

Curs 1

Programare Paralela si Distribuita

Continutul cursului

(realizat si pe baza pe <http://grid.cs.gsu.edu/~tcpp/curriculum/?q=home>)

Teoretic

- Notiuni introductive: arhitecturi, concurenta, paralelism
- Etape in dezvoltarea programelor paralele
- Evaluarea performantei programelor paralele
- Modele de programare paralela
 - Diferenta intre cele bazate pe memorie partajata si memorie distribuita
- *Patterns*
 - Pt programare paralela
 - Pt programare distribuita

Practic

- Java threads (low level API)
- C++ (\geq C++11) threads
- High-level API: pachete Java-> `java.util.concurrent` packages.
- Java streams
- OpenMP (C++)
- CUDA (C++)
- MPI –Message Passing Interface
 - exemplificari C, C++

Bibliografie

- Ian Foster. Designing and Building Parallel Programs, Addison-Wesley 1995.
- Berna L. Massingill, Timothy G. Mattson, and Beverly A. Sanders, Addison A Pattern Language for Parallel Programming. Wesley Software Patterns Series, 2004.
- Michael McCool, Arch Robinson, James Reinders, Structured Parallel Programming: Patterns for Efficient Computation, Morgan Kaufmann, 2012 .
- D. Culler, J. Pal Singh, A. Gupta. Parallel Computer Architecture: A Hardware/Software Approach. Morgan Kaufmann. 1998.
- Grama, A. Gupta, G. Karypis, V. Kumar. Introduction to Parallel Computing, Addison Wesley, 2003.
- D. Grigoras. Calculul Paralel. De la sisteme la programarea aplicatiilor. Computer Libris Agora, 2000.
- V. Niculescu. Calcul Paralel. Proiectare si dezvoltare formala a programelor paralele. Presa Univ. Clujana, 2006.
- B. Wilkinson, M. Allen, Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall, 2002
- A. Williams. C++ Concurrency in Action PRACTICAL MULTITHREADING. Manning Publisher.2012.
- Tutoriale Java: <http://docs.oracle.com/javase/tutorial/essential/concurrency/further.html>
- C++11 <http://en.cppreference.com/w/>
- OpenMP: <http://openmp.org/>
- MPI: <http://www.mpi-forum.org/>

Evaluare

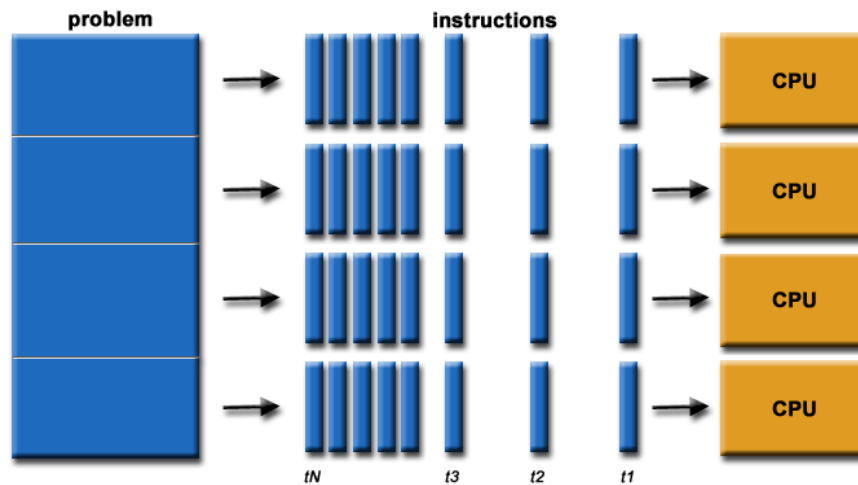
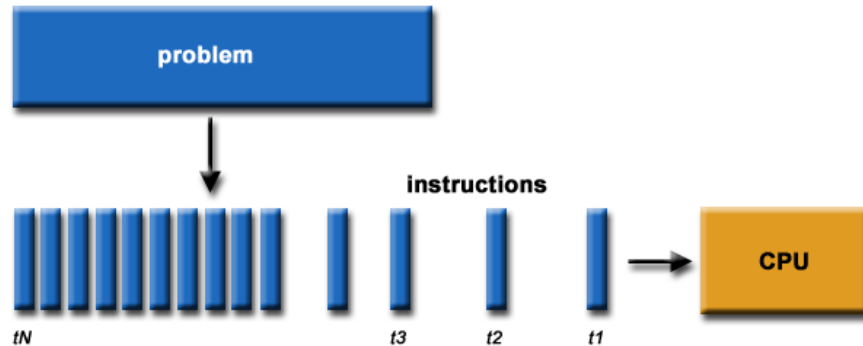
- Laborator (33%)
 - Exerciții laborator
 - Programe folosind Java/C++ threads, OpenMP, MPI
- Seminar(10%)
 - Prezenta (7 puncte <– 7 prezente)
 - Participare activa [-1(insuficient),0(suficient),1(bine),2(foarte bine)]
- Teste practice (multithreading) – 15%
- Proiect CUDA - 7%
- Examen
 - Scris – sesiune 35%
- Informatii curs
 - <http://www.cs.ubbcluj.ro/~vniculescu/didactic/PPD/CursPPD.html>

