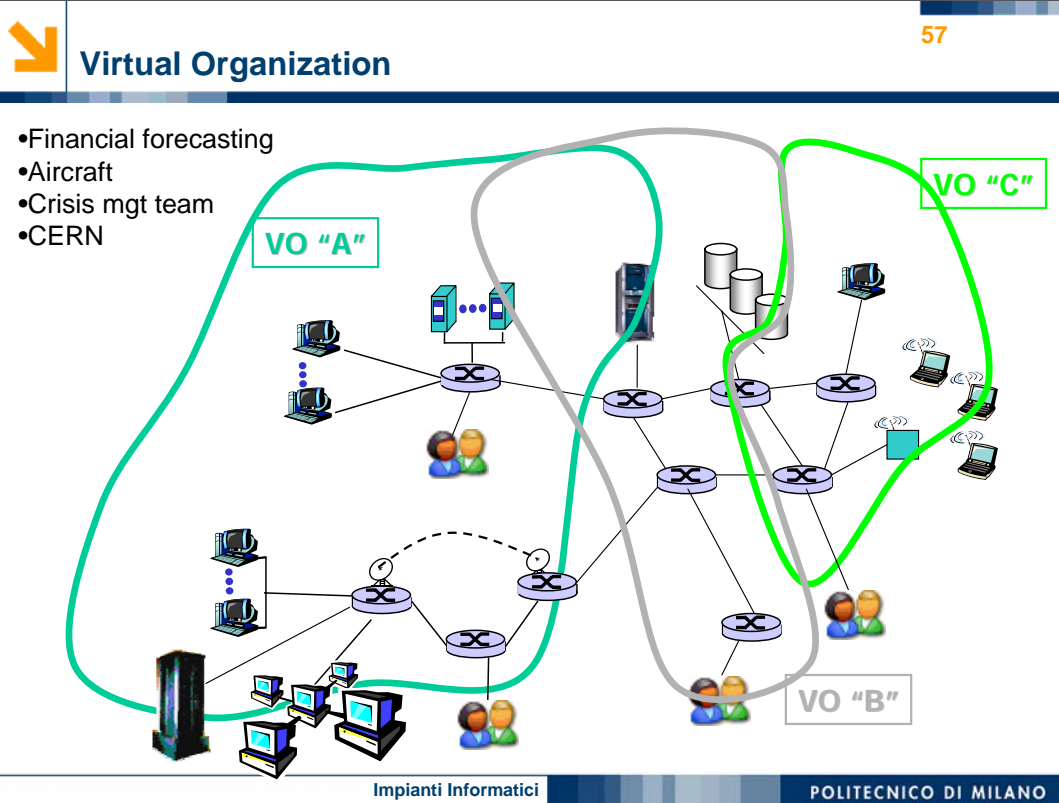


Impianti Informatici
POLITECNICO DI MILANO

Dalla griglia ci si aspetta:

1. Una forte scalabilità, ma che, nonostante le estese dimensioni, mantenga il sistema ugualmente
2. Stabile. Un altro fondamentale obiettivo è
3. la possibilità di usare la griglia come fonte di risparmio
4. Ad esempio per eseguire calcoli in tempi ridotti, con la capacità
5. Di fare simulazioni altamente realistiche e sofisticate dei fenomeni, e quindi di conoscere l'efficacia di un progetto o prodotto prima ancora di realizzarlo.
6. Chi realizza applicazioni non deve occuparsi di trovare o comprare l'hardware necessario all'esecuzione, perché la griglia fornisce le risorse necessarie a tale scopo.
7. Le scelte nel campo del grid computing spingono verso una griglia altamente accessibile dall'utente medio, in cui non occorra essere un super uomo per poter usare un super computer.





Il termine Virtual Organization è stato coniato per descrivere un'organizzazione distribuita

- il cui compito è gestire l'accesso alle risorse che possono essere accedute attraverso la rete. Successivamente ogni implementazione o progetto inerente la griglia ha dato la sua propria e più specializzata versione.
- La VO va oltre i limiti delle organizzazioni fisiche, e può comprendere risorse appartenenti a domini amministrativi, enti o organismi differenti tra loro.
- Può inoltre essere una infrastruttura temporanea, e
- Una risorsa appartiene fisicamente ad una certa organizzazione, ma può far capo a molteplici Virtual Organization, ciascuna gestita con le proprie policy



Collaborazione di individui/istituzioni

Condivisione di risorse

Dinamiche

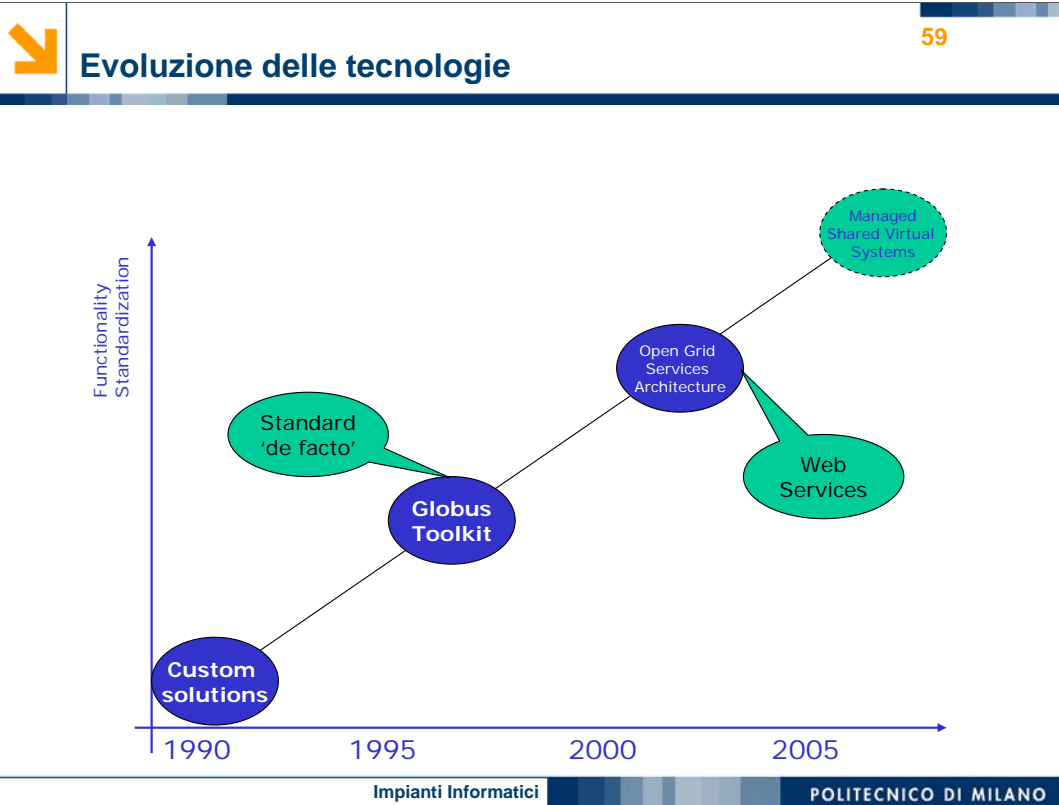
- VO che nascono e cessano
- Risorse che si uniscono e lasciano una VO
- Risorse che cambiano stato o falliscono

