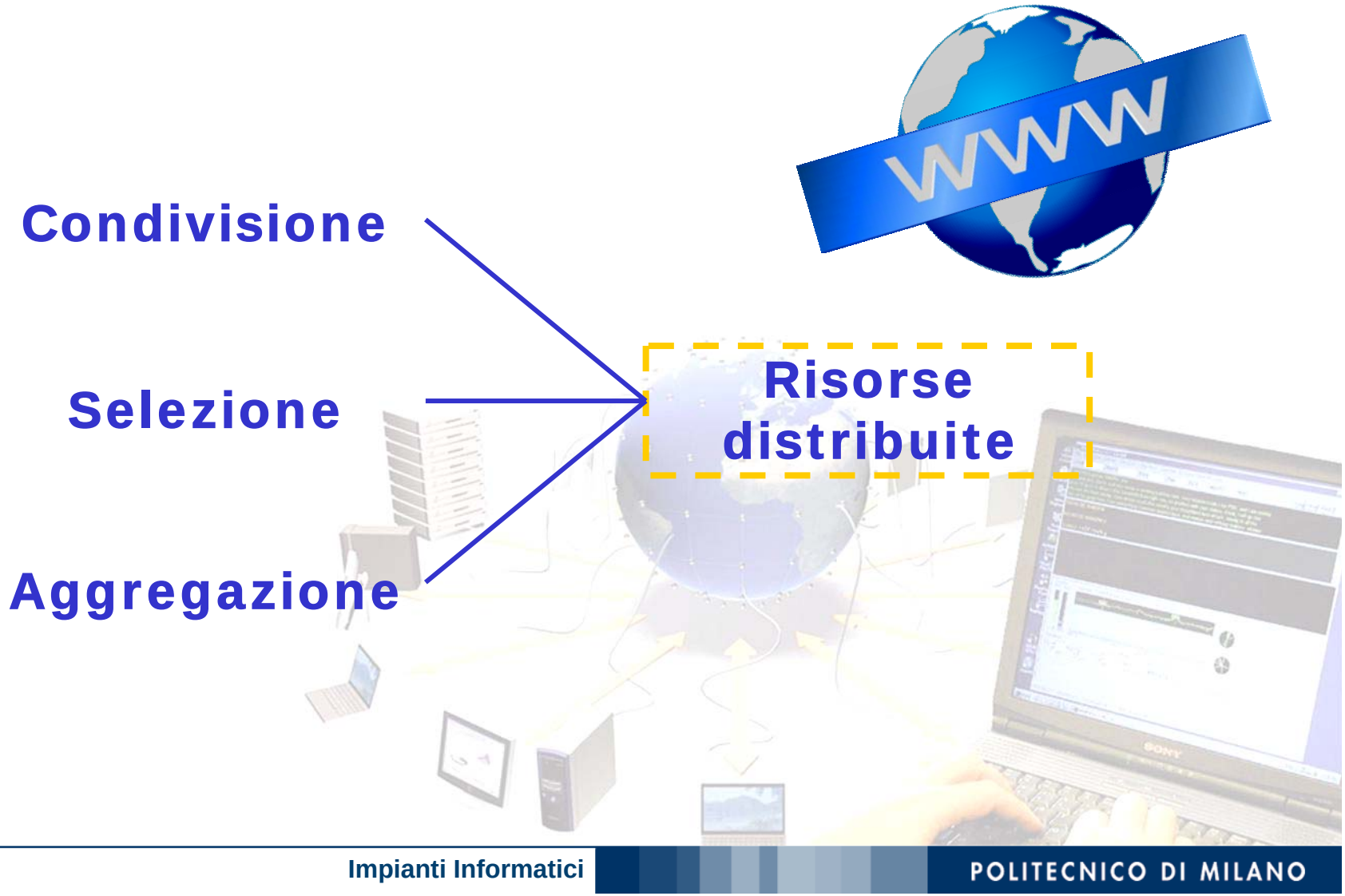
POLITECNICO DI MILANO

Impianti Informatici

Grid computing e architetture parallele

Cos’è il Grid Computing: breve anteprima



Cos’è il Grid Computing: breve anteprima

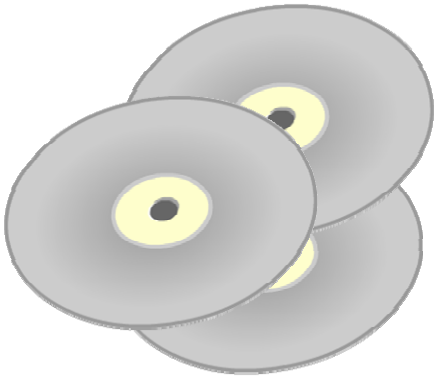
Computer

Software

Dati

Periferiche

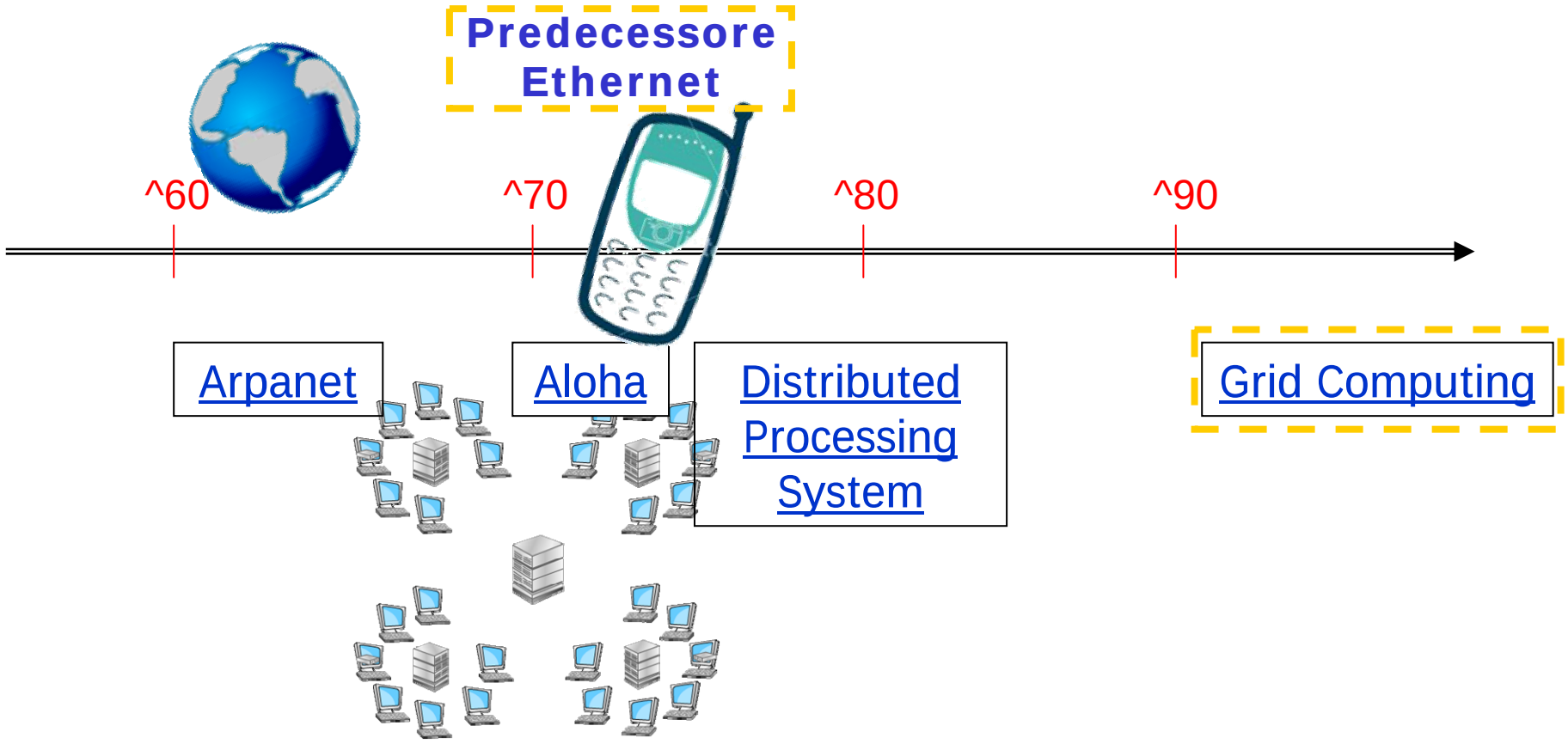
Persone



Risolvere problemi di grande scala

Risorse distribuite

Rivoluzione o evoluzione?



Necessità di più potenza computazionale..

How to Run Applications Faster? There are 3 ways to improve performance:

- Work Harder
- Work Smarter
- Get Help

Computer Analogy

- Using faster hardware
  - Improve the operating speed of processors & other components
    - constrained by the speed of light, thermodynamic laws, & the high financial costs for processor fabrication
- Optimized algorithms and techniques used to solve computational tasks
- Multiple computers to solve a particular task
  - Connect multiple processors together & coordinate their computational efforts
    - parallel computers
    - allow the sharing of a computational task among multiple processors

Sistemi di calcolo distribuito

Availibility e Reliability

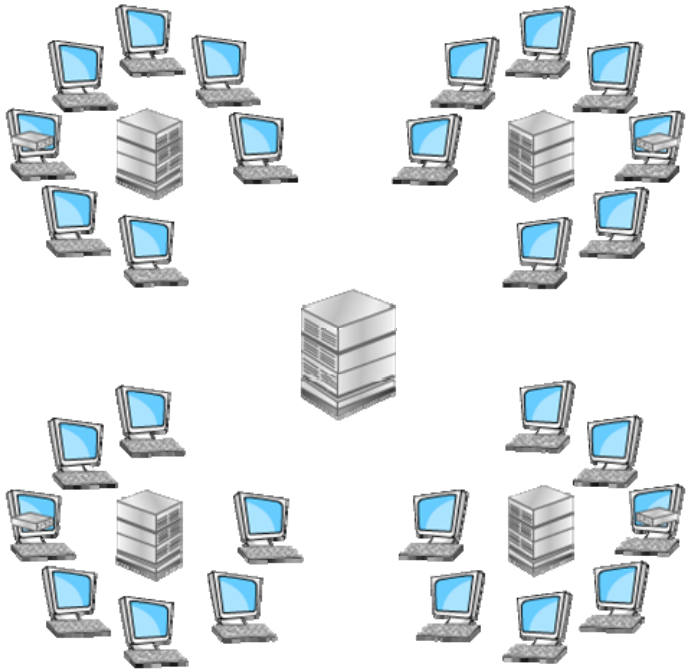
Prestazioni

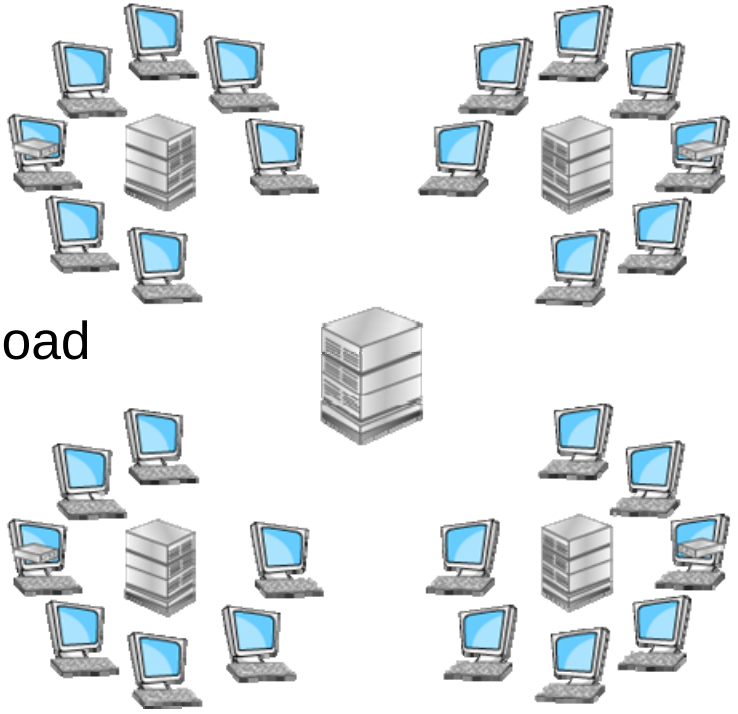
Modularità e facilità di espansione

Condivisione delle risorse

Automatica ripartizione del carico

Buone prestazioni anche in caso di overload





Sistemi di calcolo distribuito: peculiarità

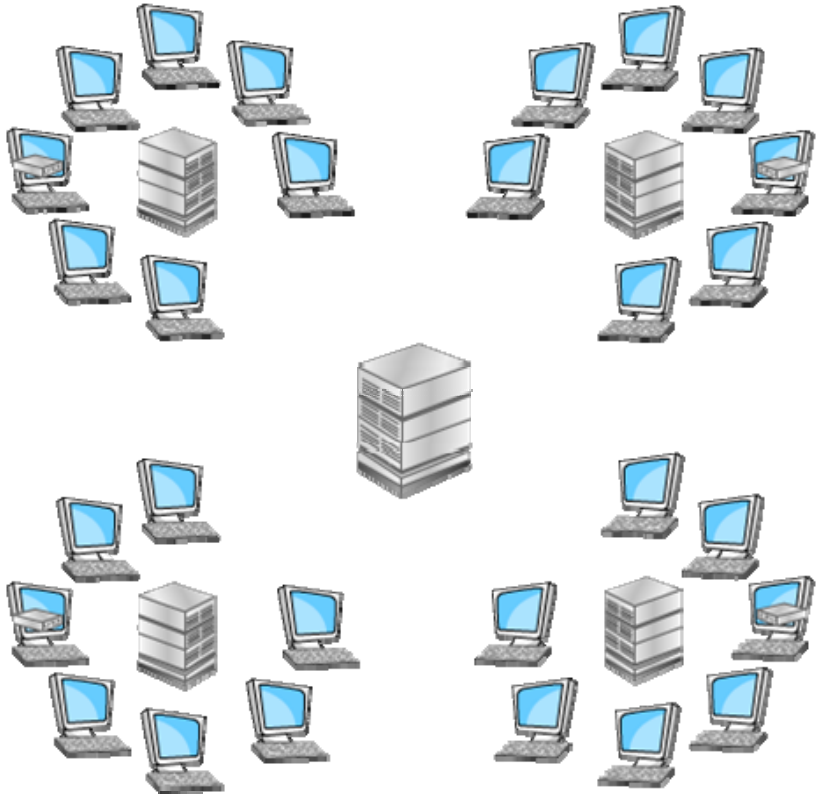
Molteplicità di risorse (repliche per affidabilità/prestazioni)

Interconnessioni (disaccoppiamento tra componenti, es: web services)

Unità di controllo

Trasparenza

Autonomia dei componenti



Sistemi di calcolo distribuito: problematiche

Eterogeneità

Addressing degli oggetti

Nomenclatura

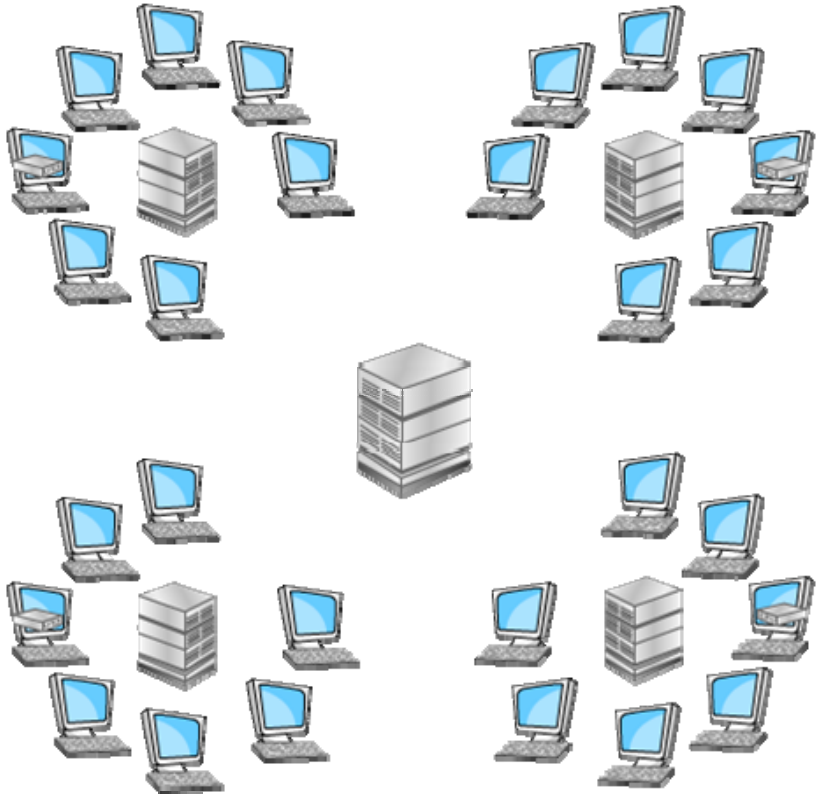
Ownership

Accesso concorrente

Coerenza delle cache

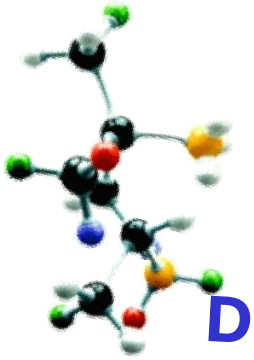
Parallelizzazione del problema

(scomposizione del dominio)

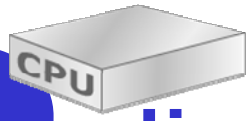


Esigenze

Simulazioni



Dinamica molecolare



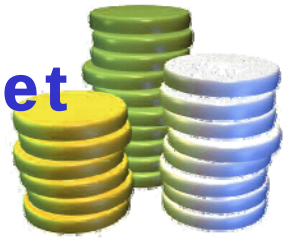
Potenza di calcolo

Previsioni del tempo

Filtraggio di filmati

Evita prototipazione

Time-to-market



Parallel Computing

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

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

Legge di Amdahl (1967)

Tempo di esecuzione della frazione seriale

$$T_1 = T_s + T_p$$

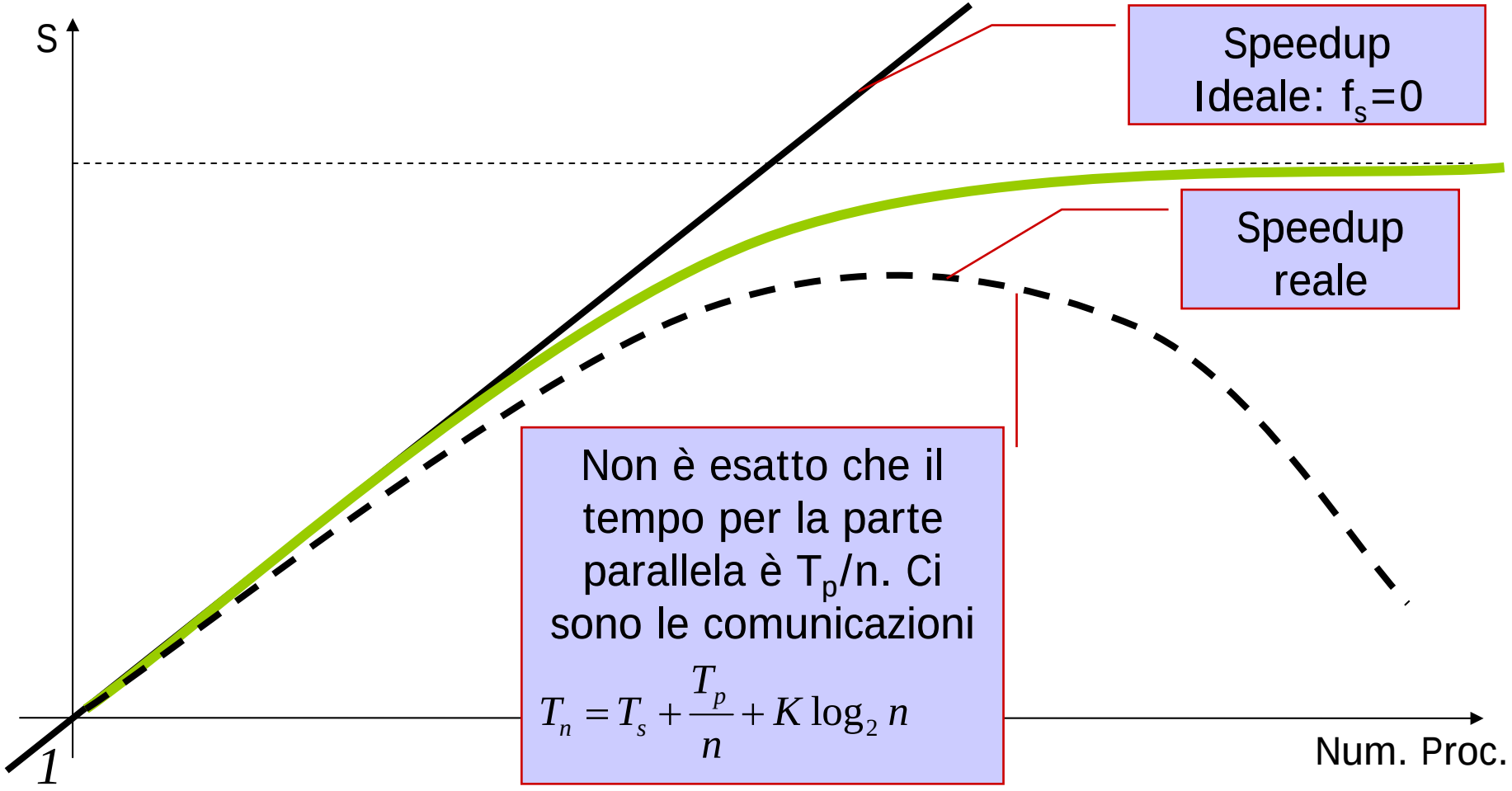
Tempo di esecuzione della frazione parallela

$$T_n = T_s + T_p/n$$

Frazione seriale → $f_s = T_s/(T_s + T_p)$

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

Conseguenze della legge di Amdahl



Legge di Amdahl: esempio

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 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 s

E = 50.25/100 = 50%

Lo speedup *"scalato"*

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:

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

Tipologie di elaborazione parallela (Flynn's Taxonomy)

[SISD]: elaborazione sequenziale

SISD	SIMD
(MISD)	MIMD

SIMD: *Single Instruction, Multiple Data*.