Procesare Paralela

- Un *calculator paralel* este un calculator (sistem) care foloseste multiple elemente de procesare simultana intr-o maniera cooperativa pentru a rezolva o problema computationala.
- Procesarea Paralela include tehnici si tehnologii care fac posibil calculul in paralel
 - Hardware, retele, SO, biblioteci, limbaje, compilatoare, algoritmi ...
- Paralelismul este natural.
- PERFORMANTA
 - *Parallelism is very much about performance!*

Calcul Serial vs. Paralel

(images from **Introduction to Parallel Computing** *Blaise Barney*)



“It would appear that we have reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in 5 years. “

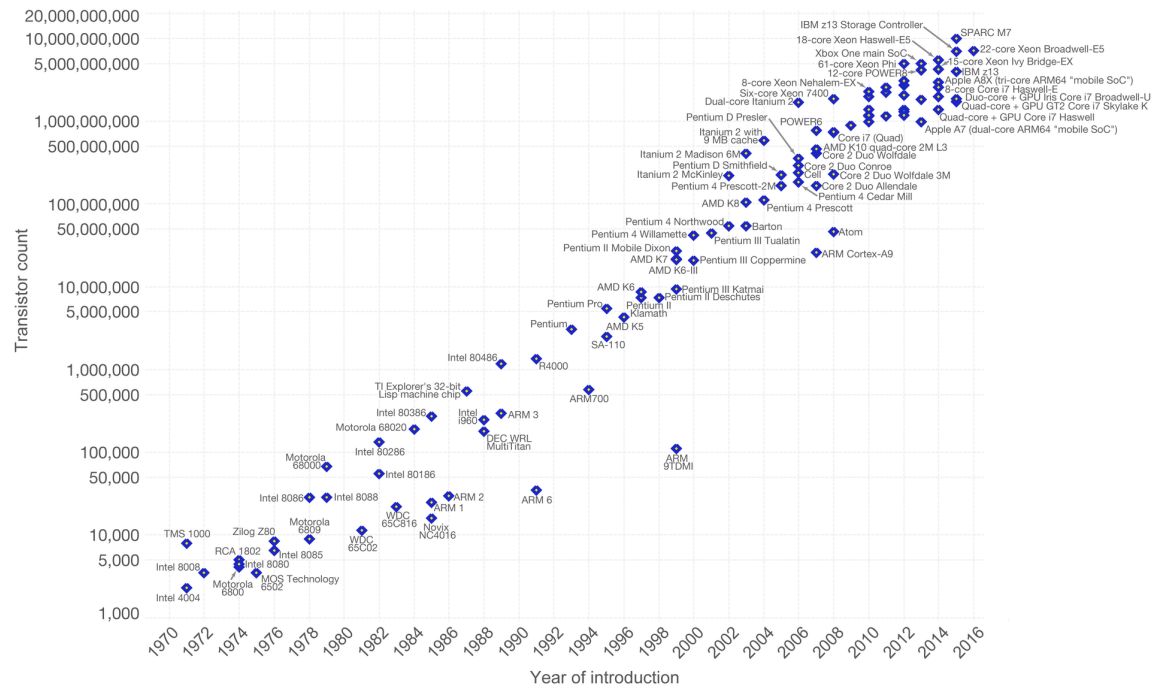
(John von Neumann, 1949)