Obiettivo comune da raggiungere

Non devono interferire tra loro

Le Virtual Organization consentono ad individui ed istituzioni di

1. collaborare ed interagire fra loro. Essi possono
2. condividere le risorse che dispongono ed utilizzare quelle messe a disposizione dagli altri, nei limiti delle politiche stabilite tanto dalla Virtual Organization, quanto dal proprietario della risorsa.
3. Le VO sono altamente dinamiche in natura. Questo dinamismo si evidenzia a diversi livelli:
4. Le virtual organization possono essere temporanee, ovvero costituite per determinati obiettivi e poi rimosse.
5. All'interno di una VO le risorse vengono aggiunte e tolte, a seconda delle necessità di elaborazione e della disponibilità delle stesse.
6. Tanto il crash o il guasto di un elemento, quanto la decisione di non condividere più una risorsa sulla griglia, oppure un cambiamento dello stato, deve risultare il più trasparente possibile alla griglia
7. Ogni organizzazione ha un preciso obiettivo da raggiungere, che implica come ciascuna risorsa al suo interno viene utilizzata,
8. tenendo comunque conto che tale risorsa può appartenere a virtual organization differenti, con diverse finalità, e che non devono interferire tra loro



Quando si è cominciato a parlare di Grid Computing negli anni '90,

1. Le prime soluzioni erano soluzione custom, ovvero implementazioni fatte ad hoc dalle singole organizzazioni che necessitavano di potenza computazionale per eseguire applicazioni complesse. Esse erano praticamente dei grandi cluster di computer. Non c'era praticamente nulla di standardizzato, fino alla fine degli anni 90
2. Grazie al Globus Toolkit, che pur non essendo nulla di ufficiale,
3. è considerato lo standard de facto e il punto di riferimento per i sistemi grid
4. L'evoluzione delle tecnologie è andata verso il modello dei
5. Web Services, introdotto tramite OGSA
6. Il futuro va verso una sempre maggiore standardizzazione dei protocolli e dei modelli utilizzati, cosicché diverse e molteplici griglie possano interagire e collaborare tra loro come un'unica grande grid.



**Middleware** che rendano il sistema *Grid enabled*



**Security mechanisms** che permettano l'accesso alle risorse solo dai chi è autorizzato

**Programming tools** che rendano le applicazioni Grid enabled

### Tools

- per tradurre i requisiti di una applicazione nei requisiti di computer, reti e storage
- per *resource discovery/monitoring, trading, composition, scheduling* e distribuzione dei jobs e raccolta dei risultati



Per creare una Griglia computazionale occorre disporre di

1. Middleware che renda il sistema Grid-enabled; è un ulteriore livello di astrazione che renda la griglia trasparente
2. Servono meccanismi di sicurezza che controllino l'accesso alle risorse, considerando la vastità del dominio e la sua eterogeneità
3. Occorrono degli strumenti o linguaggi di programmazione per le applicazioni da eseguire sulla griglia, e
4. tool in grado di convertire le esigenze di una applicazione nei requisiti delle singole risorse, quali processori, reti, dispositivi di storage.
5. Altri servizi riguardano il discovery ed il monitoring delle risorse, sia a livello individuale che collettivo,
6. E la ripartizione e pianificazione del carico computazionale tra vari componenti



“My” users need to be able to use “my” computing resources in tandem with resources at other sites.

Servono meccanismi di *sharing* delle risorse che:

- Garantiscano security e policy sia per i *resource owners* che per i *resource users*
- Siano sufficientemente flessibili da gestire molteplici tipi di risorse e di modalità di condivisione.
- Scalino col numero di risorse, di partecipanti, di componenti (es: programmi)
  - Operino efficientemente anche di fronte a molti dati e computazioni



Le applicazioni richiedono risorse (compute power, storage, data, instruments, displays) presenti in molti siti

Esigenza:

- Astrazioni e modelli per aggiungere velocità/robustezza/... allo sviluppo
  - Tools per facilitare lo sviluppo delle applicazioni e la ricerca dei bug comuni
  - *Code/tool sharing* per permettere il riuso dei componenti software sviluppati da altri



*The open source Globus Toolkit is a fundamental technology for “the Grid”, letting people share computing power, databases and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy.*

from [www.globus.org](http://www.globus.org), “About the Globus Toolkit”

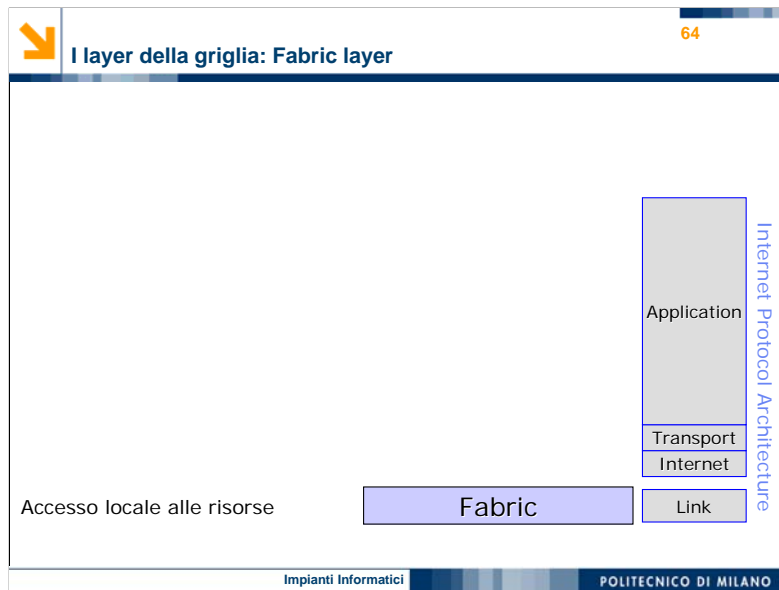
Progetto iniziato nel 1998 (ver. 1.0). Nel 2002 e 2003 rilasciate le versioni 2.0 e 3.0, rispettivamente.