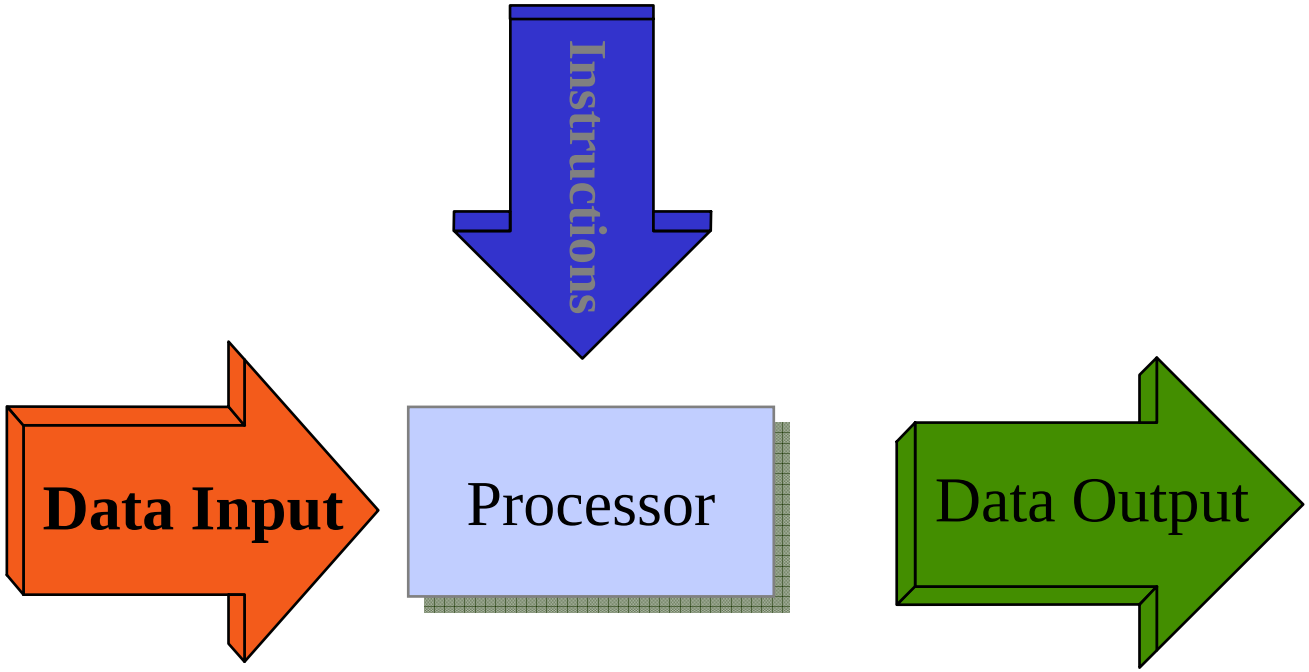
- Calcolatori vettoriali: una sola Control Unit, un solo clock

(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

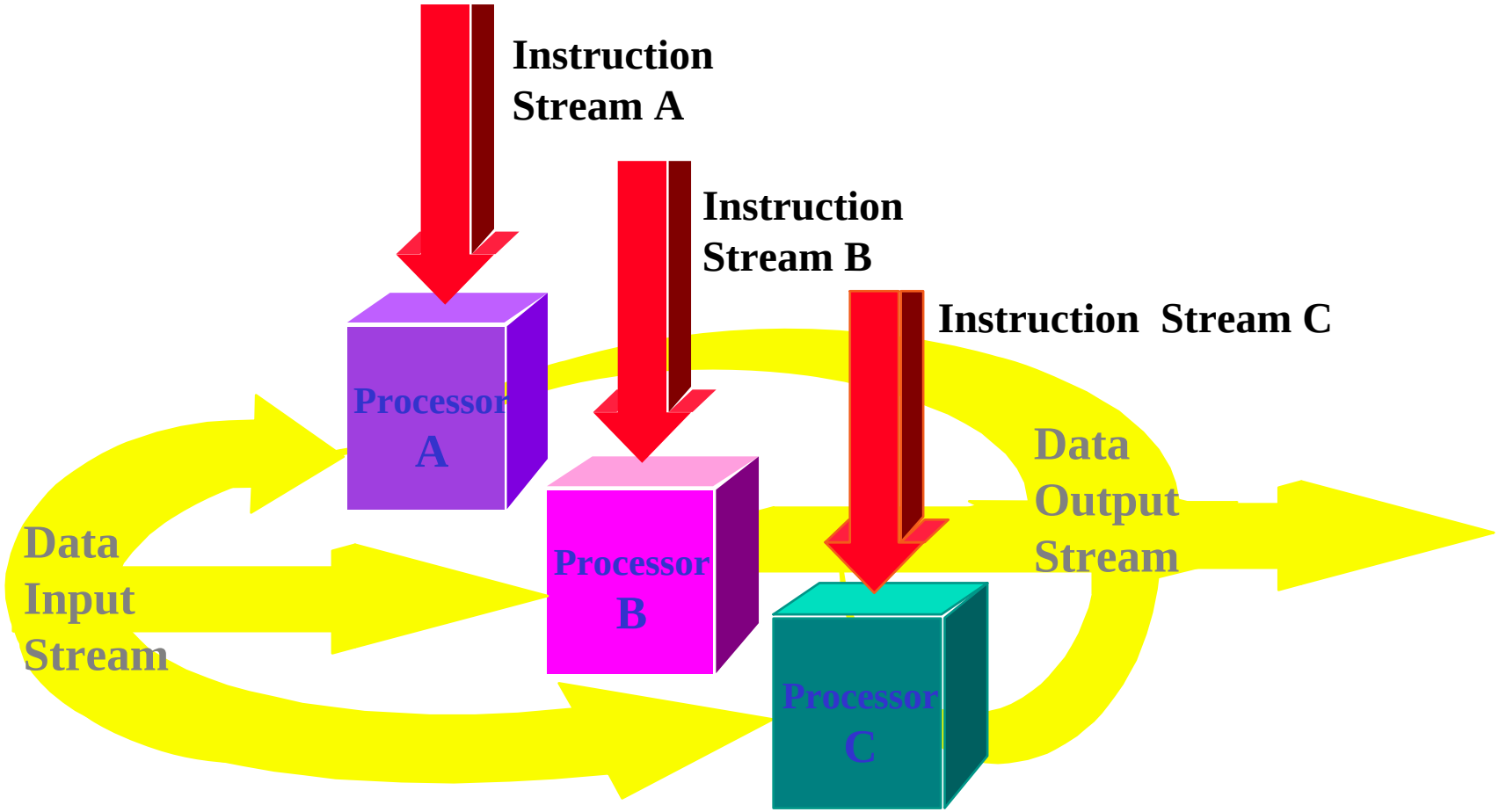
SISD : A Conventional Computer



Speed is limited by the rate at which computer can transfer information internally.

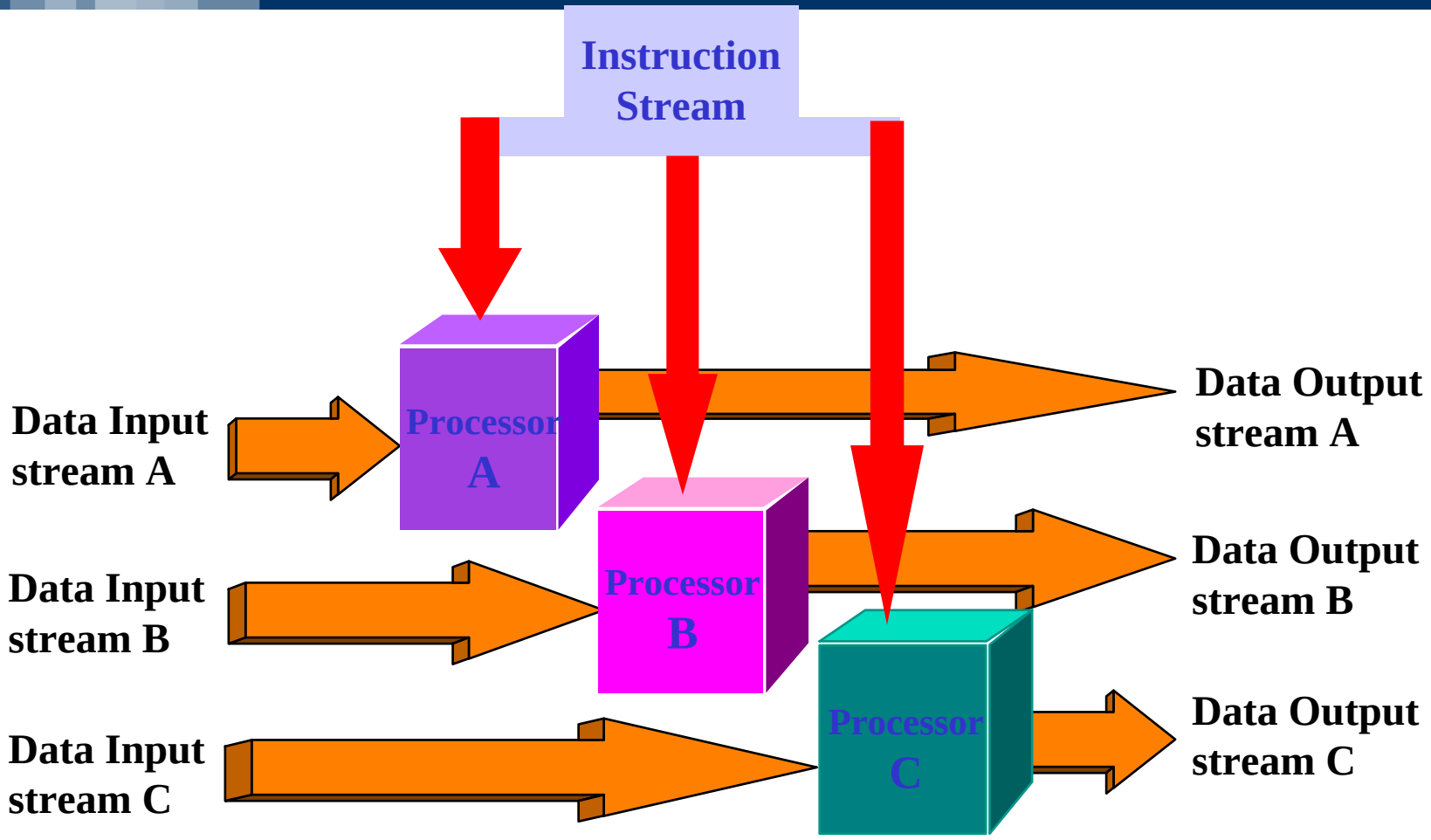
Ex:PC, Macintosh, Workstations

The MISD Architecture



More of an intellectual exercise than a practical configuration. Few built, but commercially not available

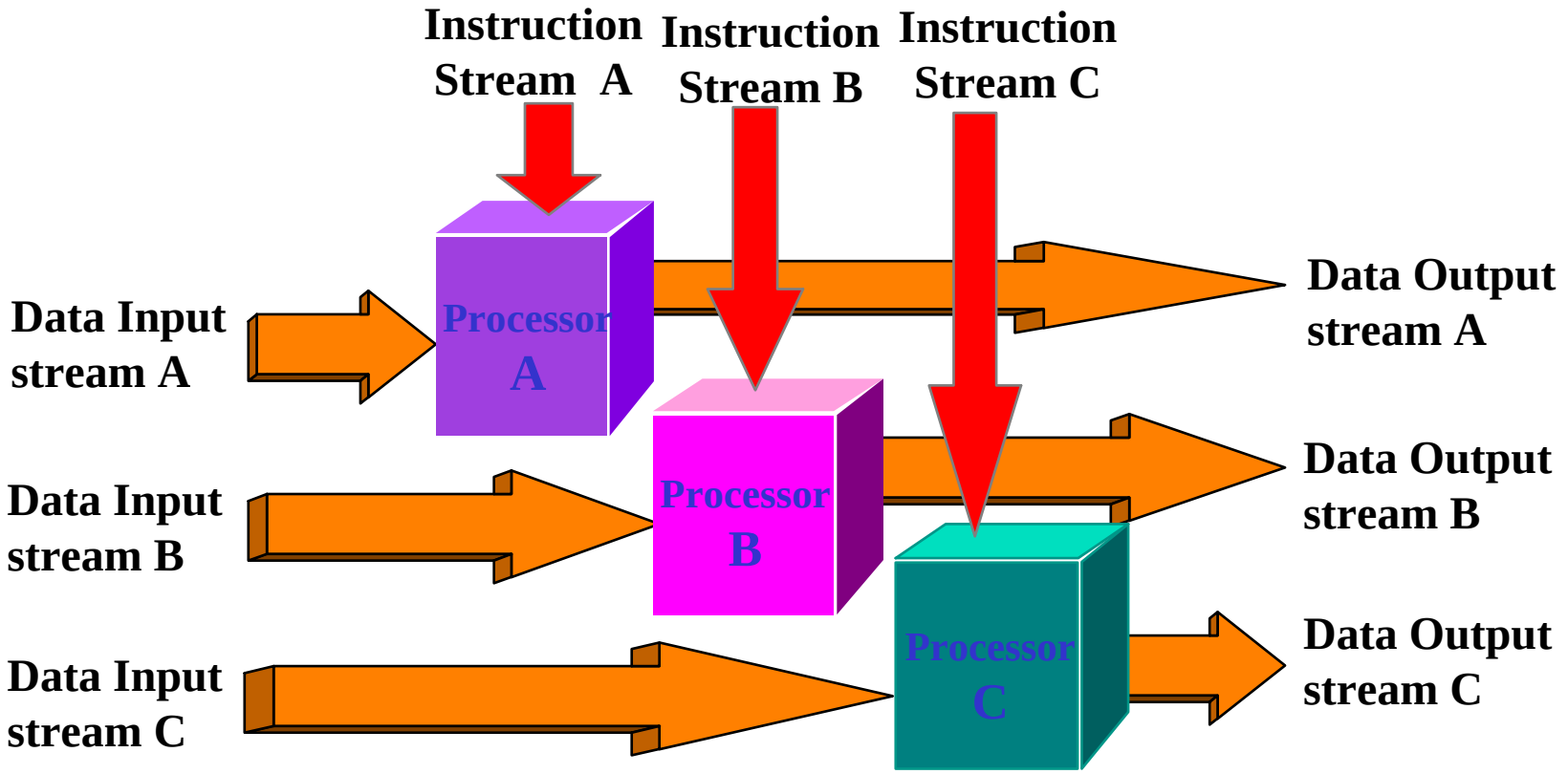
SIMD Architecture



Ex: CRAY machine vector processing, Thinking machine cm*

Intel MMX (multimedia support)

MIMD Architecture



Unlike SISD, MISD, MIMD computer works asynchronously.

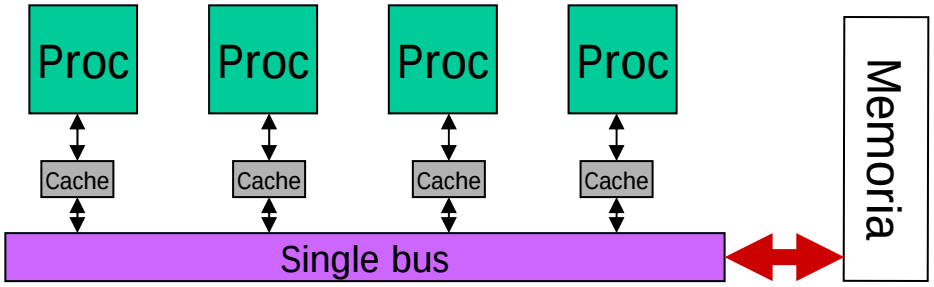
- Shared memory (tightly coupled) MIMD
- Distributed memory (loosely coupled) MIMD

Modelli hardware

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)

Modelli hardware: memoria condivisa



Each processor can name every physical location in the machine

Each process can name all data it shares with other processes

Data transfer via load and store

Data size: byte, word, ... or cache blocks

Uses virtual memory to map virtual to local or remote physical

Pro

- Ease of programming when communication patterns are complex or vary dynamically during execution
- Lower communication overhead, good utilization of communication bandwidth for small data items, no expensive I/O operations
- Hardware-controlled caching to reduce remote commutations when remote data is cached

Contro

- BUS is the bottleneck:
- limited number of processors (not more than 16)
- Required a cache to limit the bus accesses by processors

→ Cache coherency issue

memoria condivisa: UMA vs NUMA

Uniform Memory Access (UMA):

- Elaboratori costituiti da più processori identici e sovente chiamati Symmetric Multiprocessor (SMP)
- I tempi d'accesso a tutta la memoria sono uguali per ogni processore
- Tipicamente sono CC-UMA (Cache Coherent UMA)
  - *Cache Coherent* significa che, se un processore aggiorna una locazione di memoria, il sistema provvede automaticamente ad invalidare eventuali locazioni di cache degli altri processori mappate sulla locazione di memoria appena modificata
  - La *Cache Coherency* è mantenuta a livello hardware

•***Non-Uniform Memory Access (NUMA)***:

- Generalmente sono realizzate attraverso il collegamento di due o più SMP
- Ogni SMP può accedere alla memoria degli altri SMP
- I processori non hanno lo stesso tempo d'accesso a tutta la memoria (l'accesso alla memoria tramite link è più lento)
- Se è mantenuta la *cache coherency* si parla di architetture CC-NUMA (*Cache Coherent* NUMA)

Multiprocessors Cache Coherency

•“ogni *read* deve ritornare la più recente *write*” → difficile da implementare

•“ogni *write* deve essere vista da una *read*” → le *write* sono viste in ordine (serialization)

Due regole per assicurarlo:

• “Se P scrive x and P1 lo legge, la scrittura di P deve essere vista da P1 se le due operazioni sono sufficientemente separate”