Limite ale programarii seriale

- Viteza de transmisie –
 - Viteza luminii (30 cm/nanosecond),
 - limita de transmisie pe fir de cupru (9 cm/nanosecond).
 - Limitarea miniaturizarii – numar de tranzistori pe chip.
 - Legea lui Moore:
 - numărul de tranzistori
 - care pot fi plasati pe un singur
 - circuit integrat
 - (per square inch chip)
 - se dubleaza la fiecare 2 ani.
 - Moore's Law – The number of transistors on a chip doubles every two years. This advancement is important as other aspects of technology are strongly linked to Moore's law.
-
- | Year | Transistor Count |
|------|------------------|
| 1970 | ~2,000 |
| 1972 | ~4,000 |
| 1974 | ~8,000 |
| 1976 | ~16,000 |
| 1978 | ~32,000 |
| 1980 | ~64,000 |
| 1982 | ~128,000 |
| 1984 | ~256,000 |
| 1986 | ~512,000 |
| 1988 | ~1,024,000 |
| 1990 | ~2,048,000 |
| 1992 | ~4,096,000 |
| 1994 | ~8,192,000 |
| 1996 | ~16,384,000 |
| 1998 | ~32,768,000 |
| 2000 | ~65,536,000 |
| 2002 | ~131,072,000 |
| 2004 | ~262,144,000 |
| 2006 | ~524,288,000 |
| 2008 | ~1,048,576,000 |
| 2010 | ~2,097,152,000 |
| 2012 | ~4,194,304,000 |
| 2014 | ~8,388,608,000 |
| 2016 | ~16,777,216,000 |
| 2018 | ~33,554,432,000 |
| 2020 | ~67,108,864,000 |

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.

Our World
in Data


Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

The data visualization is available at [OurWorldinData.org](https://ourworldindata.org). There you find more visualizations and research on this topic.

Licensed under [CC-BY-SA](#) by the author Max Roser.

- Limitari economiche

Istoric

- Cresterea performantei procesor prin cresterea frecventei ceasului CPU (CPU clock frequency)
 - Riding Moore's law
- Probleme : incalzirea puternica a chipurilor!
 - Frecventa ceas mai mare \Rightarrow consum electric mai mare
(*Pentium 4 heat sink*  *Frying an egg on a Pentium 4*)
- Solutie – adugare mai multor core-uri pt a ajunge la performanta dorita
 - Se pastreaza frecventa de ceas la fel sau chiar micșorare
 - nu crește consumul.

Niveluri de paralelism

1. paralelism la nivel de job:

- intre joburi;
- intre faze ale joburilor;

2. paralelism la nivel de program:

- intre părți ale programului;
- in anumite cicluri;

3. paralelism la nivel de instrucțiune:

- intre diferite faze de execuție ale unei instrucțiuni;

4. paralelism la nivel aritmetic și la nivel de bit:

- intre elemente ale unei operații vectoriale;
- intre circuitele logicii aritmetice.

- Arhitecturile curente se bazeaza tot mai mult pe
paralelism la nivel hardware pentru a imbunatati performanta :
 - Multiple execution units
 - Pipelined instructions
 - Multi-core

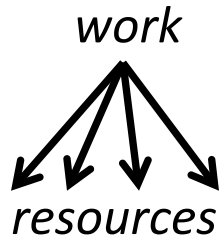
Paralelism <-> Concurenta

- Consideram mai multe taskuri care trebuie executate pe un calculator
- Taskurile se considera a fi ***pur paralele*** daca:
 - Se pot executa in acelasi timp (*parallel execution*)
- Dependente -> executie concurenta:
 - Un task are nevoie de rezultatele altora;
 - Un task trebuie sa se execute dupa ce o anumita conditie e indeplinita
 - Mai multe taskuri incearca sa foloseasca aceeaasi resursa
 - => **Forme de sincronizare trebuie folosite pentru a satisface conditiile/dependentele**
- Concurenta este fundamentala in *computer science*
 - Sisteme de operare, baze de date, networking, ...