Versione corrente: 4.0 released in April 2005

Globus Toolkit è lo standard “de facto” e il riferimento per i sistemi grid

Sopportato da molti istituti e compagnie:

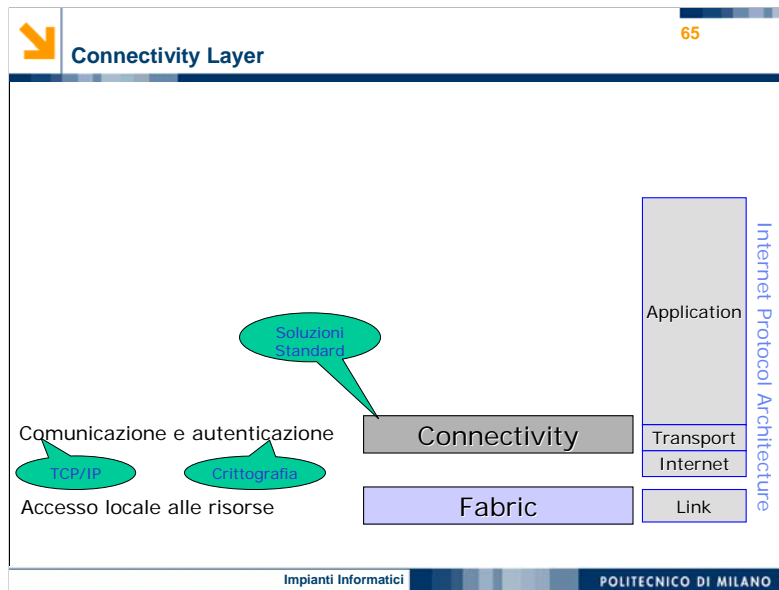
- Avaki, dataSynapse, Entropia, Fujitsu, HP, IBM, NEC, Oracle, Platform, Sun and united Devices



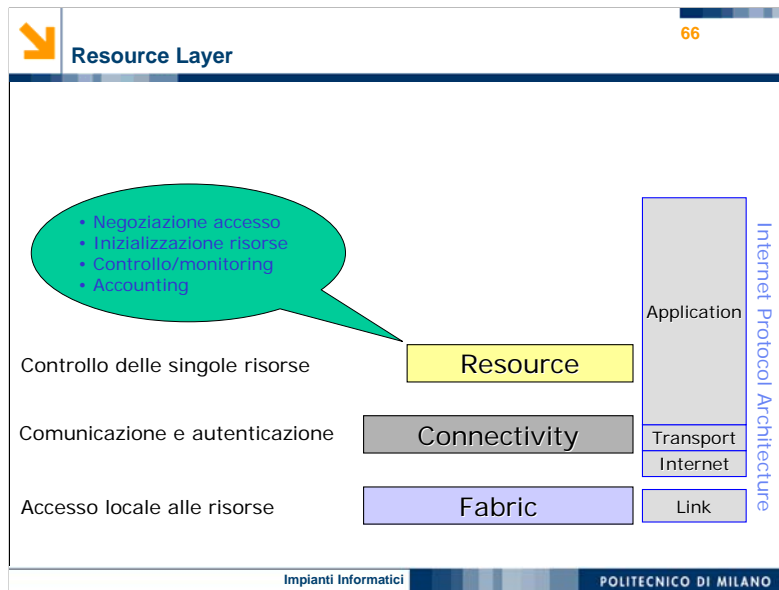
L'architettura della Griglia computazionale può essere analizzata secondo un modello a livelli,

1. con le relative analogie all'architettura di Internet
2. Il livello inferiore è il fabric layer; include i protocolli e le interfacce che forniscono l'accesso alle risorse che devono essere condivise, includendo computer, risorse computazionali, sistemi di storage, cataloghi, risorse di rete, sensor network, ma anche entità logiche come file system o cluster di pc.
3. Questo livello controlla le cose localmente, gestendone l'accesso meccanismi di priorità, di prenotazione delle risorse, di introspezione per estrarre dai componenti le caratteristiche e lo stato.

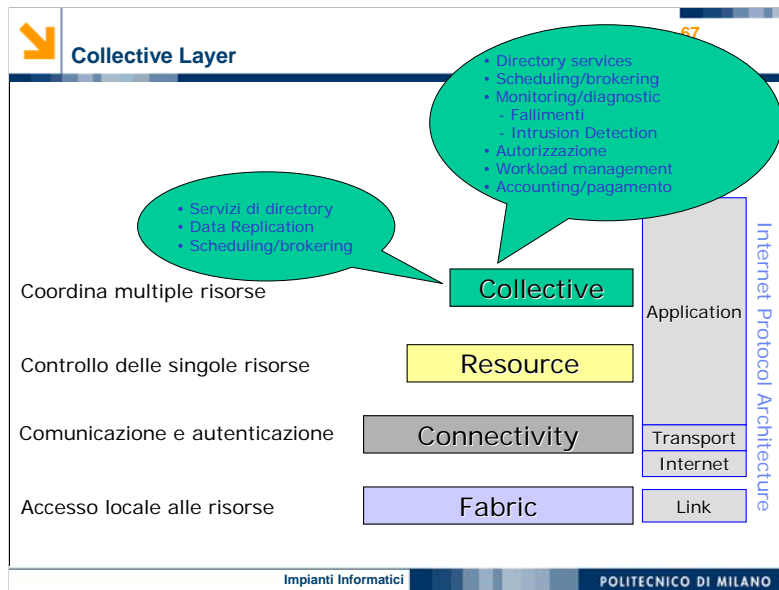




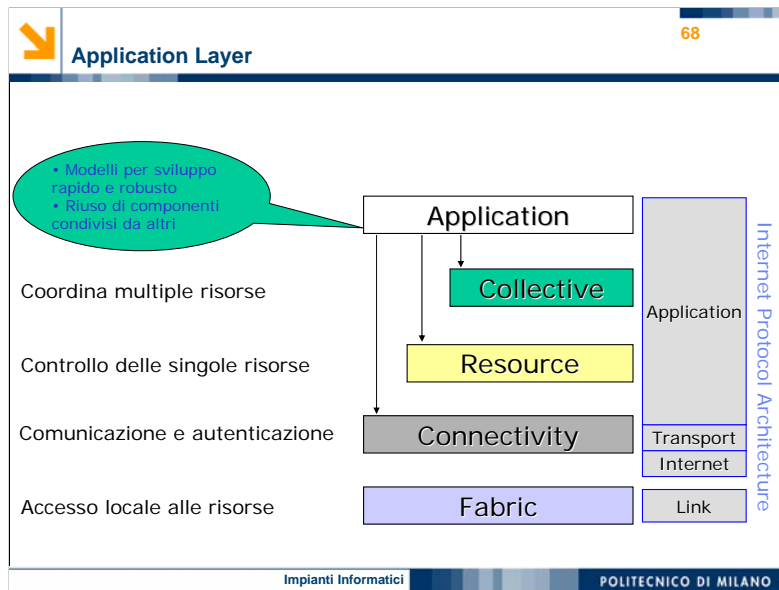
1. Il livello connectivity si riferisce agli aspetti di
2. comunicazione e autenticazione. Occorrono perciò dei protocolli per permettere lo scambio di dati tra le risorse presenti a livello Fabric, con servizi di trasporto, routing e naming.
3. Finora i protocolli usati sono quelli standard per Internet, in particolare basati sullo stack TCP/IP. Probabilmente le esigenze di trasferimento di ingenti quantità di dati porterà allo sviluppo di protocolli specifici per tali scopi.
4. I protocolli di autenticazione mettono a disposizione dei meccanismi crittografici per identificare utenti e risorse.
5. La complessità del problema rende importante l'uso di soluzioni standard.