• Scritture ad un singolo indirizzo sono serializzate: in ordine di arrivo

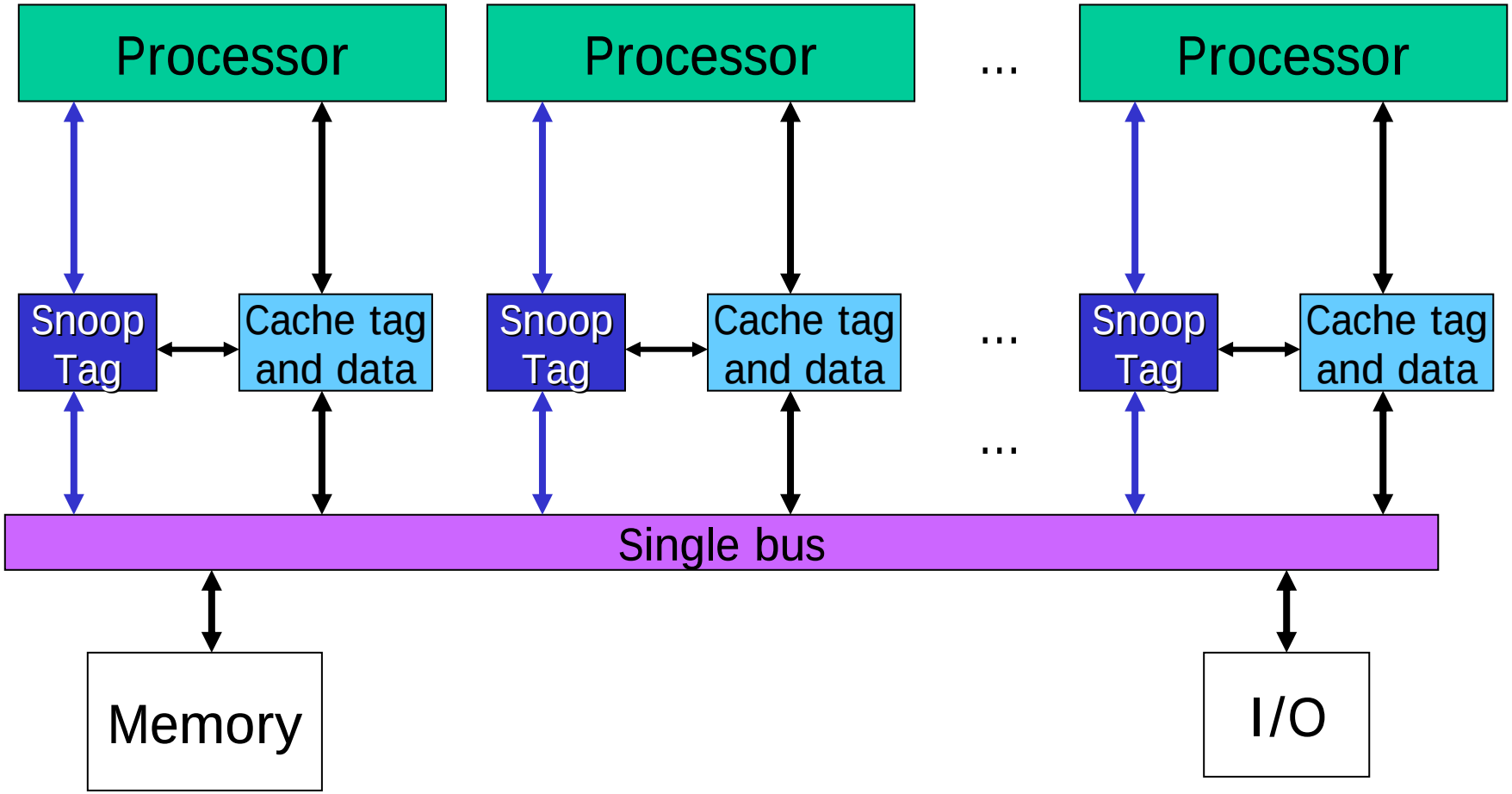
  • Viene vista l’ultima *write*

  • Altrimenti verrebbe visto un valore più vecchio dopo il più nuovo

---

- 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)
- Letture multiple dello stesso dato non creano problemi
  - è importante, invece, garantire le coerenza in fase di scrittura

Multiprocessors Cache Coherency



Alcune soluzioni HW per la coerenza

- Snooping Solution (Snoopy Bus):
  - Send all requests for data to all processors
  - Processors snoop to see if they have a copy and respond accordingly
  - Requires broadcast, since caching information is at processors
  - Works well with bus (natural broadcast medium)
  - Dominates for small scale machines (most of the market)

- Directory-Based Schemes
  - Keep track of what is being shared in 1 centralized place (logically)
  - distributed directory for scalability (distributed memory avoids bottlenecks)
  - Send point-to-point requests to processors via network
  - Scales better than Snooping
  - Actually existed BEFORE Snooping-based schemes

Basic Snoopy Protocols

Write Invalidate Protocol:

- Multiple readers, single writer
- Write di un dato shared: viene inviato un segnale di *invalidate* a tutte le cache
- Read Miss:
  - Write-back: snoop in caches to find most recent copy

    → 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 Broadcast Protocol (typically with write through):

- Write to shared data: broadcast on bus, processors snoop, and *update* any copies
- Read miss: memory is always up-to-date
  - Write-through: la memoria è sempre up-to-date

    → Update: invece di invalidare le cache, si aggiornano direttamente mandando il nuovo dato su bus
    - chi possiede quello vecchio, lo aggiorna.
    - E' molto band consuming

Write serialization: bus serializes requests!

- Bus is single point of arbitration

Write Invalidate versus Broadcast:

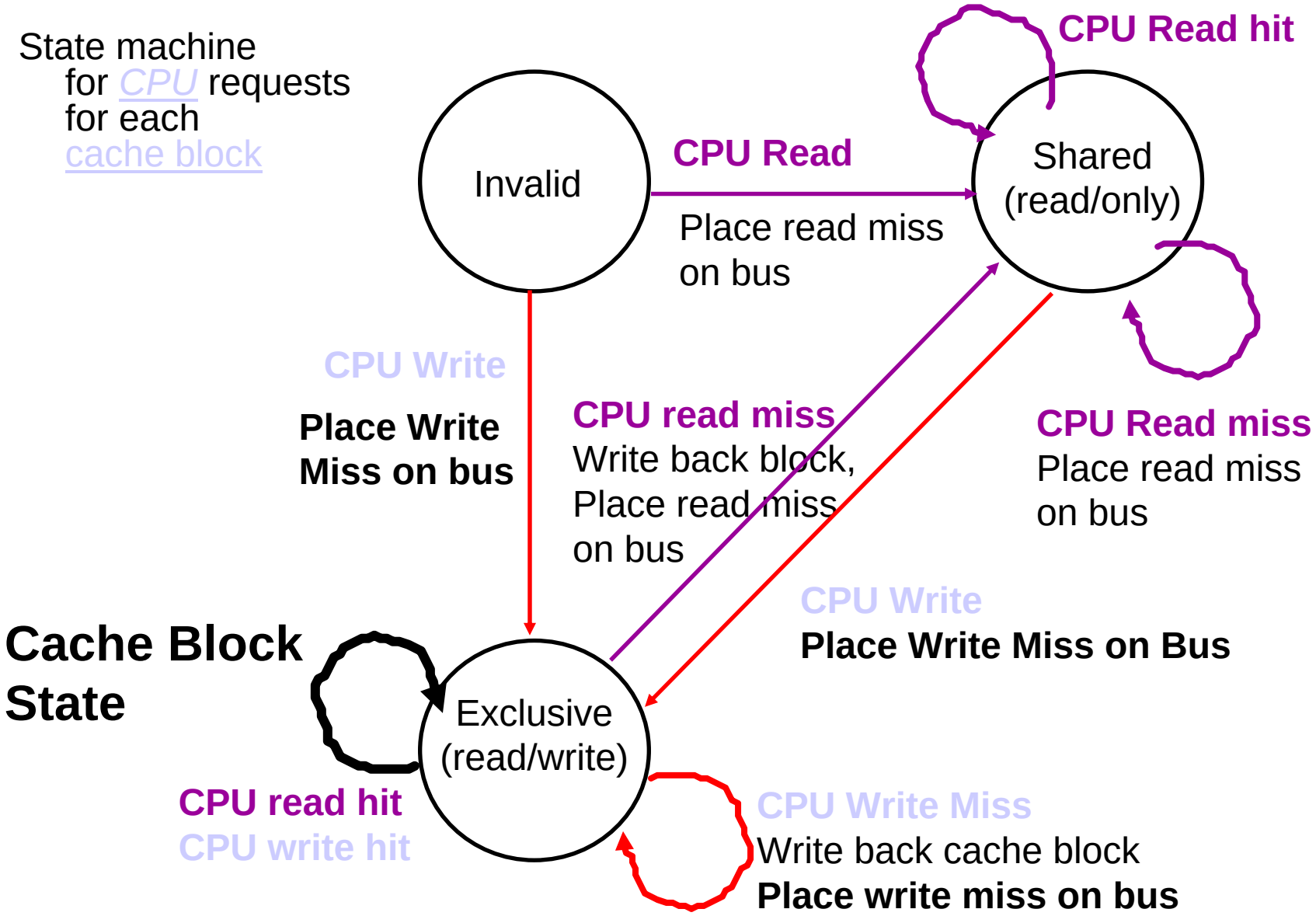
- Invalidate requires one transaction per write-run
- Invalidate uses spatial locality: one transaction per block
- Broadcast has lower latency between write and read

An Example Snoopy Protocol

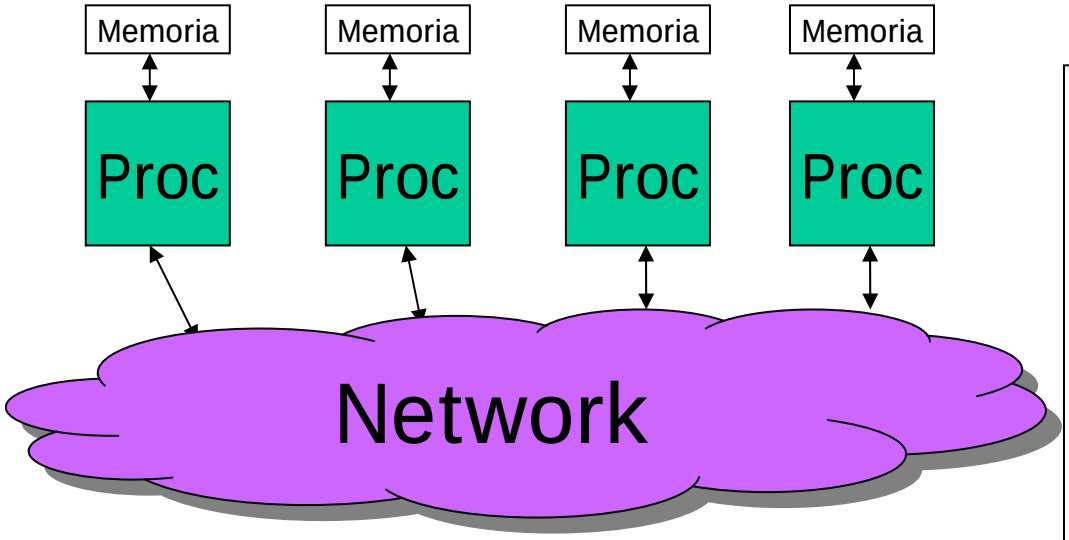
Invalidation protocol, write-back cache

- Each block of memory is in one state:
  - Clean in all caches and up-to-date in memory (Shared)
  - OR Dirty in exactly one cache (Exclusive)
  - OR Not in any caches
- Each cache block is in one state (track these):
  - Shared : block can be read
  - OR Exclusive : cache has only copy, it's writeable, and dirty
  - OR Invalid : block contains no data
- Read misses: cause all caches to snoop bus
- Writes to clean line are treated as misses

Snoopy-Cache State Machine-I



Modelli hardware: memoria distribuita



Pro

- Ogni processore ha il suo spazio di indirizzamento privato
- La quantità di memoria è scalabile con il numero di processori, ovvero al crescere del numero dei processori la memoria disponibile cresce proporzionalmente
- Ogni processore accede ”rapidamente” alla propria memoria, senza interferenze e senza l’*overhead* dovuto al mantenimento della *cache coherency*.
- Costi contenuti: può essere realizzata tramite processori ed infrastruttura di comunicazione disponibili sul mercato di massa (*commodity off-the-shelf*)

Contro

- Il programmatore è responsabile di tutti i dettagli legati alla comunicazione di dati tra cpu
- Potrebbe essere complesso “mappare” il layout dei dati di codici esistenti (pensati per un unico spazio di indirizzamento) su un modello di memoria distribuita
- Esiste una gerarchia di tempi d’accesso alla memoria, più complessa di NUMA

Esempi di programmazione

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

Soluzione sequenziale


```
#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];
...
}
```

Soluzione parallela


```
#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 parallela

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

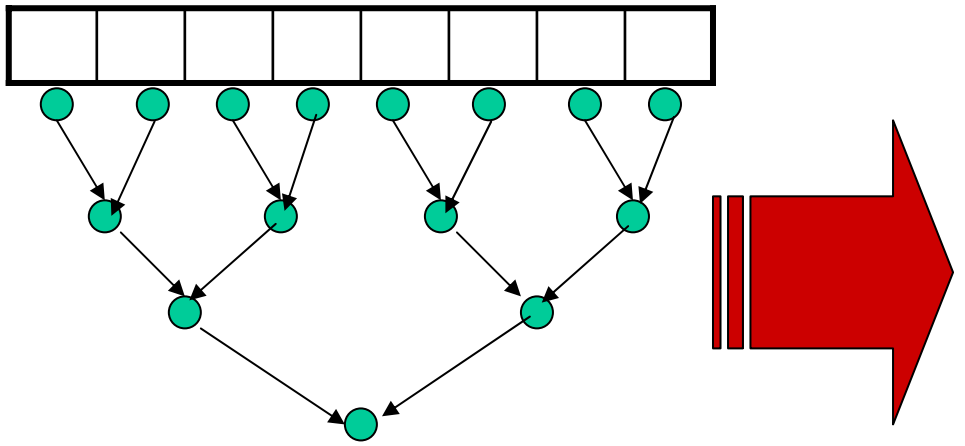
```
synch();

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

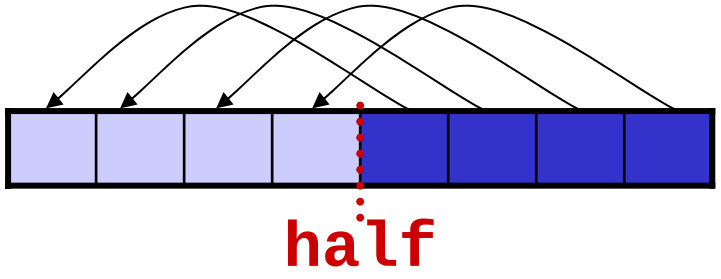
Per evitare che cominci a sommare i parziali prima che gli altri abbiano finito di calcolarli

Soluzione parallela

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



$log_2\ N$ somme invece di N



Applico questo passaggio iterativamente

Soluzione parallela


```
   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

Paradigmi a memoria condivisa

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:2,j=1:n:3) 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

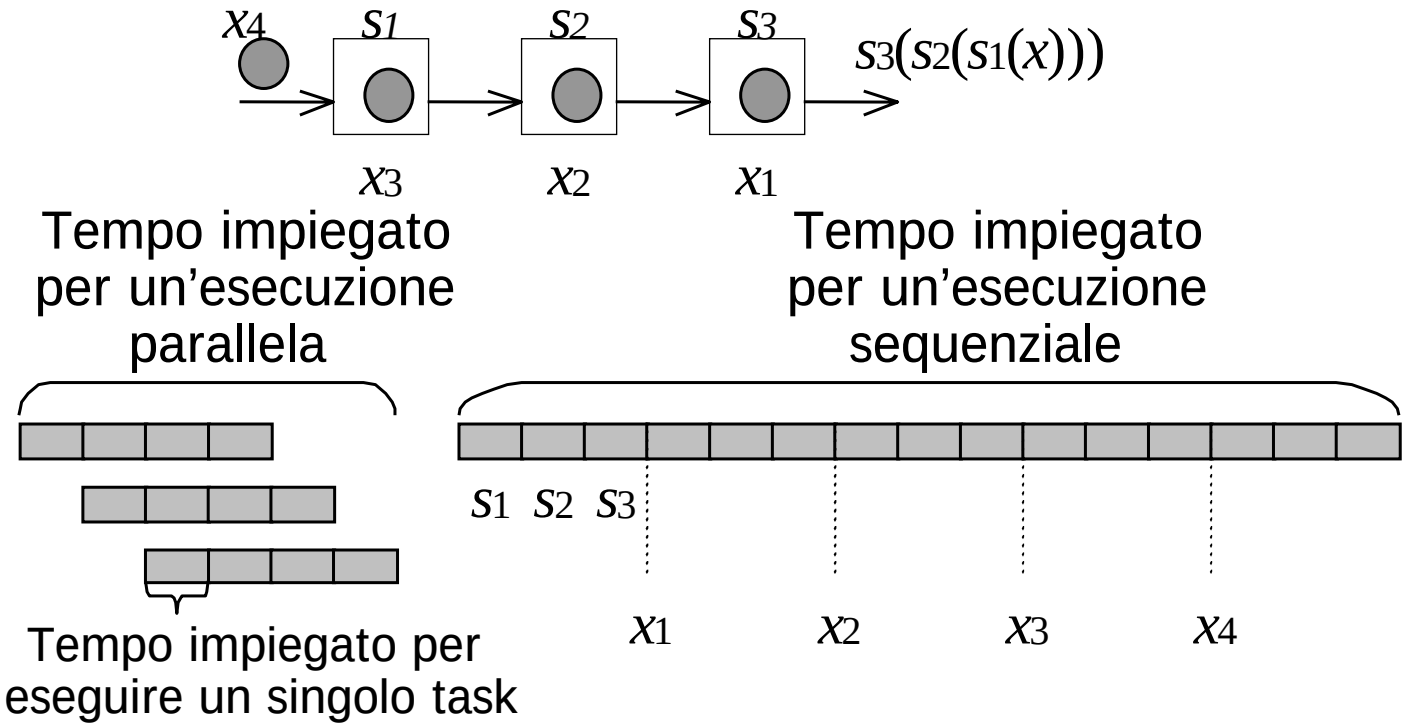
Paradigmi a Skeleton

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
```

Paradigmi a Memoria Distribuita

PVM (Parallel Virtual Machine, 1992) – 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, 1994): approfondito nel seguito…

MPI: Message Passing Interface

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

MPI

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: i *"device"*

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)

MPI: caratteristiche

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

Quando usare MPI…

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)

Esempio MPI: calcolo di pi-greco in parallelo

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

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.