Paralelism vs. Concurenta

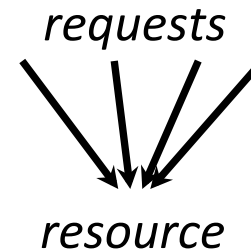
Paralelism:

Se folosesc mai multe resurse pentru a rezolva o problema mai rapid



Concurenta:

Gestiunea corecta si eficienta a accesului la resurse comune



Obs:

- Se pot folosi *threaduri* sau procese in ambele cazuri
- Daca un calcul paralel necesita acces la resurse comune atunci este nevoie sa se gestioneze corect concurenta

=> **Paralelismul poate implica concurenta**

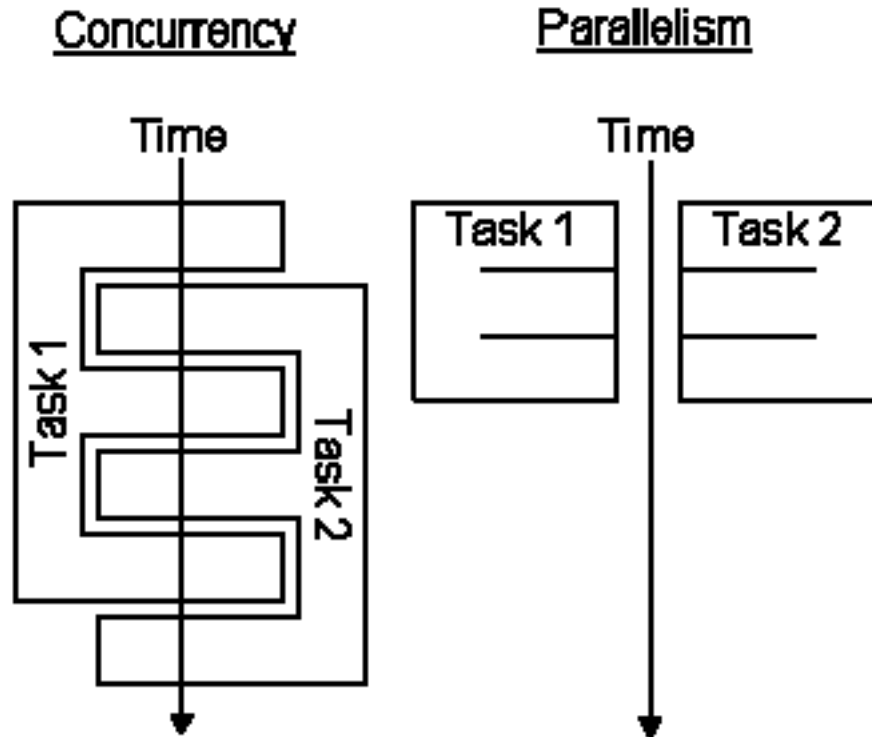
Concurenta si Paralelism

- Concurrent vs. paralel
- Executie Paralela:
 - Taskurile se executa efectiv in acelasi timp;
 - Este necesara existenta de multiple resurse de calcul
- **Paralelism = concurenta + hardware “paralel”**

Parallelism

- Exista mai multe niveluri de paralelism:
 - Procese, threads, routine, instructiuni, ...
- Trebuie sa fie suportate de resursele hardware
 - Procesoare, nuclee(cores), ... (executia instructiunilor)
 - Memorii, DMA, retele , ... (operatii asociate)

o abordare simplista



<http://www.java-programming.info/tutorial/pdf/java/11-Java-Multithreaded-Programming.pdf>

De ce sa folosim programare paralela?