1. Il livello Resource permette l'interazione con risorse e servizi remoti.
2. Definisce i protocolli per la negoziazione sicura,
3. L'inizializzazione delle risorse,
4. Il controllo del loro uso e il monitoring, e infine
5. L'accounting di risorse e operazioni, anche nell'ottica di un uso a pagamento dei servizi.
6. Questo livello interagisce con le singole risorse condivise.



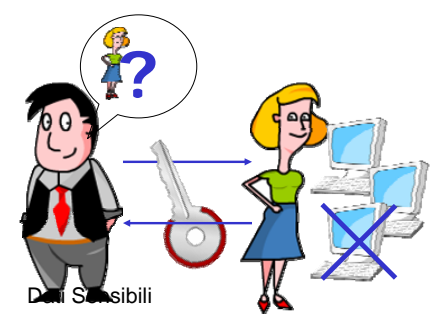
1. Il livello Collective riguarda i protocolli responsabili di
2. catturare le interazioni tra le collezioni di risorse,
3. ad esempio servizi di directory,
4. di replica dei dati, o
5. di scheduling, di brokering, o ancora
6. di monitoraggio di insiemi di risorse, anche a fini diagnostici
7. per individuare fallimenti
8. o per Intrusion Detection.
9. Il Collective Layer può inoltre comprendere server di autorizzazione,
10. O servizi di pagamento o accounting.



Il livello più alto dell'architettura è relativo

1. alle applicazioni che usano la griglia, le quali usano le interfacce messe a disposizione dal livello
  2. Collective,
  3. Resource
  4. e connectivity.
- Le applicazioni richiedono risorse disponibili in siti differenti e necessitano quindi di
5. astrazioni e modelli per uno sviluppo rapido e allo stesso tempo robusto
  6. Sempre considerando il punto di forza della griglia, ovvero la capacità di condividere risorse di qualsiasi natura, quindi anche l'uso di codice sviluppato da altri

La sicurezza
69



Autenticazione  
Crittografia  
Autorizzazione

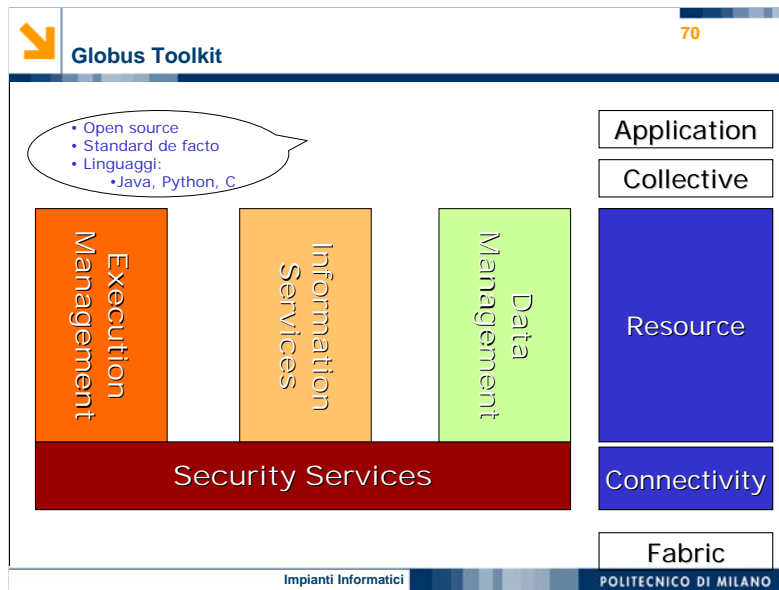
**Dati Sensibili**  
 Risorse appartenenti a diversi domini amministrativi  
 VO molto dinamiche ed estese  
 Availability e Reliability

- Protocolli standard e ben testati

Impianti Informatici
POLITECNICO DI MILANO

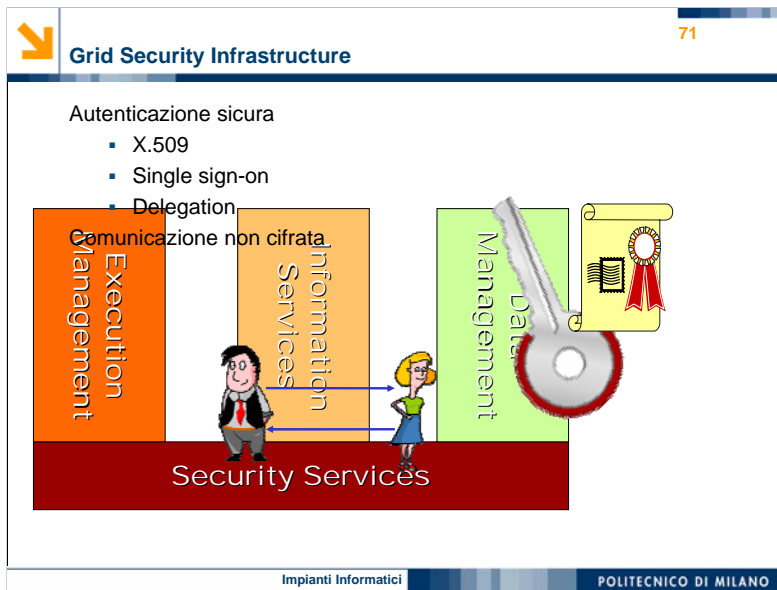
La sicurezza riguarda quei meccanismi per

1. stabilire l'identità degli utenti,
  2. Per la protezione delle comunicazioni, e la determinazione
  3. dei permessi di utilizzo.
- La questione sicurezza assume un particolare rilievo nel Grid Computing.
4. Infatti Le risorse presenti all'interno di una organizzazione potrebbero riguardare dati altamente sensibili; un utente della griglia potrebbe non essere intenzionato a condividere tali dati, pur volendone condividere altri, oppure potrebbe volerli condividere sotto certe garanzie e verso determinati utilizzatori.
  5. Le risorse sono inoltre distribuite tra domini amministrativi distinti, ciascuno con le proprie politiche e i propri meccanismi di accesso
  6. L'insieme di risorse usato per una computazione può essere particolarmente ampio, ma soprattutto dinamico e non predicibile.
  7. Infine, dato che l'uso delle stesse risorse da parte di molteplici applicazioni, o anche da differenti Virtual Organization con diversi obiettivi, occorre garantire che la griglia sia sempre disponibile ed utilizzabile,
  8. Il che richiede l'uso di modelli e protocolli standard, ben testati e maturi.