Esempio: pi-greco

```
#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);
```

Esempio: pi-greco (2)

```
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)

```
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
```

POLITECNICO DI MILANO

Impianti Informatici

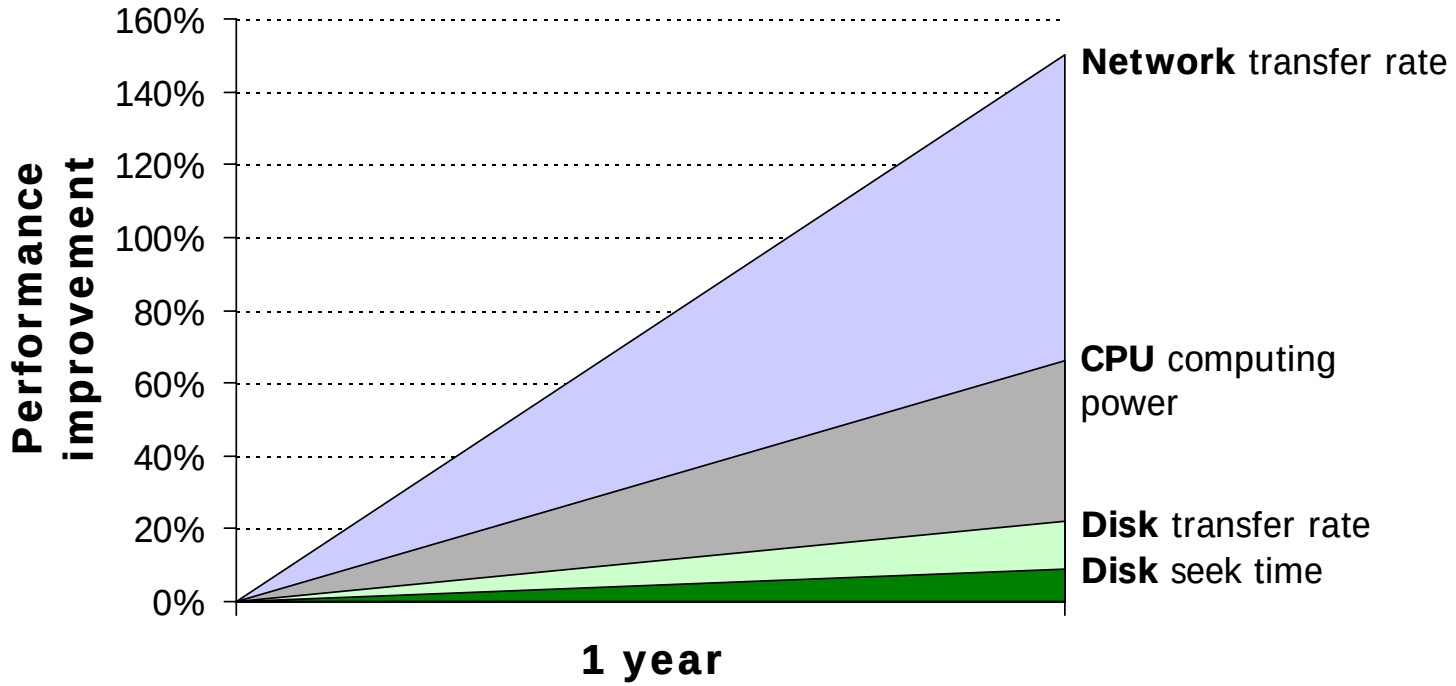
Grid computing: infrastrutture

Clusters

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



Cos'è un Cluster?

- A cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers cooperatively working together as a single, integrated computing resource.
  - "stand-alone" (whole computer) computer that can be used on its own (full hardware and OS).

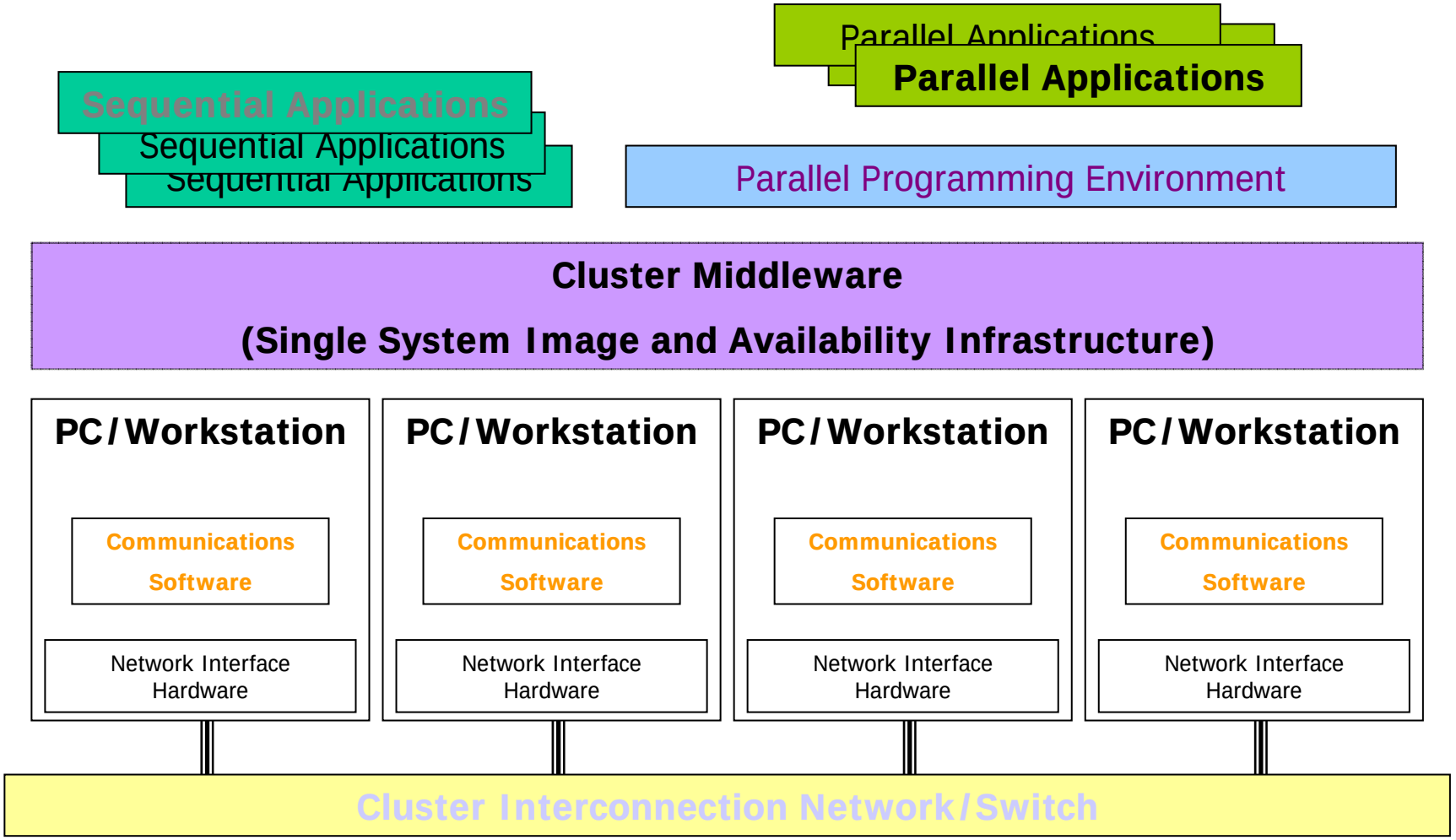
SSI (Single System Image)

- A single system image is the illusion, created by software or hardware, that presents a collection of resources as one, more powerful resource.
- SSI makes the cluster appear like a single machine to the user, to applications, and to the network.
- A cluster without a SSI is not a cluster

SSI Benefits

- Provide a simple, straightforward view of all system resources and activities, from any node of the cluster
- Free the end user from having to know where an application will run
- Free the operator from having to know where a resource is located
- Let the user work with familiar interface and commands and allows the administrators to manage the entire clusters as a single entity
- Reduce the risk of operator errors, with the result that end users see improved reliability and higher availability of the system
- Allowing centralize/decentralize system management and control to avoid the need of skilled administrators from system administration
- Present multiple, cooperating components of an application to the administrator as a single application
- Greatly simplify system management
- Provide location-independent message communication
- Help track the locations of all resource so that there is no longer any need for system operators to be concerned with their physical location
- Provide transparent process migration and load balancing across nodes.
- Improved system response time and performance

Cluster Computer Architecture



Cluster Applications

- Numerous Scientific & engineering Apps.
- Business Applications:
  - E-commerce Applications (Amazon, eBay ….);
  - Database Applications (Oracle on clusters).
- Internet Applications:
  - ASPs (Application Service Providers);
  - Computing Portals;