- Motive primare:
 - Timp de calcul mai rapid (*response time*)
 - Rezolvarea problemelor ‘mari’ de calcul (in timp rezonabil de calcul)
- Motive secundare:
 - Folosirea efectiva a resurselor de calcul
 - Costuri reduce
 - Reducerea constrangerilor asociate memoriei
 - Limitarile masinilor seriale
- **Paralelism = concurenta + hardware ‘paralel’+ performanta**

- **Rezolvarea problemelor dificile, mari:**
 - "Grand Challenge" (en.wikipedia.org/wiki/Grand_Challenge) problems requiring PetaFLOPS and PetaBytes of computing resources.
 - Web search engines/databases processing millions of transactions per second
- **Folosirea resurselor non-locale:**
 - SETI@home (setiathome.berkeley.edu) uses over 330,000 computers for a compute power over 528 TeraFLOPS (as of August 04, 2008)
 - Folding@home (folding.stanford.edu) uses over 340,000 computers for a compute power of 4.2 PetaFLOPS (as of November 4, 2008)

Directii in procesarea paralela

- Arhitecturi paralele
 - Necesitati Hardware
 - Computer system design
- Sisteme de operare (Paralelism/concurenta)
- Gestionarea aspectelor sistem pentru un calculator paralel
- Programare paralela
 - Biblioteci (low-level, high-level)
 - Limbaje
 - Medii de dezvoltare
 - Software
- Algoritmi Paraleli
- Evaluarea performantei programelor paralele
- Testarea vs. asigurarea corectitudinii
- *Parallel tools:*
 - Performanta, analize, vizualizare, ...

De ce sa studiem programare paralela?

- Arhitecturi de calcul
 - Inovatiile conduc la noi modele de programare
- Convergenta tehnologica
 - “killer micro” este peste tot
 - Laptop-urile si supercomputere sunt fundamental similare
 - Trend-urile actuale conduc la convergenta abordarilor diverse
- Tredurile tehnologice fac calculul paralel inevitabil
 - Multi-core processors!
 - Acum orice sistem de calcul este paralel
- **Intelegerea principiilor fundamentale !!!**
 - Programare, comunicatii, memorie, ...
 - Performanta
- **“Parallelism is the future of computing” - Blaise Barney**
 - M. Andrews, J. S. Walicki. “Concurrency and parallelism—future of computing” in Proceeding of ACM '85 Proceedings of the 1985 ACM annual conference on The range of computing : mid-80's perspective. pp.224-231.

Inevitabilitatea Procesarii Paralele

- Cerintele pt aplicatii
 - Necesitatea uriasa de cicluri de calcul
- Trenduri tehnologice
 - Procesare si memorie
- Trenduri Arhitecturale
- Factori economici
- Trenduri actuale:
 - *Today's microprocessors have multiprocessor support*
 - *Servers and workstations available as multiprocessors*
 - *Tomorrow's microprocessors are multiprocessors*
 - *Multi-core is here to stay and #cores/processor is growing*
 - *Accelerators (GPUs, gaming systems)*

exemple...

- Procesor AMD Ryzen Threadripper 1950X ~ 1300USD (2017)
 - Memorie Cache 40 MB
 - Frecventa procesor (MHz) 3500
 - Turbo Boost pana la 4000 MHz
 - Numar nuclee 16 Nuclee =>32 Threads
- Intel® Xeon® Processor E7 v4 Family ~ 8000USD (2017)
 - # of Cores=24
 - # of Threads=48
 - Processor Base Frequency=2.40 GHz
 - Max Turbo Frequency=3.40 GHz
 - Cache= 60 MB

UBB CLUSTER – IBM Intelligent Cluster

- Hybrid architecture
 - HPC system +
 - private cloud



HPC – IBM NextScale

- Rpeak 62 Tflops, Rmax 40 Tflops
- 68 noduri NX360 M5, din care
 - 12 nodes with 2 GPU Nvidia K40X,
 - 6 nodes with Intel Phi
- 2 processors E5-2660 v3 with 10Cores per node
- 128 GB RAM per node, 2 HDD SATA de 500 Gb / node
- Subscription rate 1:1 between nodes based on Switch: IB Mellanox SX6512 with 216 ports
- Storage NetApp E5660, 120 HDD SAS cu 600 Gb/Hdd => total 72Tb
 - IBM GPFS 4.x -parallel file system
- IBM TS3100 Tape library for data archivation
- Operating systems on each node : RedHat Linux 6 with subscription
- Management Software: IBM Platform HPC 4.2