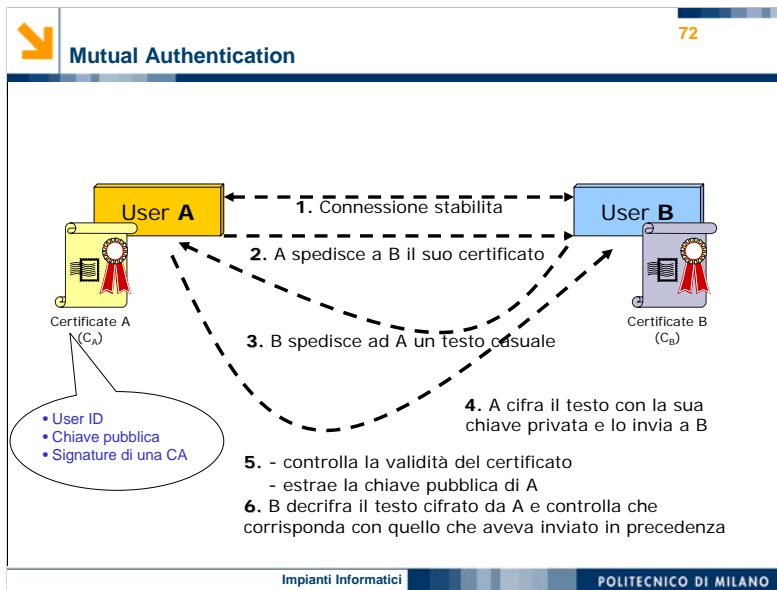
Il Globus Toolkit è il riferimento

1. open source ufficiale per il grid computing,
2. che permette alle persone di condividere potenza computazionale, database e altre risorse all'interno di una virtual organization, senza che venga meno l'autonomia locale.
3. I linguaggi supportati dall'ultima versione, la 4.0, sono java, c e python.
4. L'infrastruttura di globus si incastra nei livelli connectivity e resource dell'architettura grid,
5. Offrendo servizi inerenti la sicurezza,
6. l'esecuzione di job,
7. Servizi informativi e di
8. Trasferimento e gestione dei dati



L'infrastruttura GSI, Grid Security Infrastructure, è quella porzione di Globus che fornisce i servizi

1. di sicurezza in supporto alla Griglia
2. In particolare fornisce elementi per l'autenticazione sicura,
3. tramite un meccanismo a chiave pubblica basato sul protocollo X.509, con in più
4. Il supporto al single sign-on e alla
5. Delegation. Una volta che l'autenticazione è stata effettuata,
6. la comunicazione continua senza crittografia per ragioni di prestazione. Può cmq essere abilitata.



L'autenticazione avviene mutuamente

1. prima tra A e B, poi tra B e A.
2. Una volta stabilita la connessione tra i due utenti, o più in generale, tra le due risorse,
3. A spedisce a B il suo certificato, contenente, tra le altre cose,
4. Un suo identificativo
5. La sua chiave pubblica
6. E la firma digitale della Certificate Authority garante
7. Quindi B genera un testo casuale e lo invia ad A,
8. Il quale lo cifra con la sua chiave privata e lo rinvia a B.
9. B, a questo punto, controlla la validità del certificato di A tramite la firma digitale della Certificate Authority
10. Estrae la chiave pubblica di A e
11. Decifra il testo. Se corrisponde a quello originariamente inviatogli A è stato autenticato con successo
12. Il processo si ripeterà in modo identico partendo dall'utente B.



➔
**Delegation e Single Sign-on**
73

---


Single sign-on

Delegation: attribuire a qualcun altro i propri diritti

- Sito delega utenti
- Utente delega un servizio

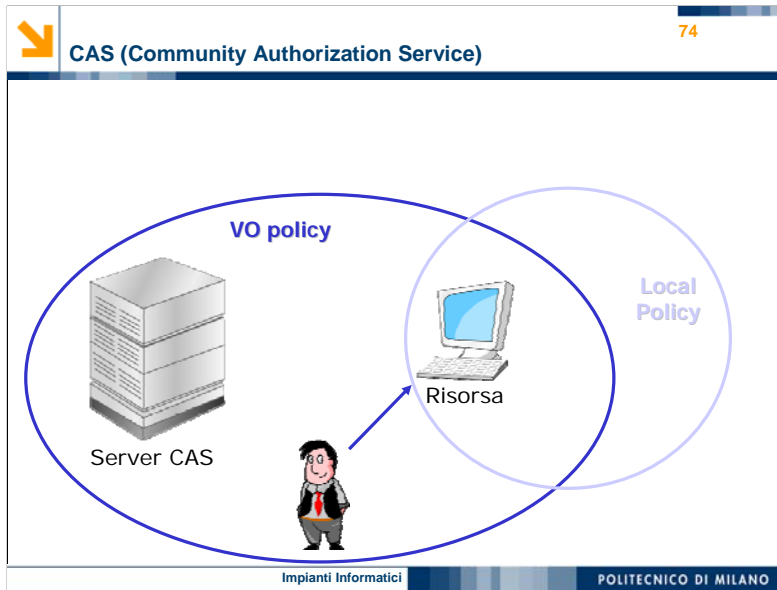
User Proxy

- Certificato X.509 per il proxy (e coppia di chiavi)
  - Certificato dalla signature dell'utente
- Ha diritti temporanei
- Limita l'esposizione della chiave privata dell'utente