  - E-commerce and E-business.
- Mission Critical Applications:
  - command control systems, banks, nuclear reactor control, star-wars, and handling life threatening situations.

Cluster: componenti promettenti

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

Cluster: punti chiave

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

Clusters: classificazione

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

Cluster: classificazione

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

Cluster: classificazione

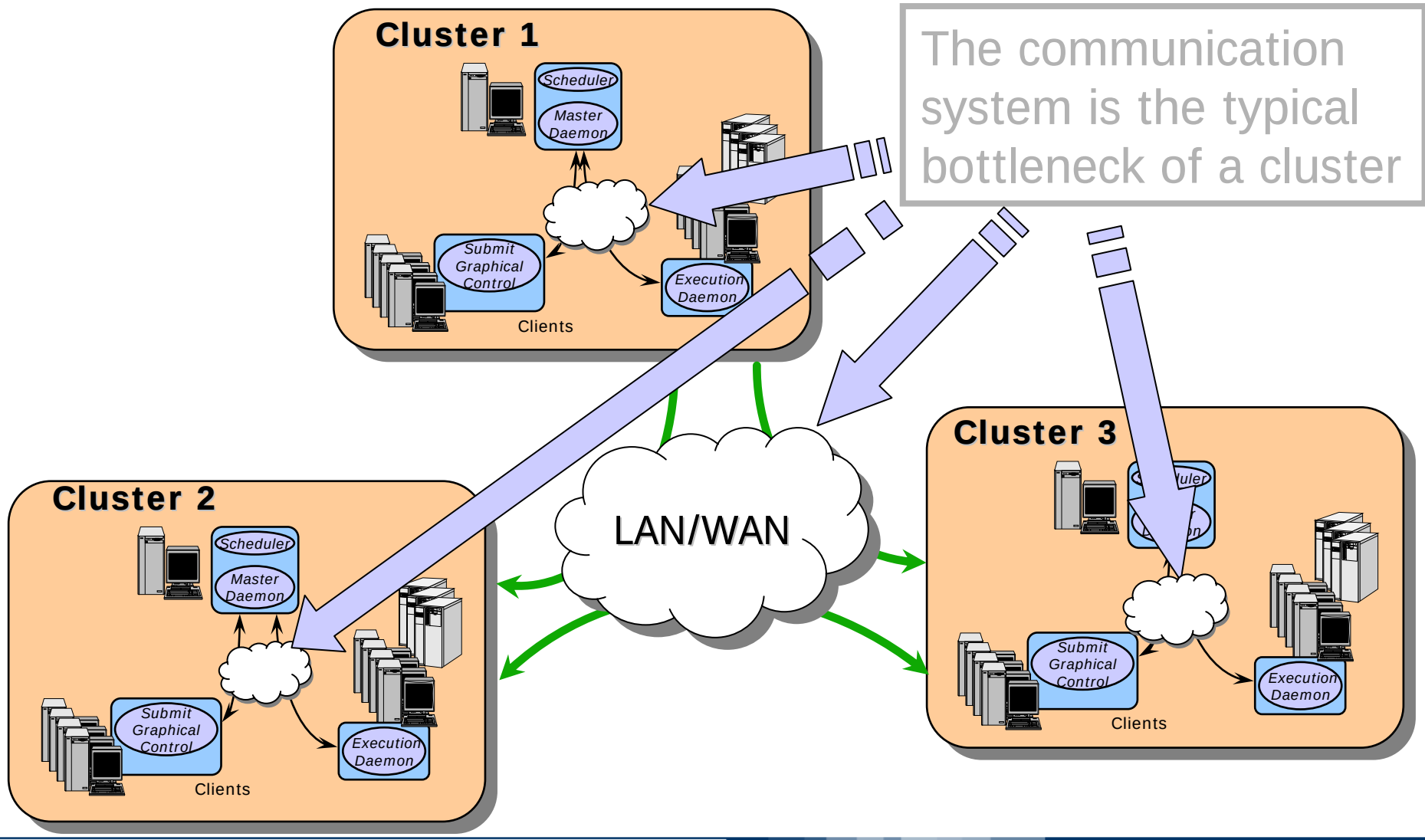
Node Configuration

- 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

Cluster di Cluster: Hyperclusters



Network Exponentials

Network vs. computer performance

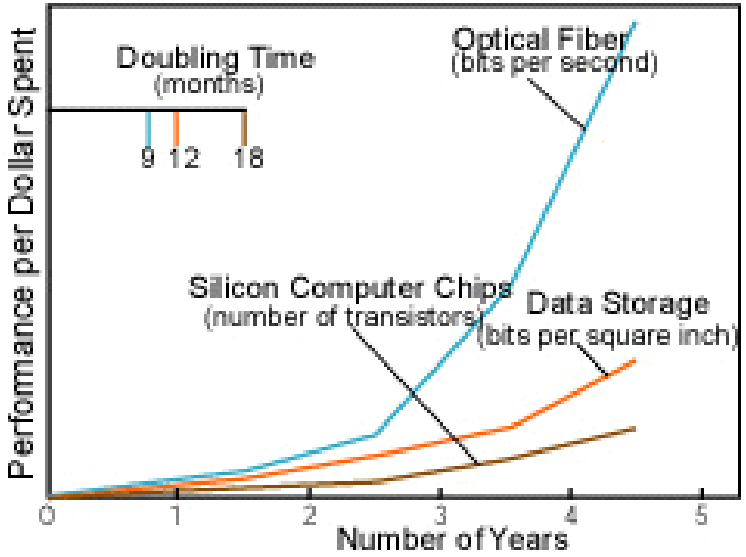
- La velocità dei computer raddoppia ogni 18 mesi
- La velocità della rete raddoppia ogni 9 mesi

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

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).

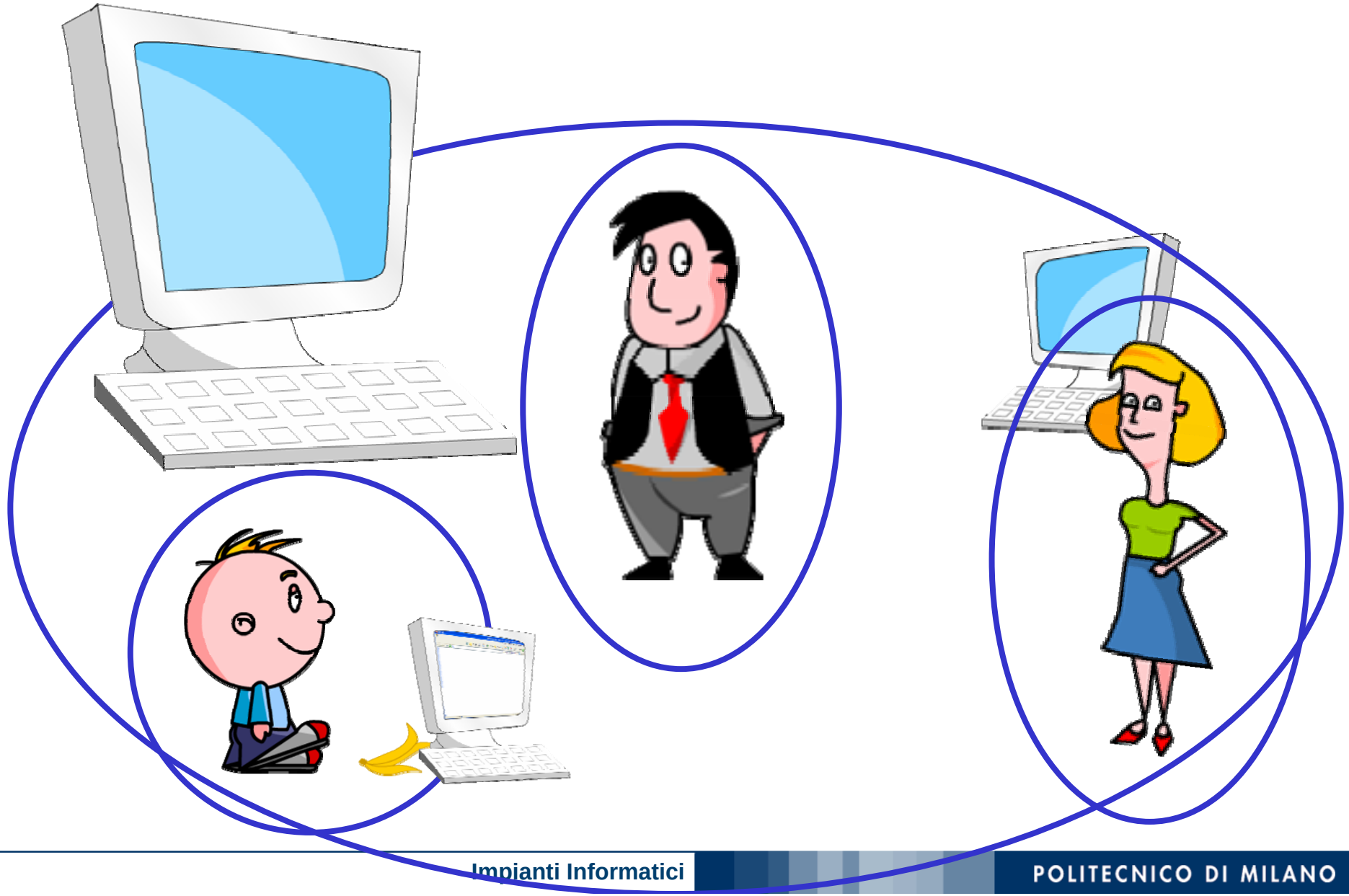
The Grid Problem

- Flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resource

  From “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”

- Enable communities (“virtual organizations”) to share geographically distributed resources as they pursue common goals -- *assuming the absence* of…
  - central location,
  - central control,
  - existing trust relationships
  - …omniscience,

Comunità



HPC vs HTC

High Performance Computing

High Throughput Computing

Potenza istantanea

Throughput

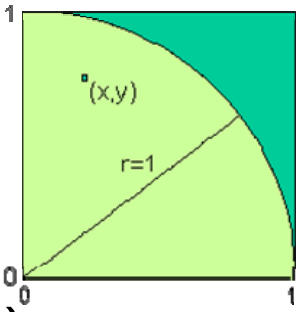
Unità/giorno

Simulazioni/mese

Istruzioni/anno

High Throughput Computing
(parameter sweep applications)

- A study involving exploration of possible scenarios - i.e., execution of the same program for various design alternatives (data).
- It consists of large number of tasks (1000s).
- Generally, no inter-task communication (task farming).
- Large size data (MBytes+) files and I/O constraints
- A large class of application areas:
  - Parameter explorations and simulations (**Monte Carlo**);
  - A large number of science, engineering, and commercial applications: Astrophysics, Drug Design, NeroScience, Network simulation, structural engineering, automobiles crash simulation, aerospace modeling, financial risk analysis
- Condor, Nimrod/G, DesignDrug@Home, SETI@Home, FOLD@Home, Distributed.net.

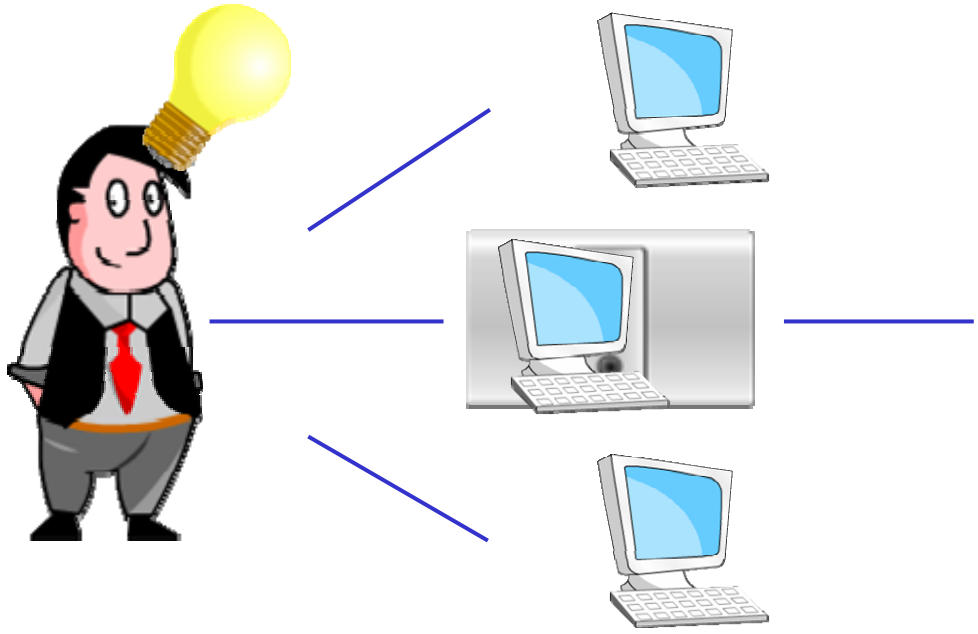


Anni 90: The Grid

Grid Computing