Private Cloud – IBM Flex System

- 10 virtualization servers Flex System x240
 - 128 Gb RAM / server
 - Procesoare 2 x Intel Xeon E5-2640 v2 / server
 - 2 x SSD SATA 240 Gb / server
- 1 management server
- Software for private cloud: IBM cloud manager with OpenStack 4.2
- Software for monitorizing and management: IBM Flex System Manager software stack
- Virtualization software: Vmware vSphere Enterprise 5.1

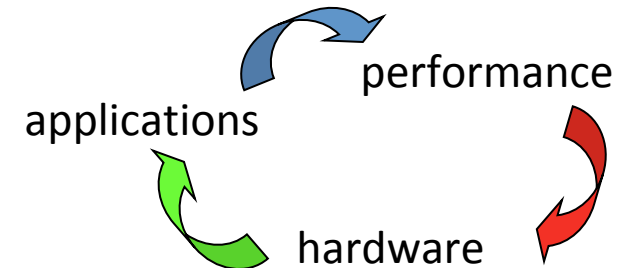
Mini cluster – IBM nestscale

~90 cores

- 4 nodes x 2 processors X 10 cores
- 1 Management node
- IP: 193.226.40.133
- **NOTA**
 - Rmax - Maximal LINPACK(benchmark) performance achieved
 - Rpeak - Theoretical peak performance.
 - Node performance in GFlops = (CPU speed in GHz) x (number of CPU cores) x (CPU instruction per cycle) x (number of CPUs per node)

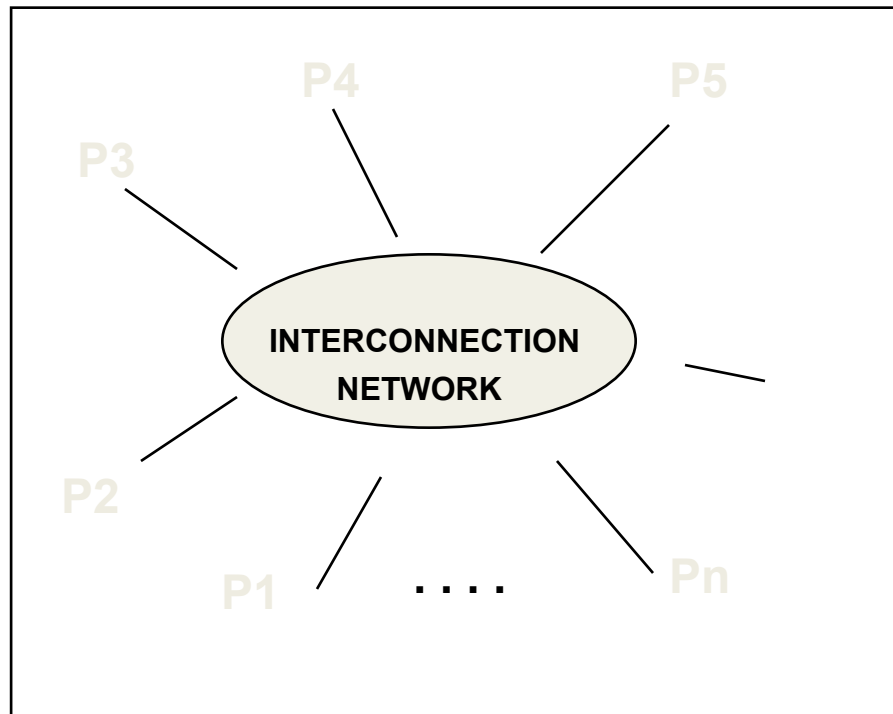
Aplicatii informatice performante

- Performanta aplicatiilor impune hardware performant (rapid, resurse multiple, etc)
- Hardware-ul avansat genereaza noi aplicatii
- Noile aplicatii au cereri de performanta mai mari
 - Crestere exponentiala a performantei microprocesoarelor
 - Inovatii in arhitecturile paralele si in integrare