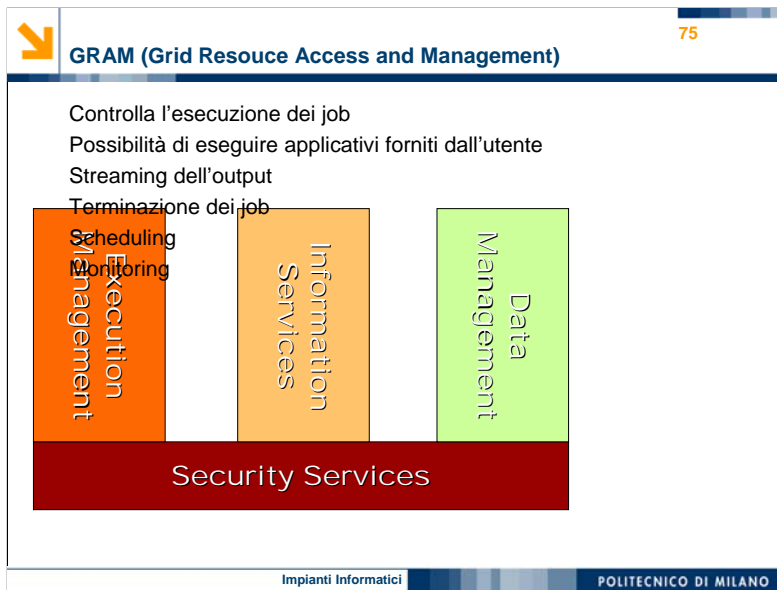


Impianti Informatici
POLITECNICO DI MILANO

1. Il single sign-on indica la possibilità di potersi autenticare una singola volta e poi poter usare tutti i servizi e le risorse della Virtual Organization senza doversi identificare nuovamente
2. Il meccanismo della delegation si riferisce all'atto di dare ad un'organizzazione, una persona o un servizio i diritti per agire nella vece di un'altra risorsa
3. Un sito può delegare la responsabilità per gli utenti che possono accedere alle sue risorse per gestire il sistema, così come un'organizzazione può delegare i diritti ad un utente
4. Un utente può delegare la sua autenticazione ad un servizio che gli consenta di eseguire certi programmi su un sito remoto
5. Tipicamente, si utilizza un PROXY, che si occupa di garantire tanto il single sign-on, quanto la delegation.
6. Il meccanismo della mutua autenticazione si complica leggermente perché ci sono in gioco una nuova coppia di chiavi e un certificato X.509 per il proxy,
7. questa volta certificato dalla firma digitale dell'utente. Quindi l'utente B dialogherà direttamente con il proxy, verificando la validità del suo certificato a ritroso, prima controllando la signature dell'utente, quindi quella della Certificate Authority.
8. Solitamente il proxy ha una validità temporanea
9. e il grande vantaggio di limitare l'esposizione all'esterno della chiave privata dell'utente

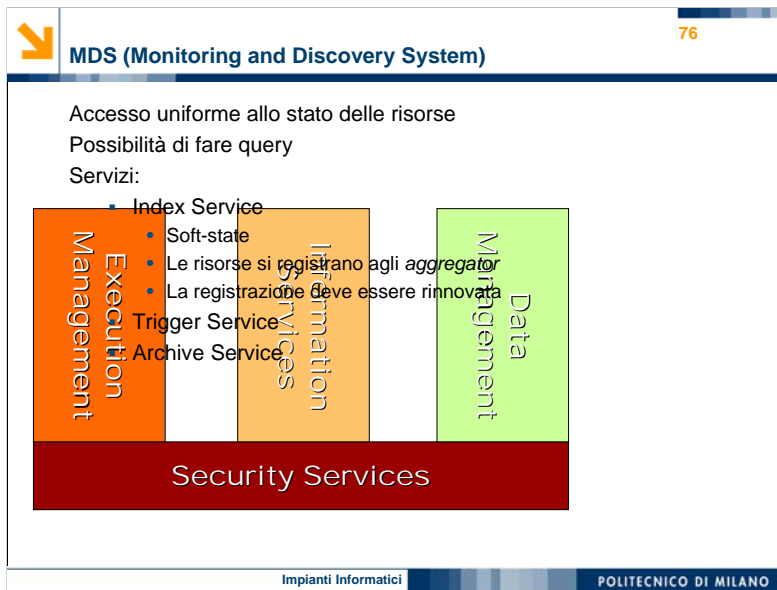


1. I server CAS permettono
2. ad una virtual organization di esprimere le politiche di accesso ed uso
3. delle risorse distribuite in un numero molteplice di siti.  
Le politiche da applicare alle risorse saranno un
4. sottoinsieme dei permessi delle Virtual Organization e di quelli delle local policy.

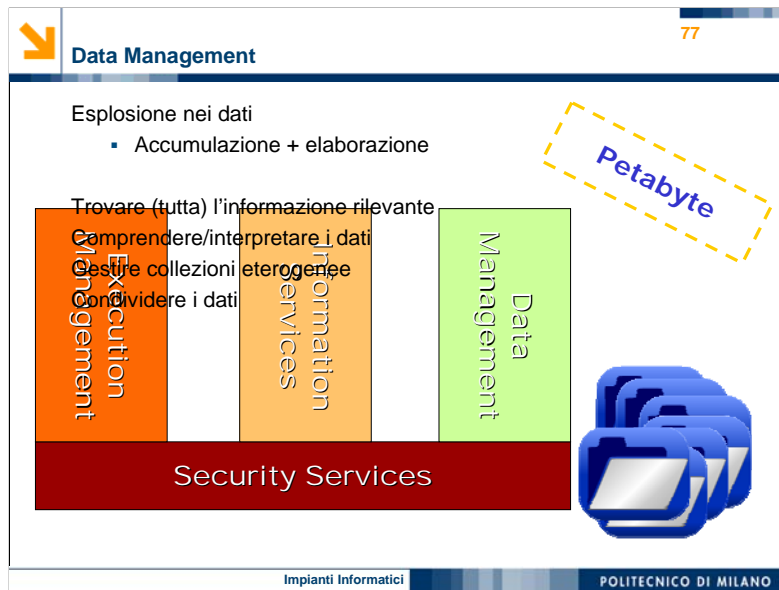


GRAM è un'architettura di Globus che rappresenta un'interfaccia


1. per la sottomissione e la gestione di JOB sulla griglia.
2. Esso permette di controllare, fra le altre cose,
3. che un job venga eseguito una ed una sola volta. Infatti, dato che lo stato di una risorsa può essere condiviso tra più operazioni, è importante che il job venga completato e che non venga effettuato più volte.
4. L'utente deve poter fornire i programmi da eseguire sulla griglia e,
5. Deve essere possibile l'accesso all'output mentre i job sono in esecuzione.
6. e terminare un job, ad esempio perché consuma molte risorse,
7. Oltre a poterne pianificare l'esecuzione e
8. Monitorare dettagliatamente lo stato.