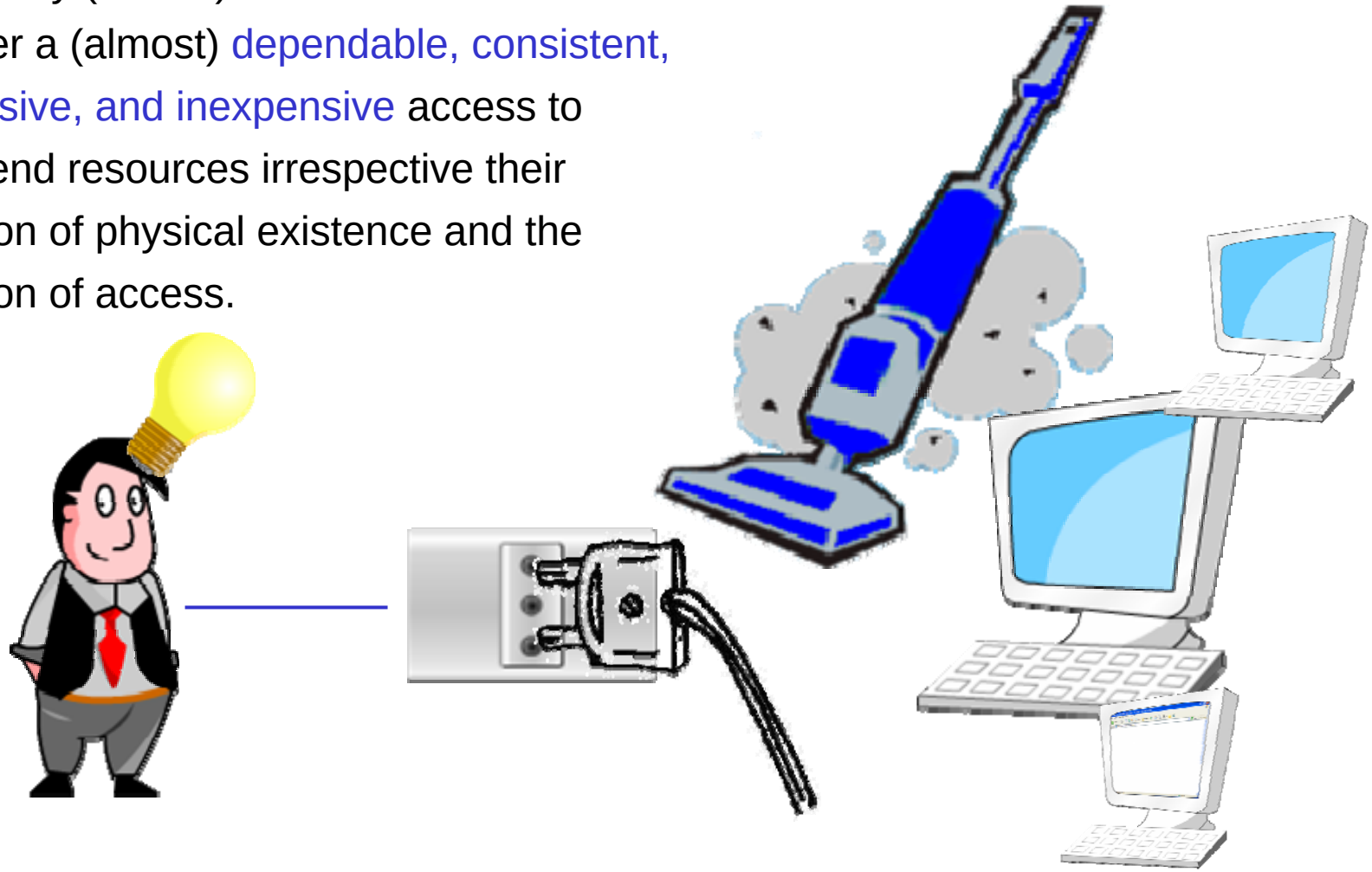
Infrastruttura di computazione distribuita

Supporto a problemi complessi



Grid Computing vs Power Grid

”The Computational Grid” is analogous to Electricity (Power) Grid and the vision is to offer a (almost) dependable, consistent, pervasive, and inexpensive access to high-end resources irrespective their location of physical existence and the location of access.



Alcuni obiettivi

Scalabilità

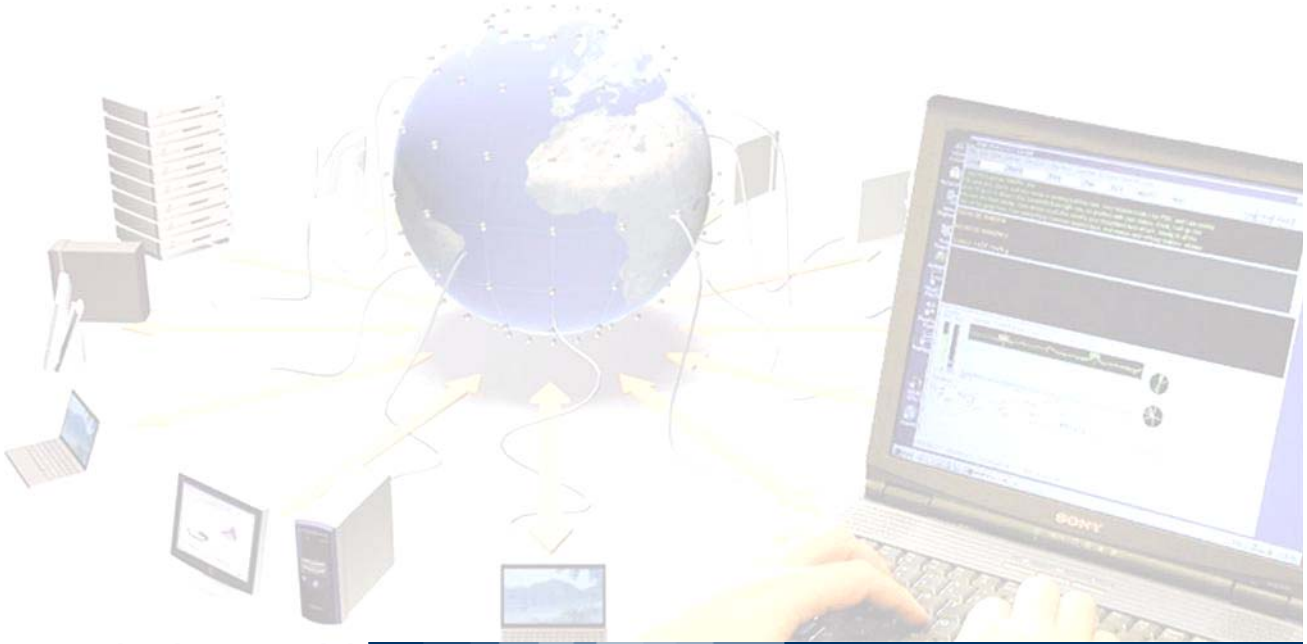
Stabilità

Cost-saving

- Performance
- Simulazioni
- Modelli complicati

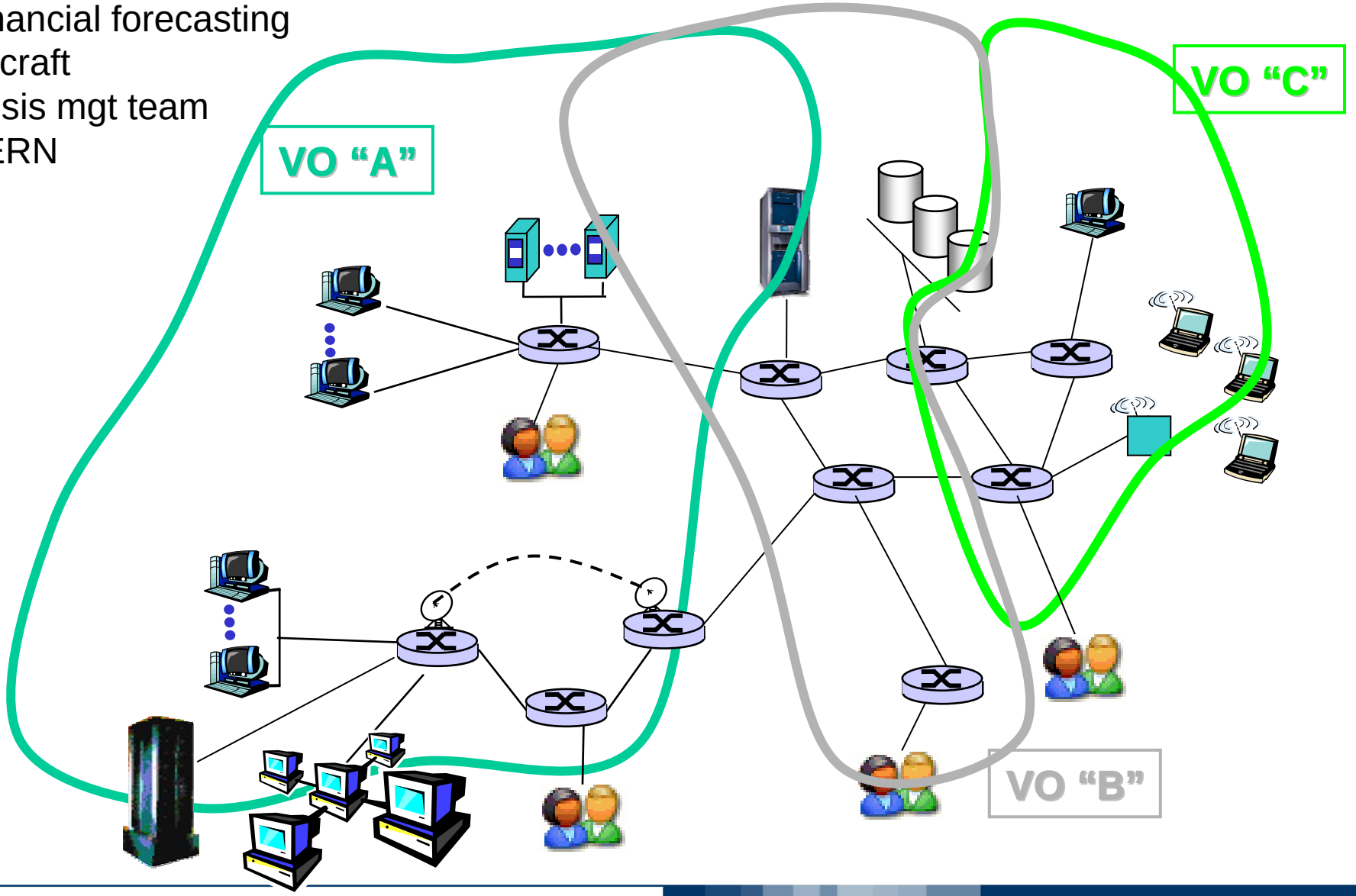
Trasparenza

Accessibilità



Virtual Organization

- Financial forecasting
- Aircraft
- Crisis mgt team
- CERN



Virtual Organization

Collaborazione di individui/istituzioni

Condivisione di risorse

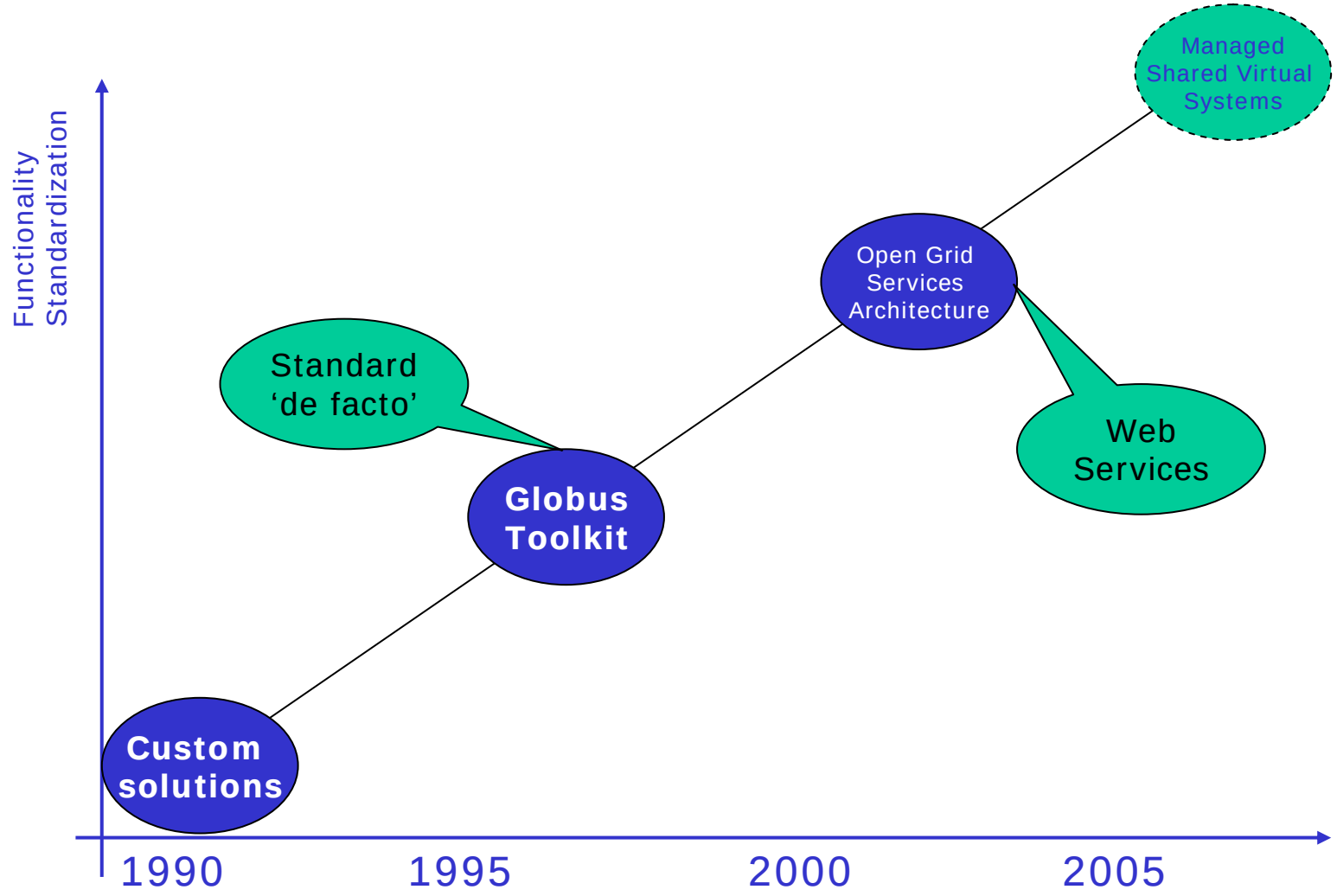
Dinamiche

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

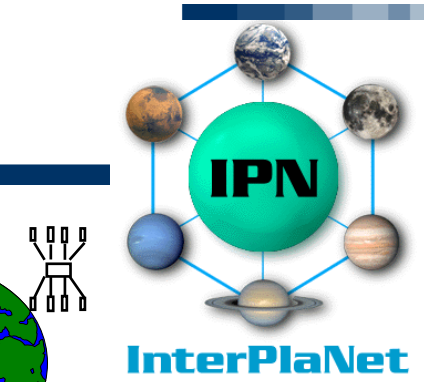
Obiettivo comune da raggiungere

Non devono interferire tra loro

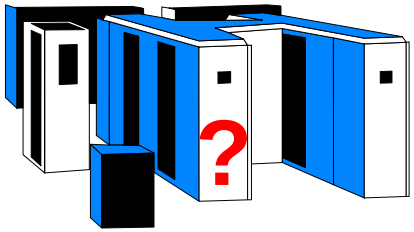
Evoluzione delle tecnologie Grid-enabling

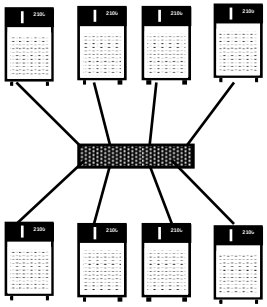


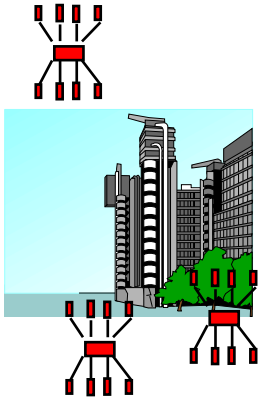
Computing Platforms Evolution: Breaking Administrative Barriers

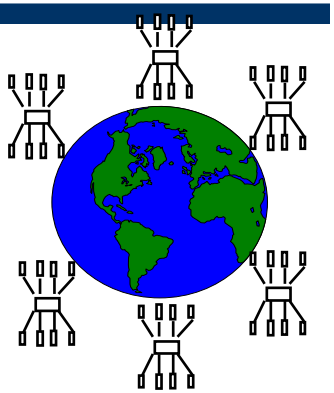


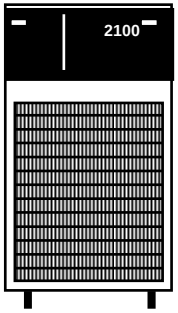
PERFORMANCE

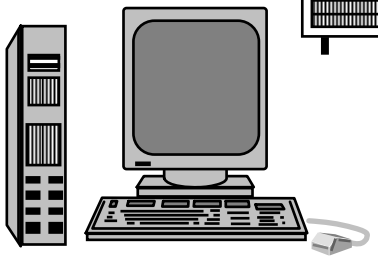


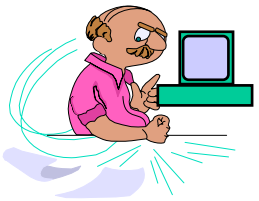












Administrative Barriers

- Individual
- Group
- Department
- Campus
- State
- National
- Globe
- Inter Planet
- Universe

Desktop (Single Processor)

SMPs or SuperComputers

Local Cluster

Enterprise Cluster/Grid

Global Cluster/Grid

Inter Planet Cluster/Grid ??

Costruire e usare la “griglia”: facilities

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

Programming & Systems Problems

The programming problem

- Facilitate development of sophisticated apps
- Facilitate code sharing
- Requires programming environments
  - APIs, SDKs, tools

The systems problem

- Facilitate coordinated use of diverse resources
- Facilitate infrastructure sharing
  - e.g., certificate authorities, information services
- Requires systems
  - protocols, services

The Systems Problem

"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

The Programming Problem

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 Globus Toolkit™

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 authonomy.

from www.globus.org, "About the Globus Toolkit"

Progetto iniziano nel 1998 (ver. 1.0). Nel 2002 e 2003 rilasciate le versioni 2.0 e 3.0, rispettivamente.
Versione corrente (stable): 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

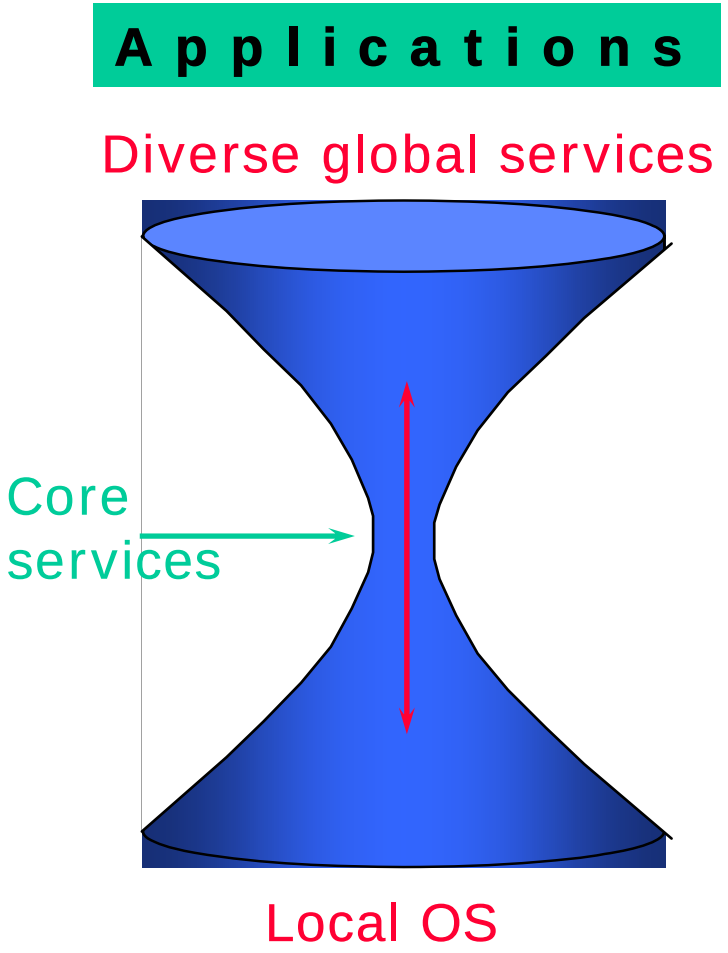
The Hourglass Model

Focus on architecture issues

- Propose set of core services as basic infrastructure
- Use to construct high-level, domain-specific solutions

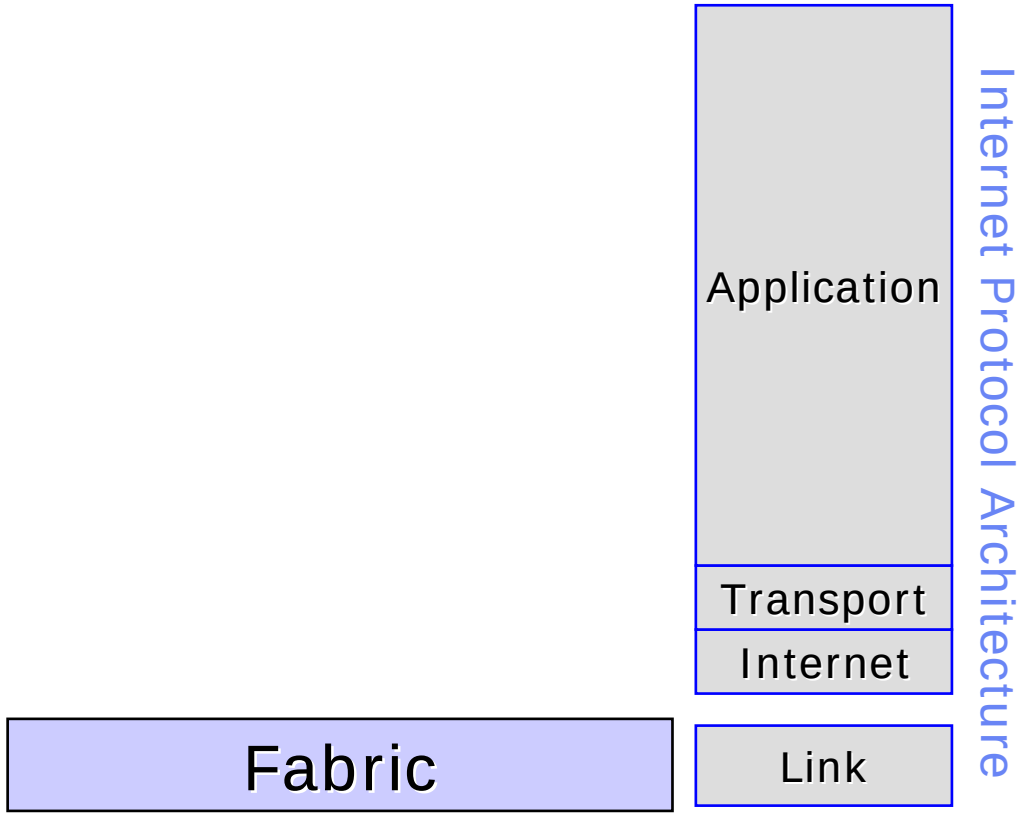
Design principles

- Keep participation cost low
- Enable local control
- Support for adaptation
- “IP hourglass” model

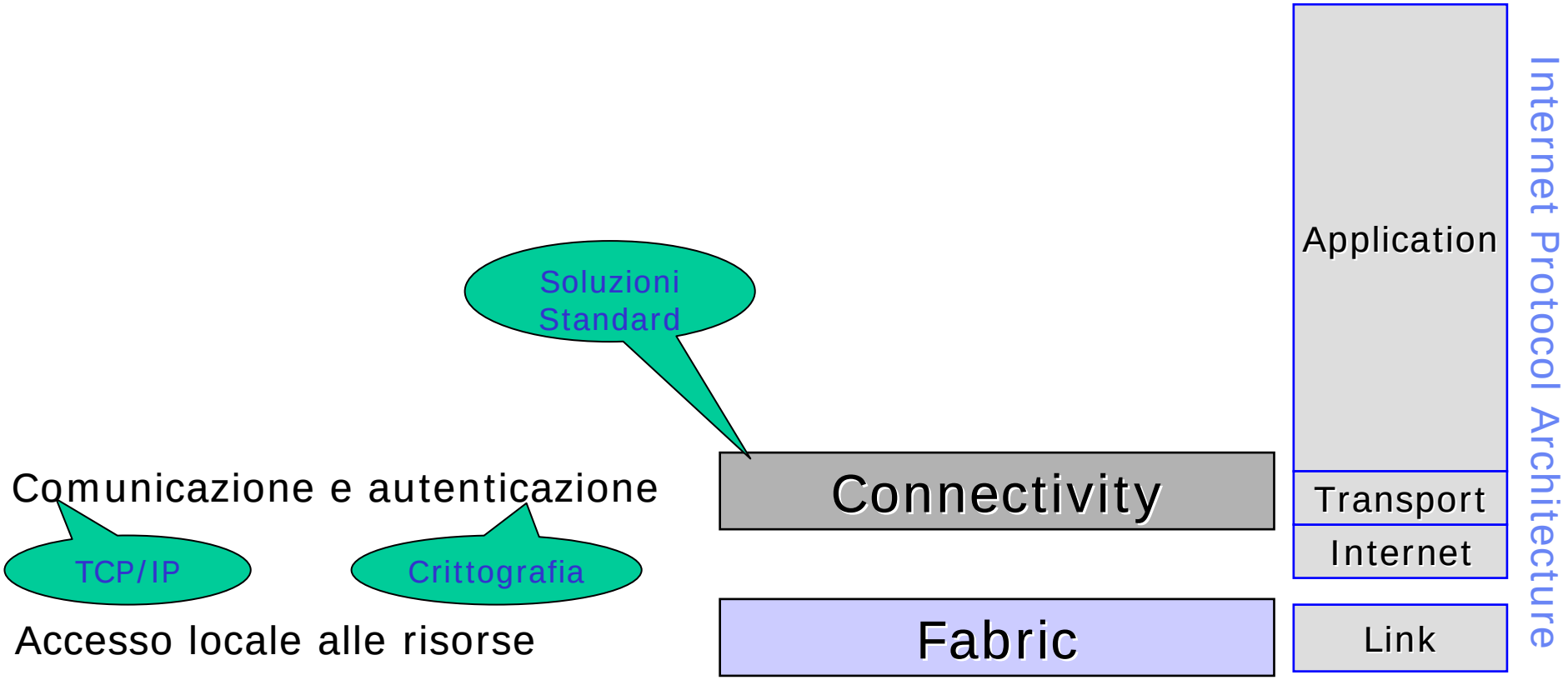


I layer della griglia: Fabric layer

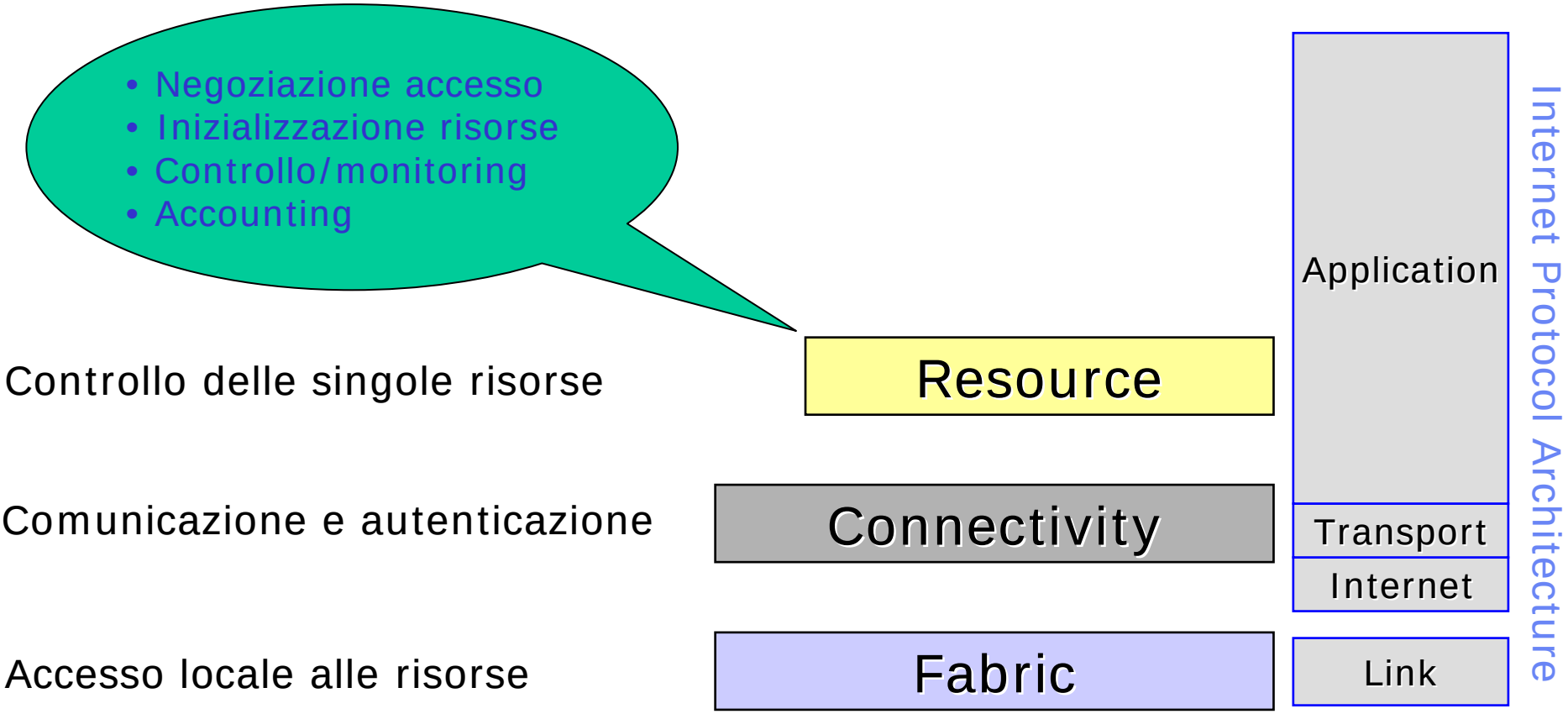
Accesso locale alle risorse



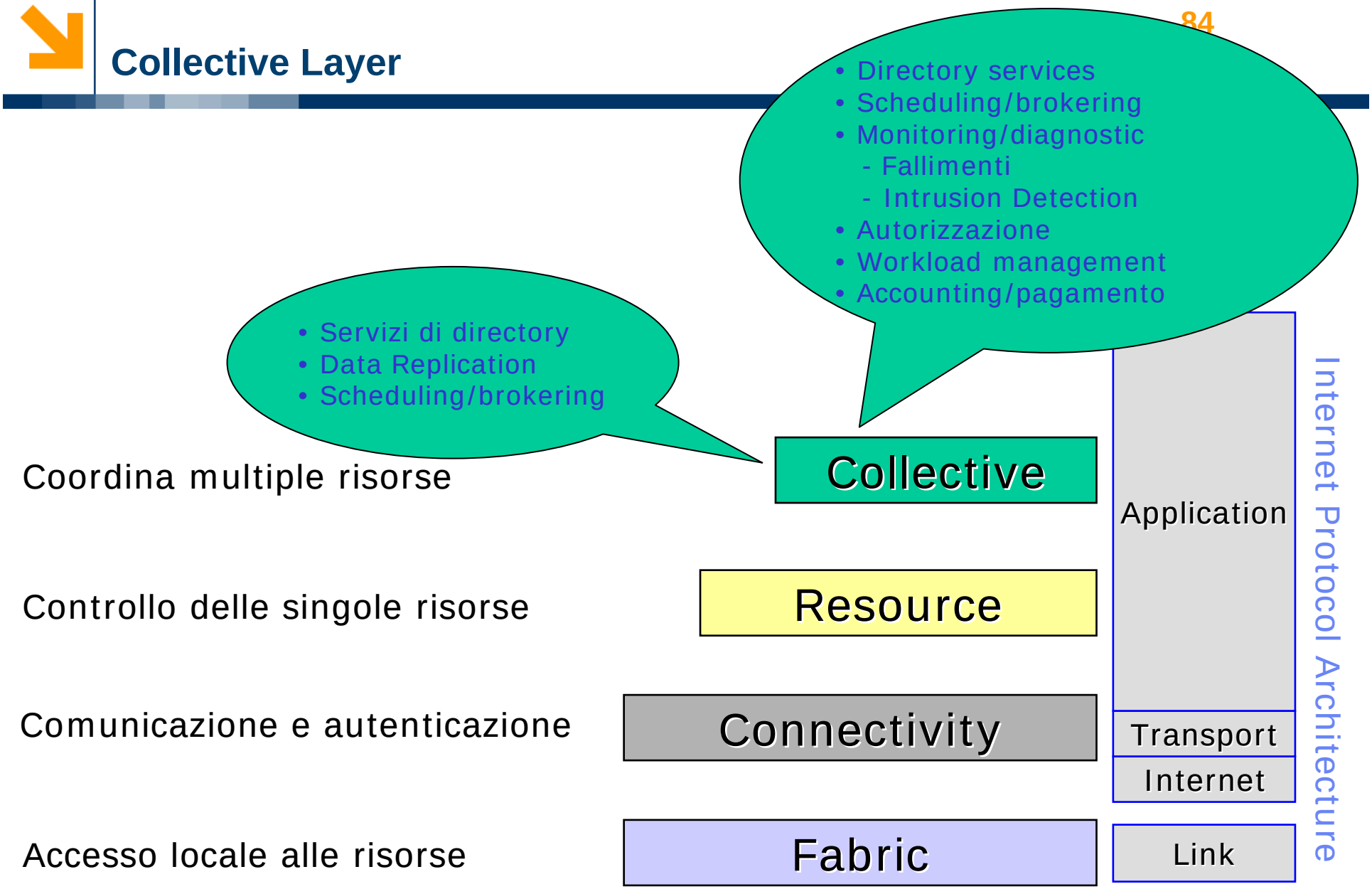
Connectivity Layer



Resource Layer



Collective Layer



Application Layer

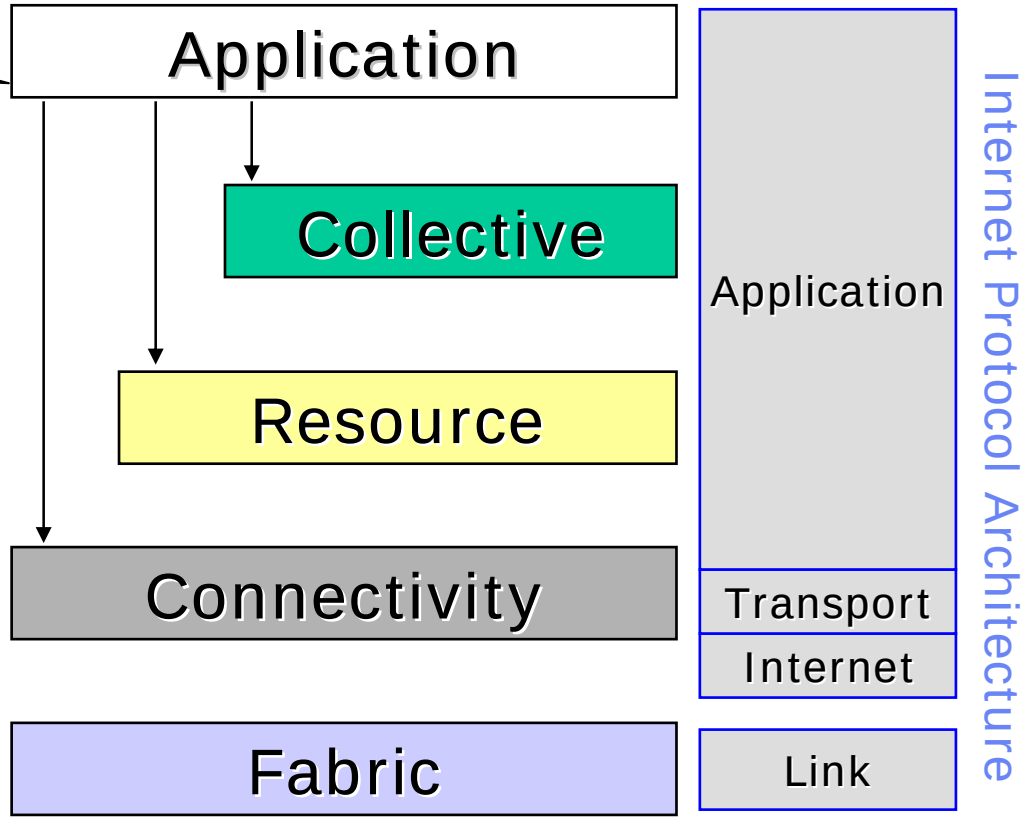
• Modelli per sviluppo rapido e robusto
• Riuso di componenti condivisi da altri

Coordina multiple risorse

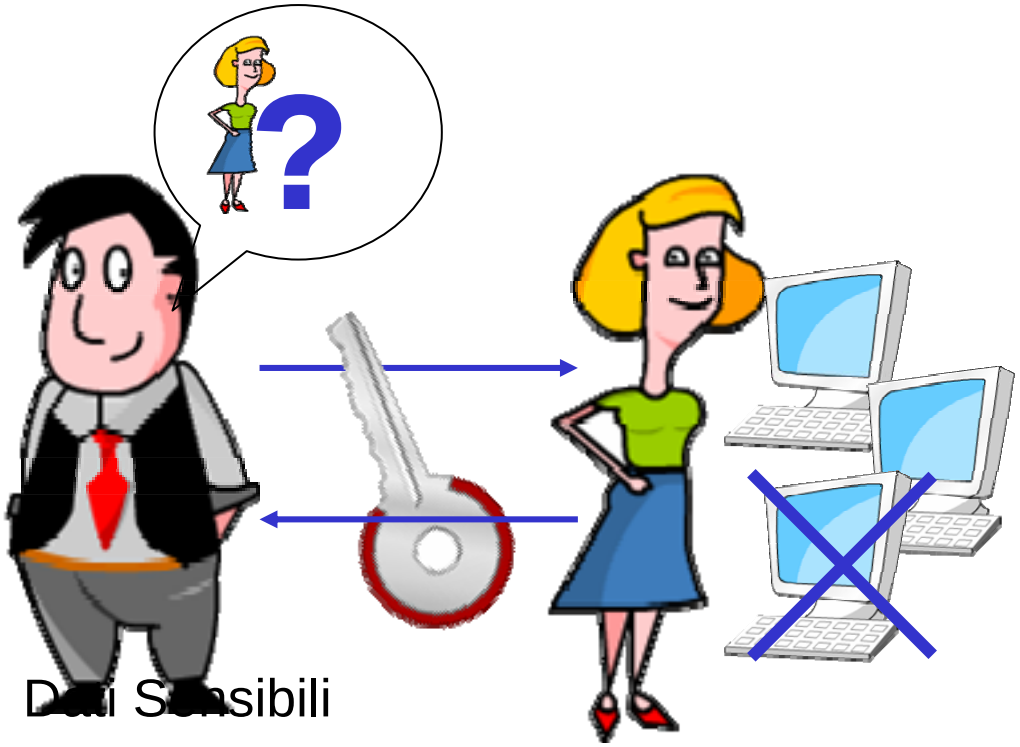
Controllo delle singole risorse

Comunicazione e autenticazione

Accesso locale alle risorse



La sicurezza



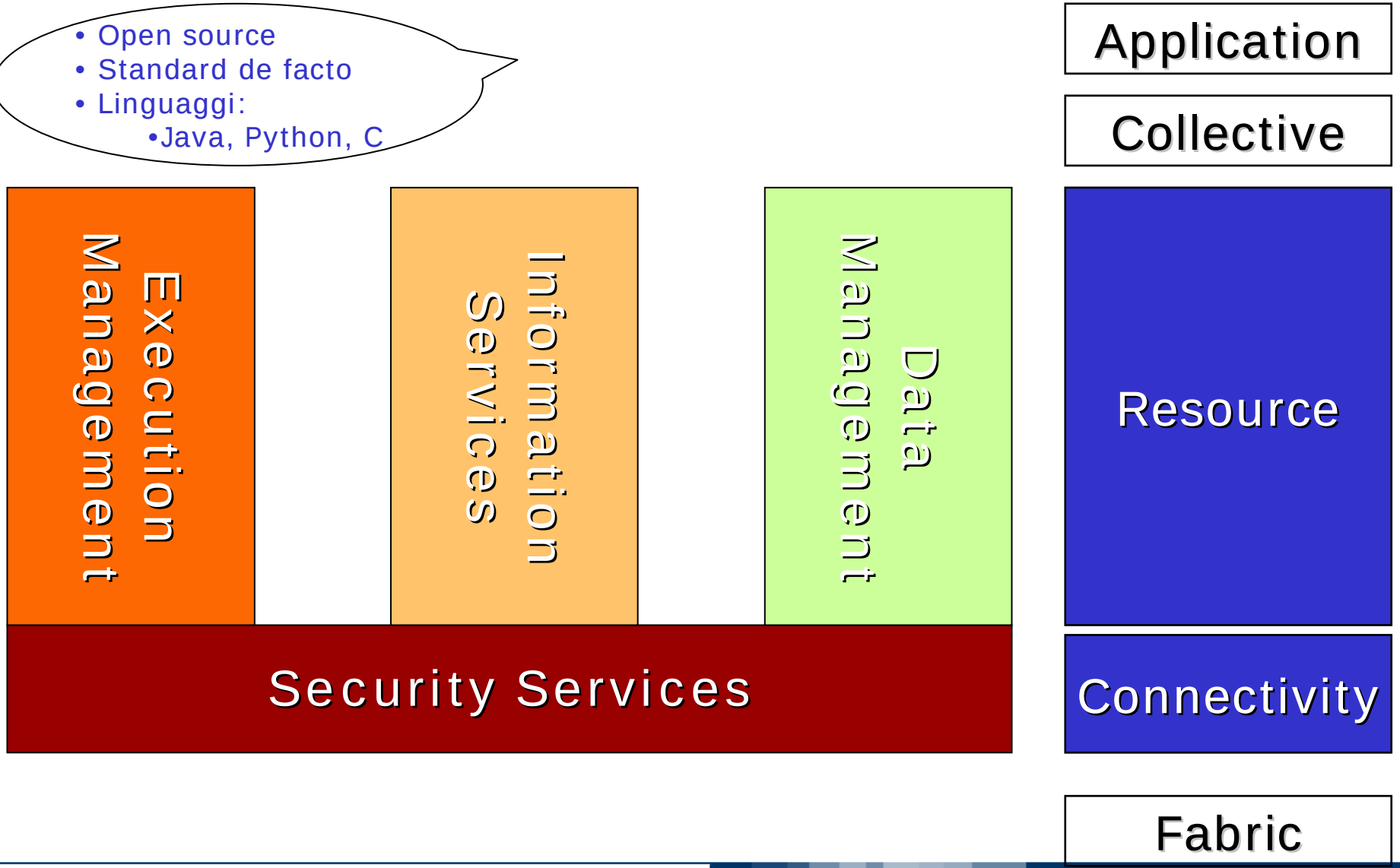
Autenticazione
Crittografia
Autorizzazione

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

- Protocolli standard e ben testati

Globus Toolkit

- Open source
- Standard de facto
- Linguaggi:
  - Java, Python, C

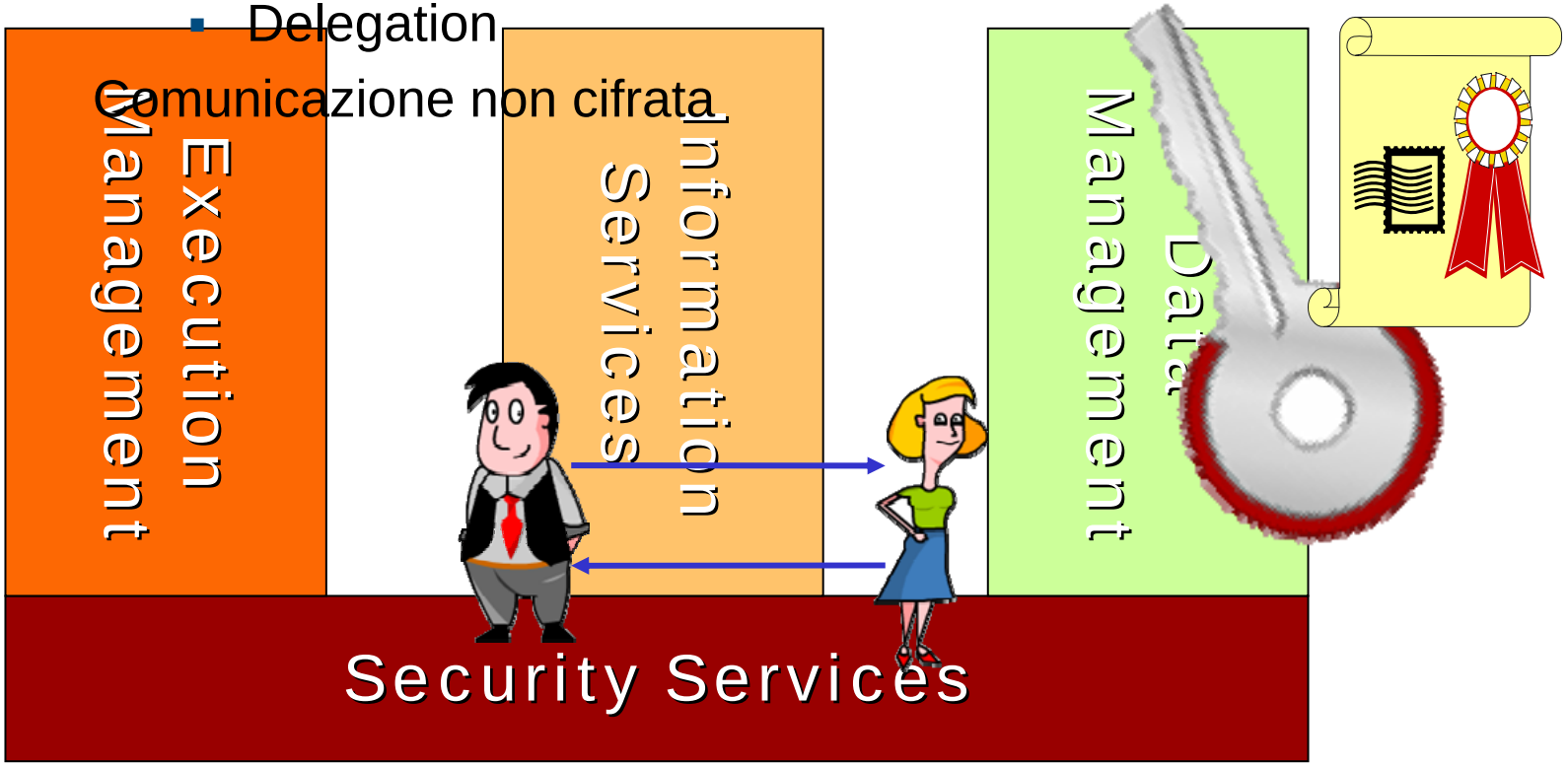


Grid Security Infrastructure

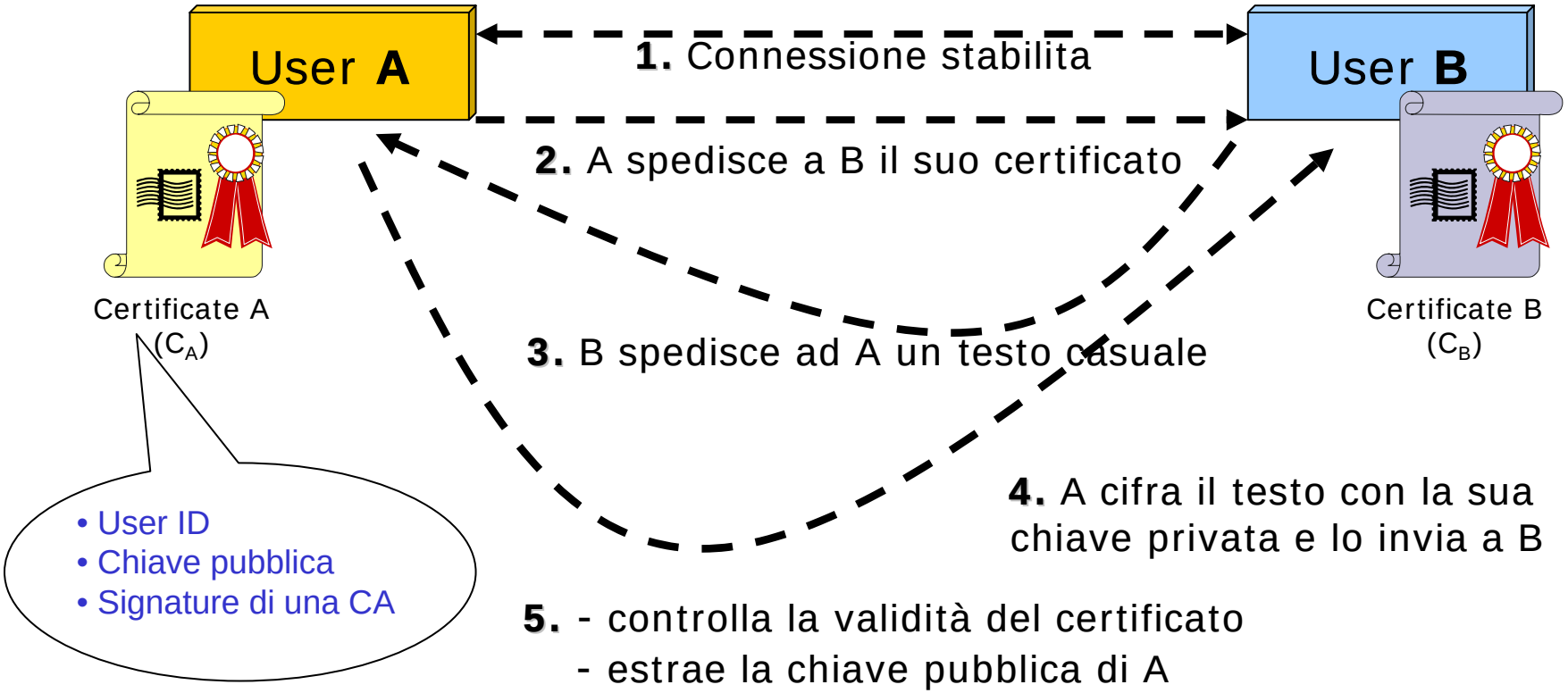
Autenticazione sicura

- X.509
- Single sign-on
- Delegation

Comunicazione non cifrata



Mutual Authentication



5. - controlla la validità del certificato
- estrae la chiave pubblica di A

6. B decrifra il testo cifrato da A e controlla che corrisponda con quello che aveva inviato in precedenza

Delegation e Single Sign-on

Sigle sign-on

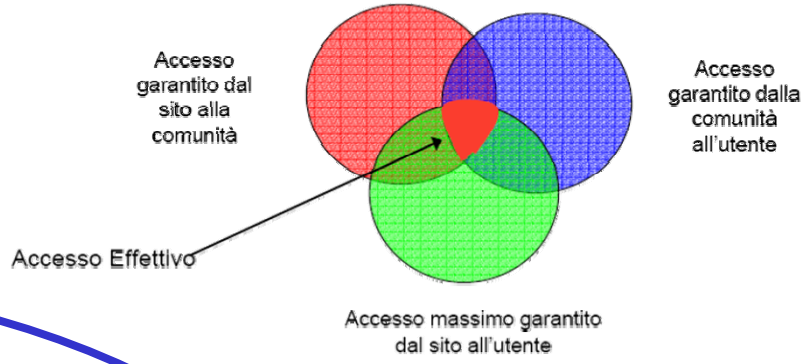
Delegation: attribuire a qualcun altro i propri diritti

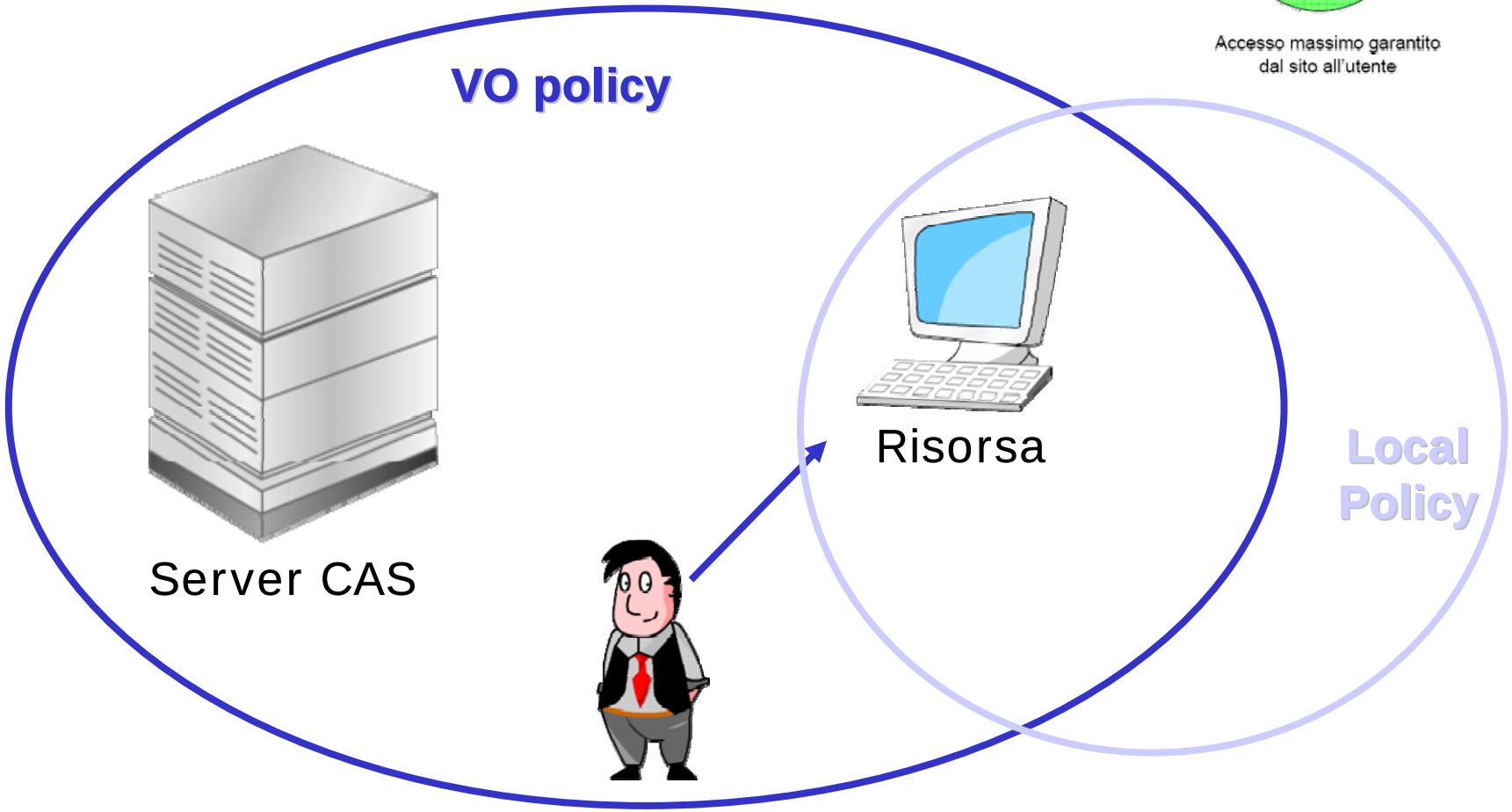
- Sito delega utenti
- Utente delega un servizio

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

CAS (Community Authorization Service)





GRAM (Grid Resouce Access and Management)

Controlla l'esecuzione dei job

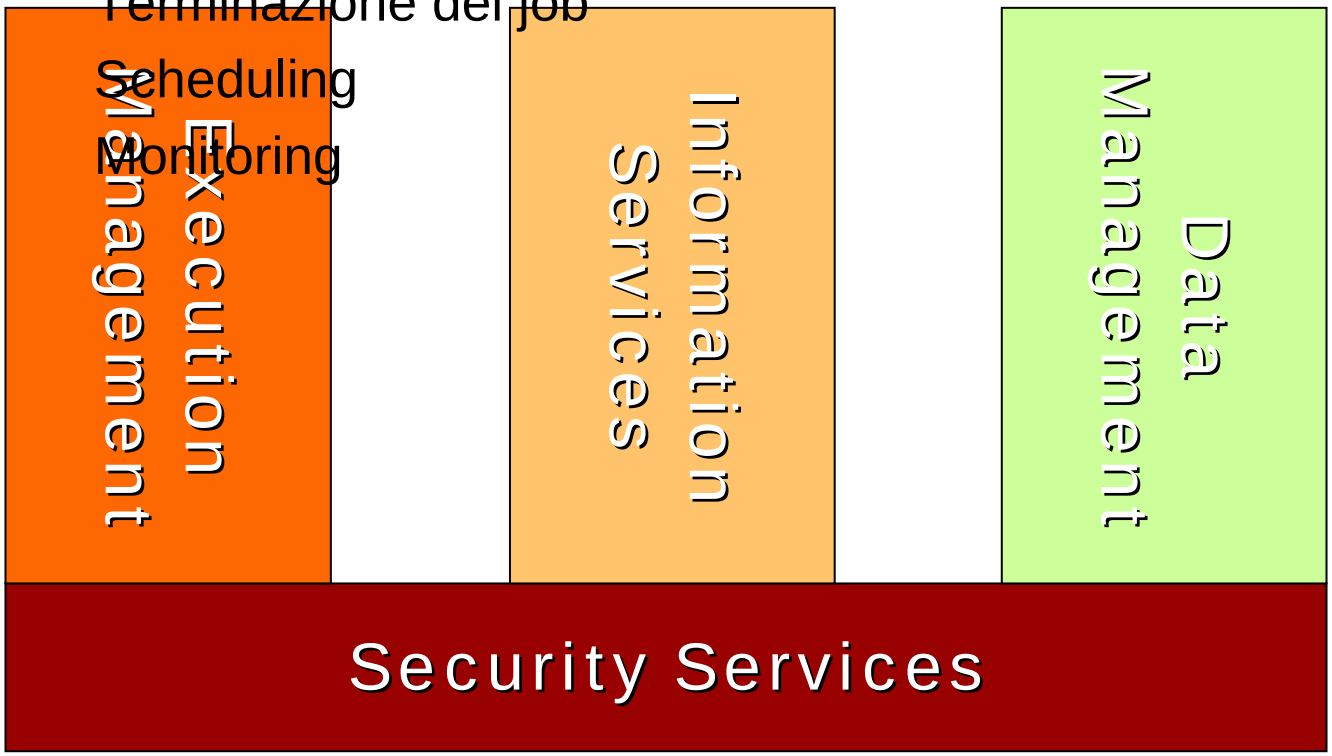
Possibilità di eseguire applicativi forniti dall'utente

Streaming dell'output

Terminazione dei job

Scheduling

Monitoring



MDS (Monitoring and Discovery System)

Accesso uniforme allo stato delle risorse