1. MDS è un insieme di servizi per il monitoring e il discovery delle risorse e dei servizi della griglia.
2. L'utilità è quella di avere un accesso uniforme a quali risorse sono disponibili, qual è il loro livello di utilizzo, così da controllare il sistema e consentirne una gestione dinamica.
3. MDS consente agli utenti di scoprire quali risorse fanno parte della Virtual Organization e offre la possibilità di effettuare query e sottoscrizioni agli indici.
4. Esistono tre tipi di servizi: l'index service, che mantiene un insieme di risorse della griglia registrate.
5. Delle risorse viene mantenuto il cosiddetto soft-state:
6. ci sono una serie di aggregator che collezionano periodicamente le informazioni sullo stato di tutte le risorse registrate.
7. La registrazione ha un tempo di vita, se non viene rinnovata scade e l'aggregator non tiene più traccia.
8. Il Trigger service è una lista di regole che fanno scattare determinate azioni quando i dati corrispondono a certi criteri, ad esempio potrebbero inviare una email all'amministratore in caso di sovraccarico del sistema, oppure di completamento di una operazione.
9. L'ultimo tipo di servizi è l'Archive Service, che semplicemente memorizza in un database lo stato delle risorse monitorate.



1. I dati assumono un ruolo sempre importante, soprattutto all'interno della griglia, dove spesso assumono dimensioni non facilmente trattabili.
2. Vi è una vera e propria esplosione nella crescita dei dati, molto superiore rispetto alla semplice accumulazione degli stessi,
3. dovuta alla loro elaborazione, filtraggio, manipolazione.
4. Si parla sempre più spesso di Petabyte.
5. Le difficoltà sono dunque nel trovare l'informazione rilevante,
6. nel comprendere ed interpretare le informazioni,
7. nel gestire dati memorizzati in maniera estremamente eterogenea
8. e nel condividere questi dati all'interno della griglia.


 GridFTP
78

---

Estensione del protocollo FTP

Estensioni

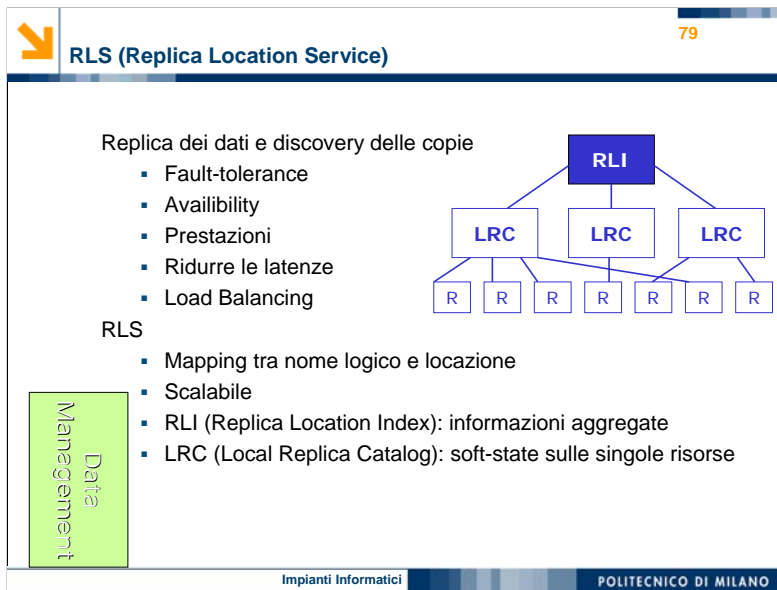
- Sicurezza
- Accesso parziale ai file
- Parallelismo
  - Striped server mode
    - Multipli stream
    - Finestra TCP allargata
    - Meccanismi di resume del trasferimento



Data Management

Impianti Informatici
POLITECNICO DI MILANO

1. Gridftp è un protocollo basato sul famoso FTP, usato per il trasferimento di file tra piattaforme eterogenee.
2. In particolare le estensioni riguardano l'integrazione di protocolli sicuri, usando GSI per garantire autenticazione, integrità e confidenza.
3. Consente L'accesso anche parziale ai file
4. E La gestione del parallelismo,
5. potendo operare il Striped Server Mode, che utilizza
6. multipli stream TCP tra sorgente e destinazione,
7. Una finestra TCP grande per ridurre l'overhead e
8. Meccanismi per riprendere un trasferimento interrotto, utile considerando le dimensioni spesso significative dei file in gioco



1. Il replica location Service è un registro distribuito che memorizza dove si trovano le copie dei dati e ne permette il discovery.
2. La replica dei dati è fondamentale per garantire fault tolerance
3. assicurare la disponibilità del sistema
4. Migliorare le prestazioni
5. Evitare latenze e
6. Consentire un adeguato bilanciamento del carico
7. RLS mantiene un mapping tra il nome logico e il target fisico
8. È importante che sia in grado di gestire milioni di oggetti e utenti, cosicché, per garantirne la scalabilità, è gestito
9. In maniera gerarchica, tramite i Replica Location Index e i Local Replica Catalog. I primi, gli RLI, rappresentano le informazioni aggregate inerenti uno o più LRC.
10. Questi mantengono uno soft-state delle copie dei dati e aggiornano periodicamente gli RLI.



 POLITECNICO DI MILANO

## Impianti Informatici



Altre soluzioni grid





Boinc is a platform to support *embarrassingly parallel* distributed computing based on the client/server paradigm

Freely distributed

It supports applications written in different languages

- C/C++
- Fortran

Digital signature used to protect against viruses

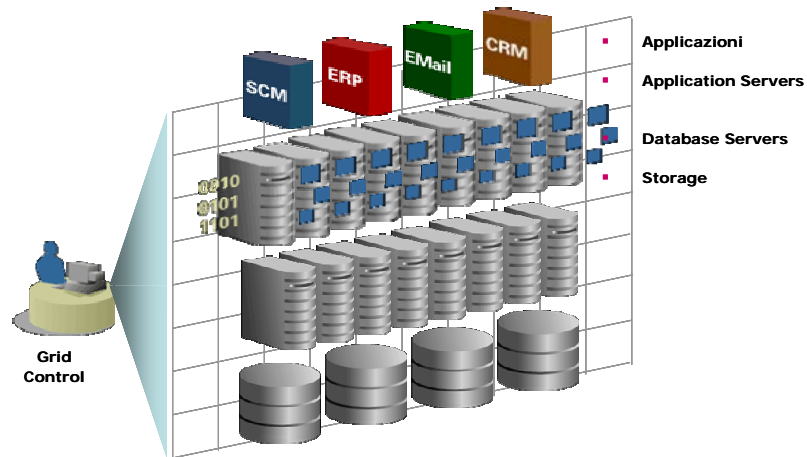
Execution monitoring tools

Project based on BOINC

- [Climateprediction.net](http://Climateprediction.net): study climate change
- [Einstein@home](http://Einstein@home): search for gravitational signals coming from pulsars
- [LHC@home](http://LHC@home): improve the design of the CERN LHC particle accelerator
- [Predictor@home](http://Predictor@home): investigate protein-related diseases
- [SETI@home](http://SETI@home): Look for radio evidence of extraterrestrial life
- [Cell Computing](http://Cell Computing) (Japanese; requires nonstandard client software)



Coordinated use of many servers  
acting as one large computer.



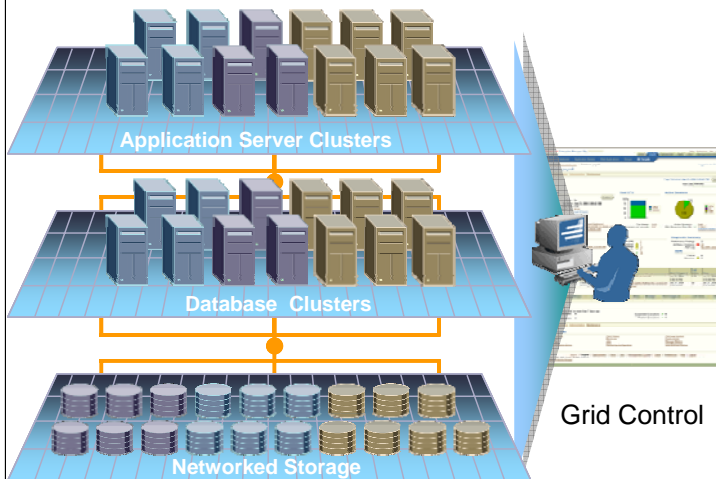
In effetti tradizionalmente in Grid e' nato in ambito scientifico e poi si e' spostato in progetto realizzati ad hoc per grandi realta'...ecco invece che Oracle ha portato questo concetto all'interno ed alla portata di ogni azienda.

Infatti il software Oracle 10g ci ha permesso di introdurre il concetto di Enterprise Grid Computing:.....

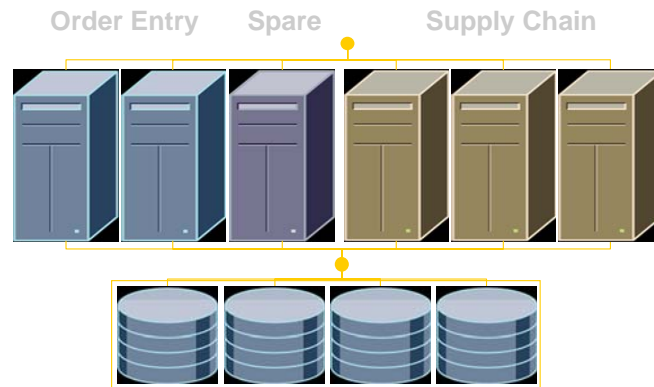
Questo concetto si inserisce perfettamente all'interno della concezione di Adaptive Enterprise di HP: infatti permette di introdurre i vantaggi del Grid Computing, che abbiamo visto prima, all'interno di ogni infrastruttura software aziendale



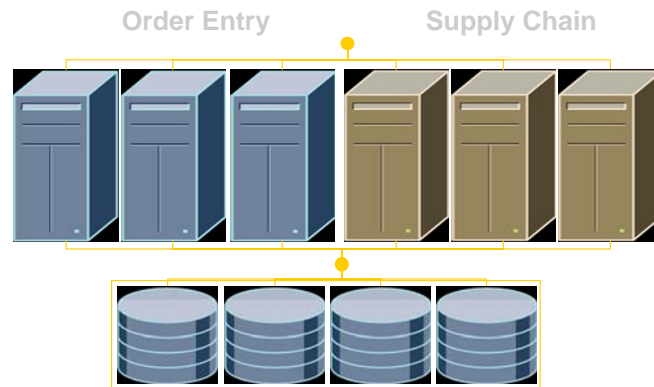
Coordinated use of many servers acting as one large computer.



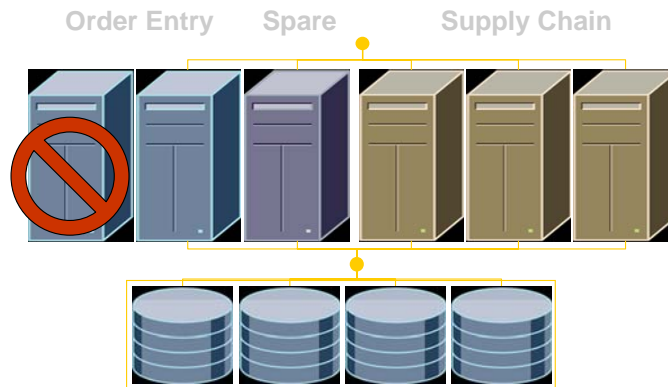
- **Pooling**
- **Virtualization & Provisioning**
- **Load Balancing**
- **Quality of Service**
- **Automation**



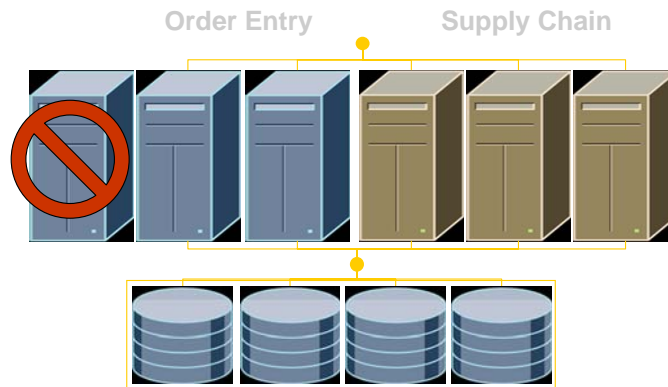
Normal application distribution



Temporary workload increase



Server failure



Recover of the Order Entry on the spare server



- Ongoing project led by Andrea Arcangeli
- The goal is to share the computational power of the machines connected to the Internet in order to create a World Wide Supercomputer
- Also economic aspects are considered: a “market for the CPU resources” chooses the price of the CPU resources using the supply and demand law in real time
- Computational resources should be available to everybody
- Those who share resources are rewarded with cash
- Using the CPUCoins (the CPUCoins are a virtual credit, like in a video game), CPUShare can be optionally used as an energy accumulator, without requiring cash transactions
- The CPUShare protocol is open and in turn it provides interoperability to all OS. Currently only x86, x86\_64 and powerpc64 CPU resources can be sold with CPUShare, but all architectures can be allowed to join in the future
- No hype disclaimer: this project is experimental and young and while I'm optimistic it's only a matter of time before it will work fine on a large scale, it's a brand new technology and it's not guaranteed it will





## References

89

Message Passing Interface: [www-unix.mcs.anl.gov/mpi/](http://www-unix.mcs.anl.gov/mpi/)

Globus Project: [www.globus.org](http://www.globus.org)

Global Grid Forum: [www.gridforum.org](http://www.gridforum.org)

BOINC: [boinc.berkeley.edu](http://boinc.berkeley.edu)

Oracle Technology Network: [otn.oracle.com](http://otn.oracle.com)

CPUShare: [www.cpushare.com](http://www.cpushare.com)