Possibilità di fare query

Servizi:

- Index Service
  - Soft-state
  - Le risorse si registrano agli *aggregator*
  - La registrazione deve essere rinnovata
- Trigger Service
- Archive Service

Data Management

Esplosione nei dati

- Accumulazione + elaborazione

Petabyte

Trovare (tutta) l'informazione rilevante

Comprendere/interpretare i dati

Gestire collezioni eterogenee

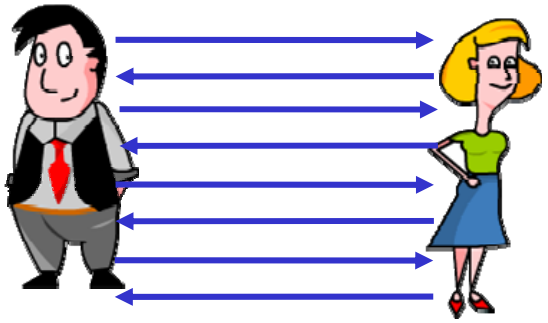
Condividere i dati

GridFTP

Estensione del protocollo FTP

Estensioni

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

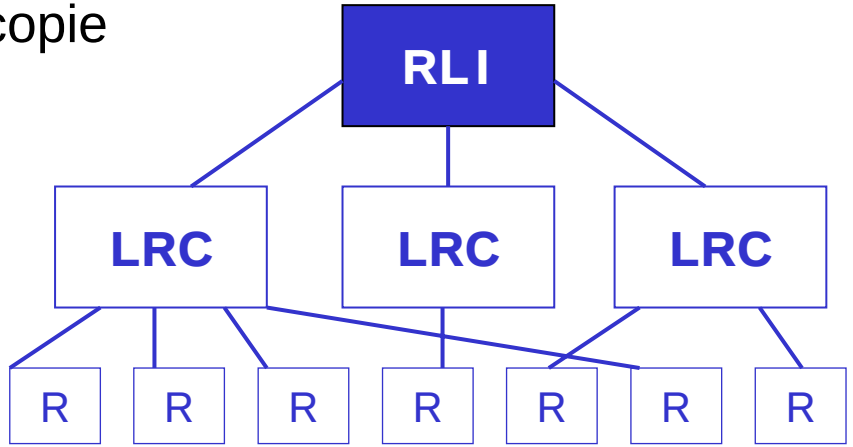


Data Management

RLS (Replica Location Service)

Replica dei dati e discovery delle copie

- Fault-tolerance
- Availibility
- Prestazioni
- Ridurre le latenze
- Load Balancing

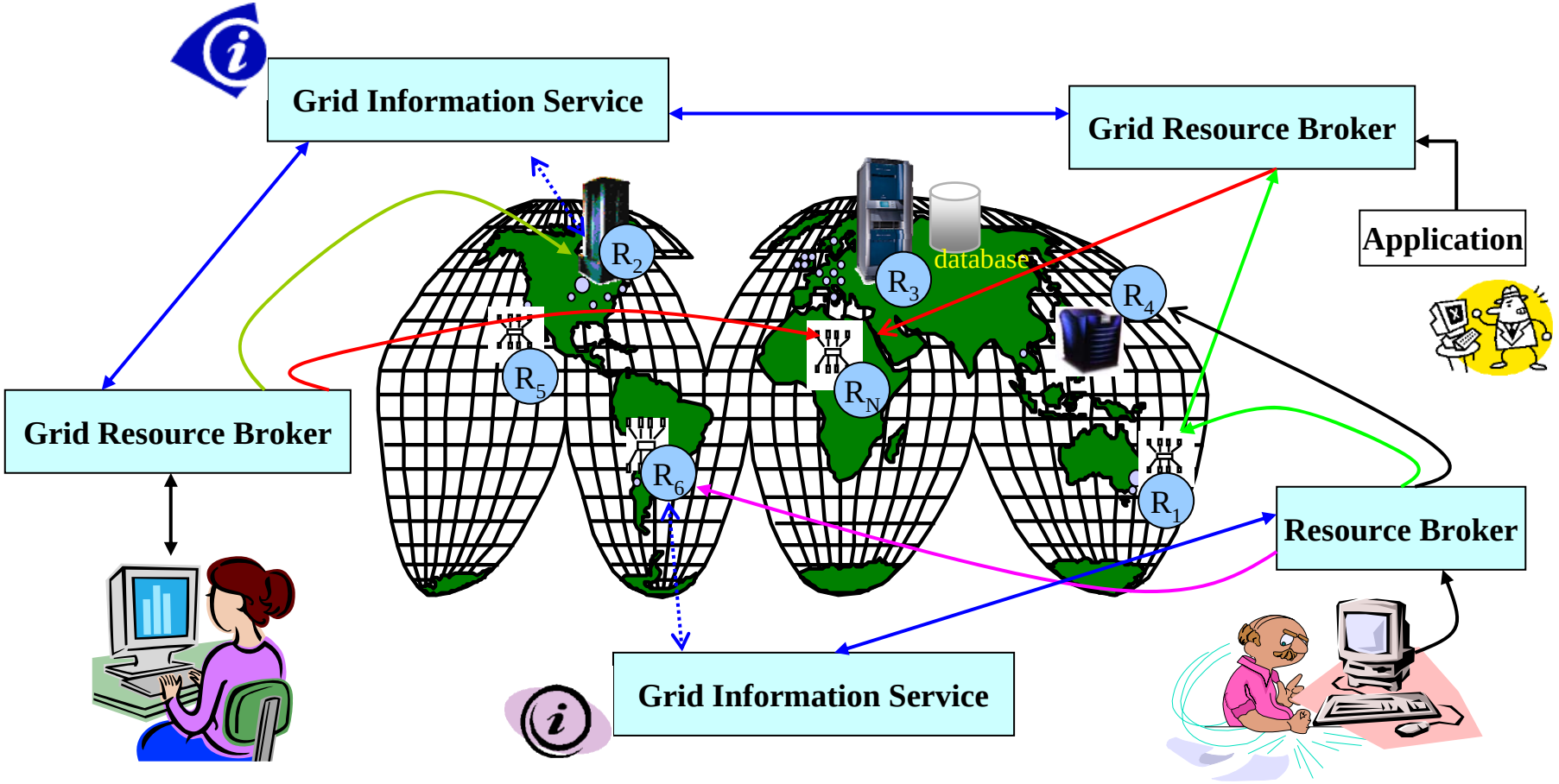


RLS

- Mapping tra nome logico e locazione
- Scalabile
- RLI (Replica Location Index): informazioni aggregate
- LRC (Local Replica Catalog): soft-state sulle singole risorse

Data Management

A Typical Grid Computing Environment



POLITECNICO DI MILANO

Impianti Informatici





Altre soluzioni grid

BOINC overview (Berkeley Open Infrastructure for Network Computing)

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

Freely distributed

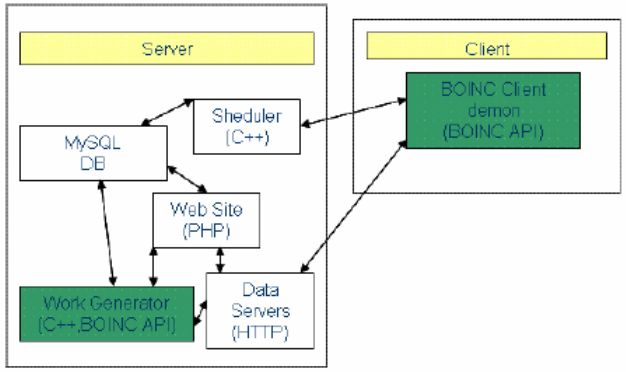
It supports applications written in different languages

- C/C++
- Fortran

Digital signature used to protect against viruses

Execution monitoring tools

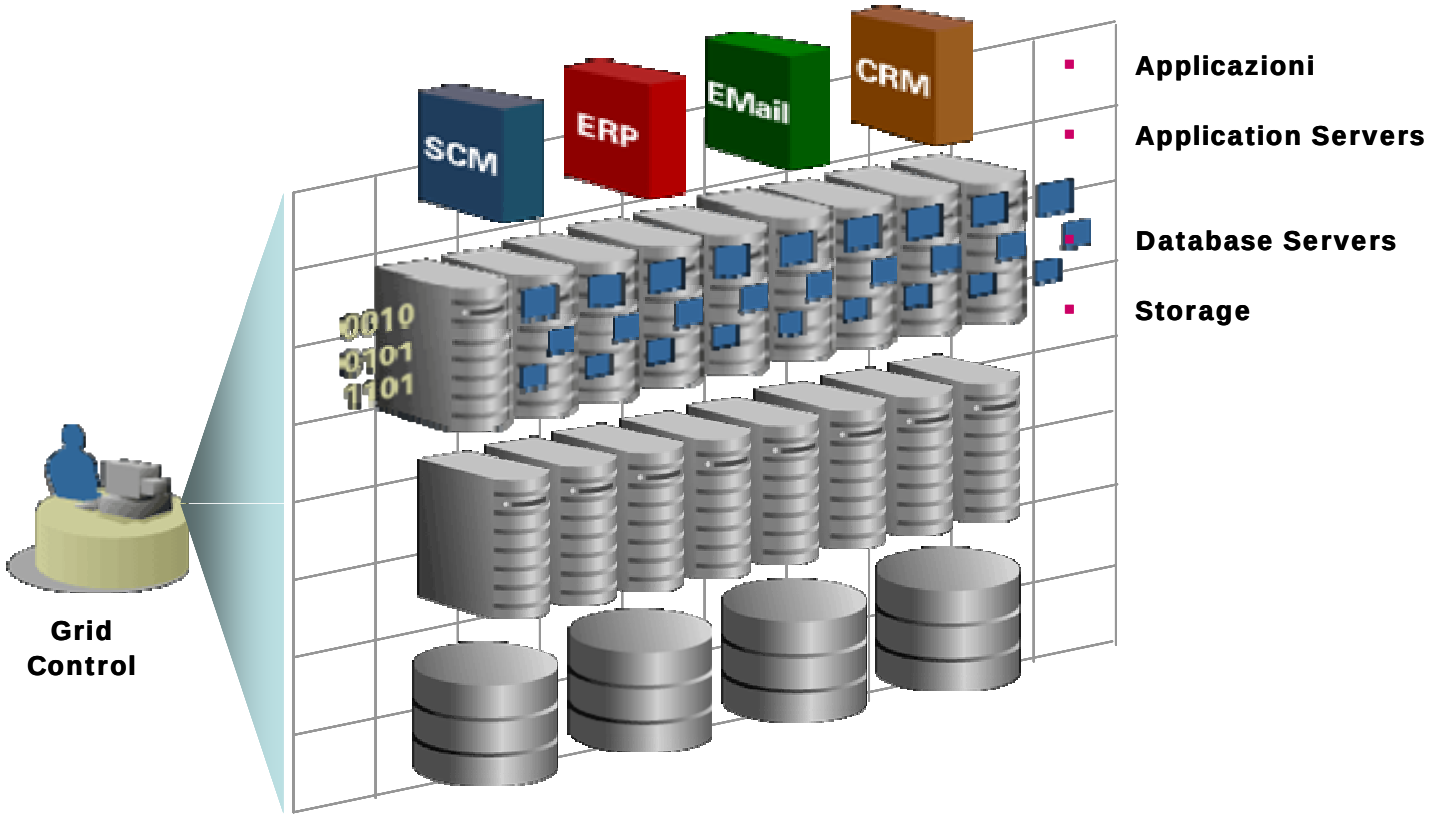
Project based on BOINC



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

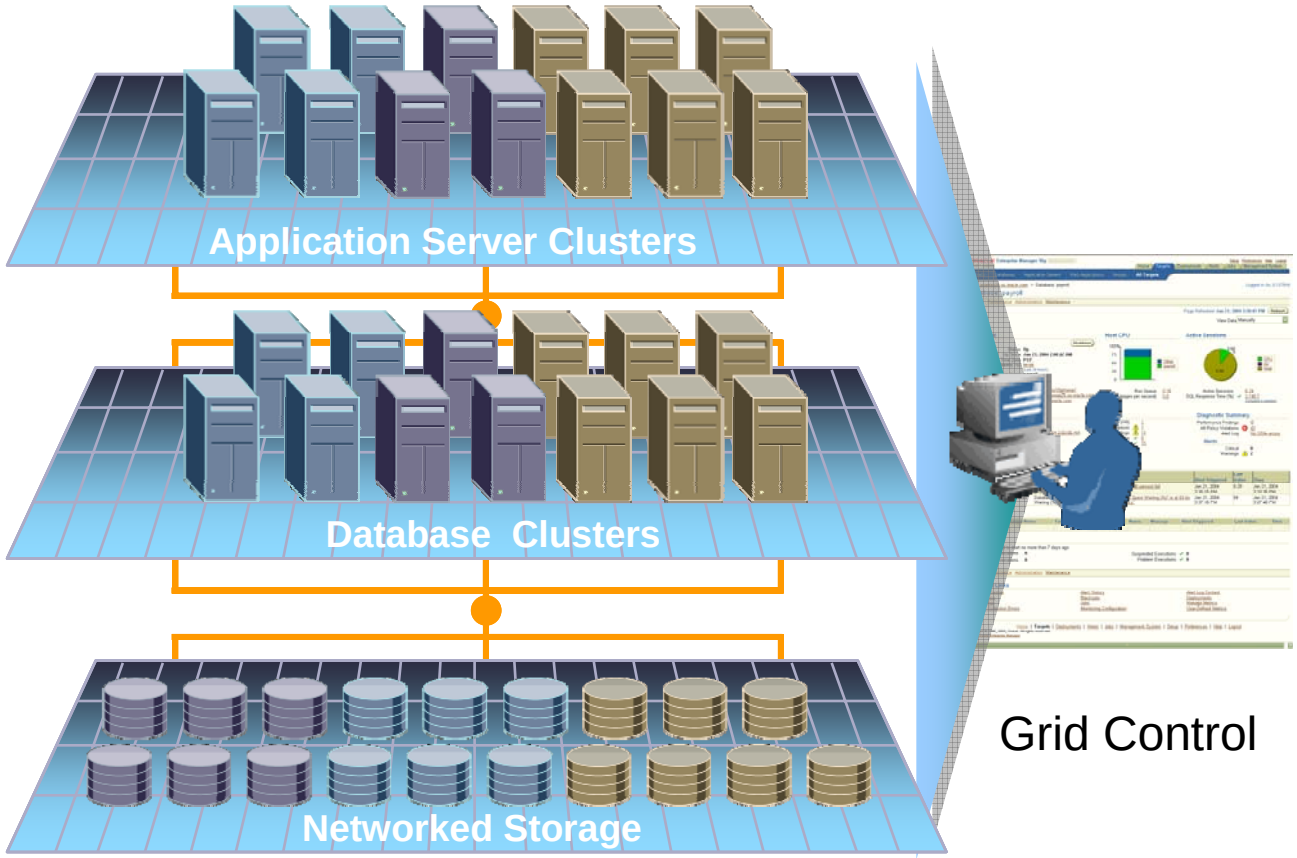
Oracle Enterprise Grid Computing

Coordinated use of many servers acting as one large computer.



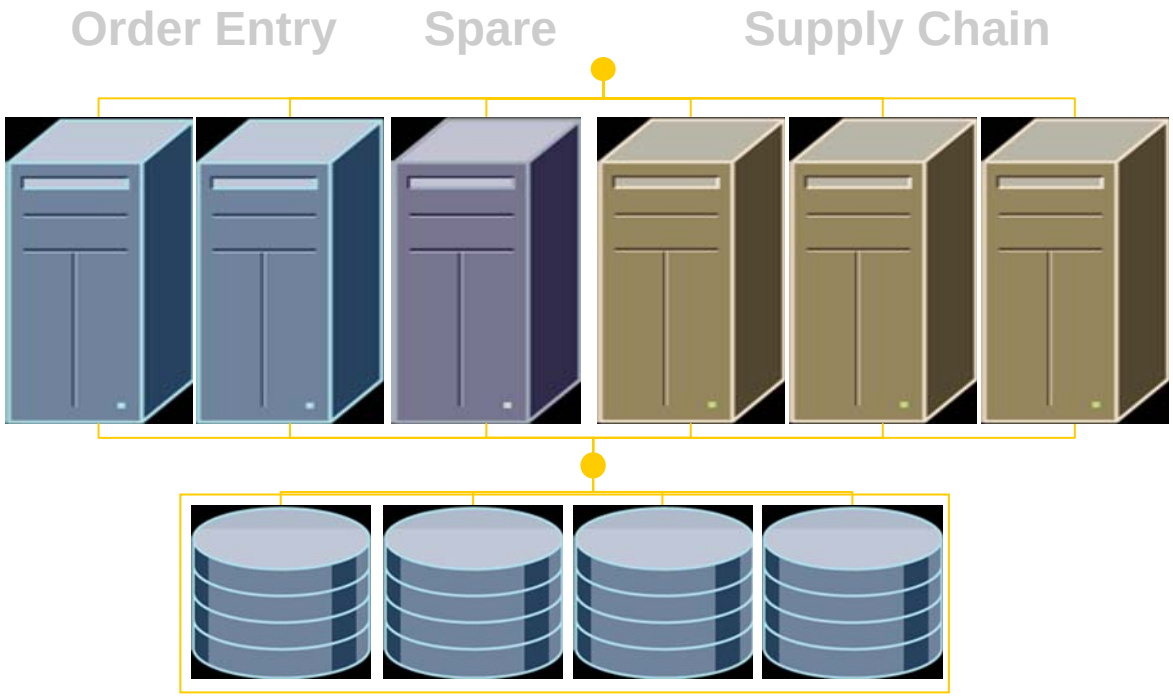
Oracle Enterprise Grid Computing

Coordinated use of many servers acting as one large computer.



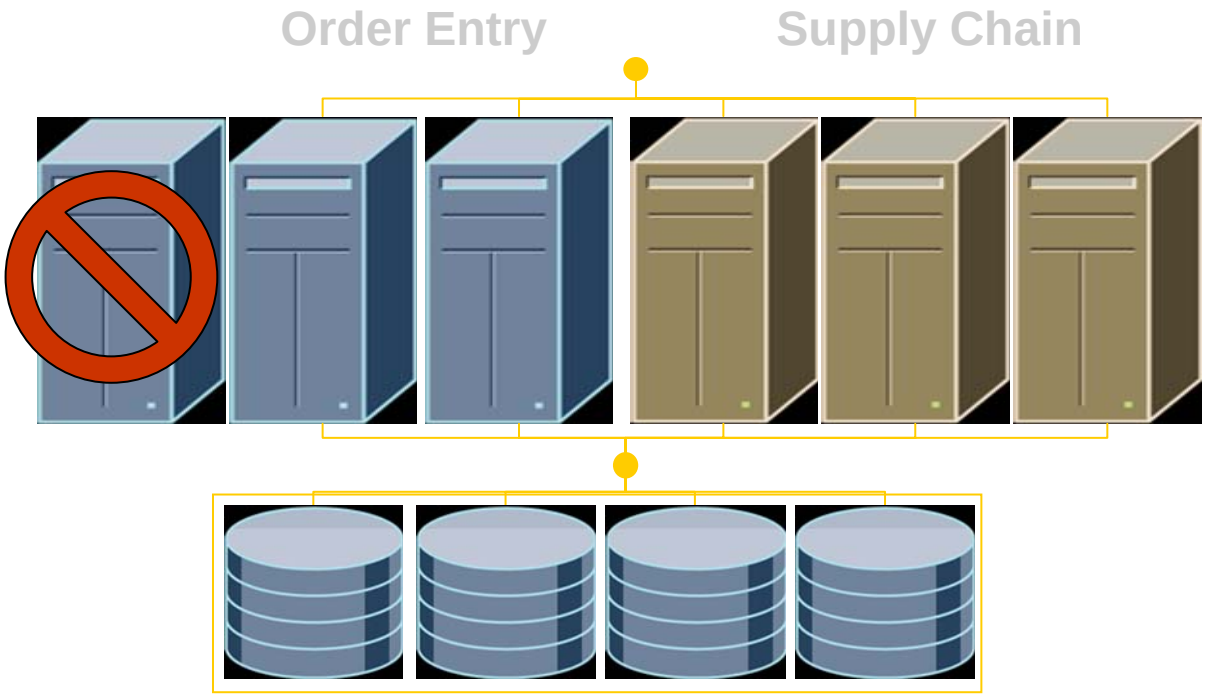
- **Pooling**
- **Virtualization & Provisioning**
- **Load Balancing**
- **Quality of Service**
- **Automation**

Automatic Workload Management



Normal application distribution

Automatic Workload Management



Recover of the Order Entry on the spare server

CPUShare overview

- 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

Message Passing Interface: www-unix.mcs.anl.gov/mpi/

Globus Project: www.globus.org

Global Grid Forum: www.gridforum.org

BOINC: boinc.berkeley.edu

Oracle Technology Network: otn.oracle.com

CPUShare: www.cpushare.com