- Cerinte de performanta=>
 - Performanta sistemelor trebuie sa se imbunatateasca in ansamblu
 - Schimbari/abordari/reevaluari in Software engineering
 - Costuri - tehnologie avansata

Programare paralela vs. Programare distribuita



TIPURI DE MULTIPROCESARE

PARALLEL DISTRIBUTED

ASPECTE TEHNICE

- **PARALLEL COMPUTERS** (- IN MOD UZUAL) LUCREAZA BAZAT PE
 - *CUPLARE STRANSA*,
 - in general bazate pe SINCRONICITATE,
 - CU UN SISTEM DE COMUNICATIE FOARTE RAPID SI FIABIL
 - Spatiu unic de adresare (intr-o masura mare)
- **DISTRIBUTED COMPUTERS**
 - MAI INDEPENDENTE,
 - COMUNICATIE MAI PUTIN FRECVENTA SI mai putin RAPIDA (ASINCRONA)
 - COOPERARE LIMITATA
 - NU EXISTA CEAS GLOBAL
 - “Independent failures”

SCOPURI

- **PARALLEL COMPUTERS** COOPEREAZA PENTRU A REZOLVA MAI EFICIENT PROBLEME DIFICILE
- **DISTRIBUTED COMPUTERS** AU SCOPURI INDIVIDUALE SI ACTIVITATI PRIVATE.
DOAR UNEORI INTERCOMUNICAREA ESTE NECESARA

PARALLEL COMPUTERS: COOPERARE IN SENS “*POSITIV*”

DISTRIBUTED COMPUTERS: COOPERARE IN SENS “*NEGATIV*” --
DOAR ATUNCI CAND ESTE NECESARA

In general ...

Aplicatii paralele

Suntem interesati sa rezolvam problemele *mai rapid* in paralel

Aplicatii distribuite

Suntem interesati sa rezolvam anumite probleme specifice :

- COMMUNICATION SERVICES
ROUTING
BROADCASTING
- MAINTENANCE OF CONTROL STUCTURE
TOPOLOGY UPDATE
LEADER ELECTION
- RESOURCE CONTROL ACTIVITIES
LOAD BALANCING
MANAGING GLOBAL DIRECTORIES

Un punct de vedere...

Parallel v.s. Distributed Systems

(from M. FUKUDA CSS434 System Models)

	Parallel Systems	Distributed Systems
Memory	Tightly coupled shared memory UMA, NUMA	Distributed memory Message passing, RPC, and/or used of distributed shared memory
Control	Global clock control SIMD, MIMD	No global clock control Synchronization algorithms needed
Processor interconnection	Order of Tbps Bus, mesh, tree, mesh of tree, and hypercube (-related) network	Order of Gbps Ethernet(bus), token ring and SCI (ring), myrinet(switching network)
Main focus	Performance Ex. - Scientific computing	Performance(cost and scalability) Reliability/availability Information/resource sharing

Sistemele Distribuite

-pot fi folosite pentru -

- Aplicatii distribuite implicit
 - BD Distribuite, rezervari bilete avion/etc. sistem bancar
- Informatii partajate intre useri
- Partajare resurse
- Raport cost / performanta mai bun pt aplicatii paralele
 - Pot fi folosite eficient pt. aplicatii cu granularitate mare(*coarse-grained*) si/sau pt aplicatii paralele de tip *embarrassingly parallel applications*
- Fiabilitate (*Reliability*).
- Scalabilitate
 - Cuplare slaba (Loosely coupled connection) ; hot plug-in
- Flexibilitate
 - Reconfigurare sistem pt a intruni cerintele

Performanta/Scalabilitate

Spre deosebire de sistemele paralele cele distribuite implica:

- mediu mai putin rapid de transfer al datelor (retea mai putin rapida)
- Heterogenitate

Solutii:

- Procesare *batch* a mesajelor:
 - Se evita interventia SO pt fiecare transfer de mesaj.
- Cache data
 - Se evita repetarea transferului aceleiasi date
- Evitarea entitatilor si a algoritmilor centralizati
 - Evitarea saturarii retelei
- Realizare operatii “post” la nivelul clientului
 - Evitarea traficului intens intre clienti si servere
-

Securitate

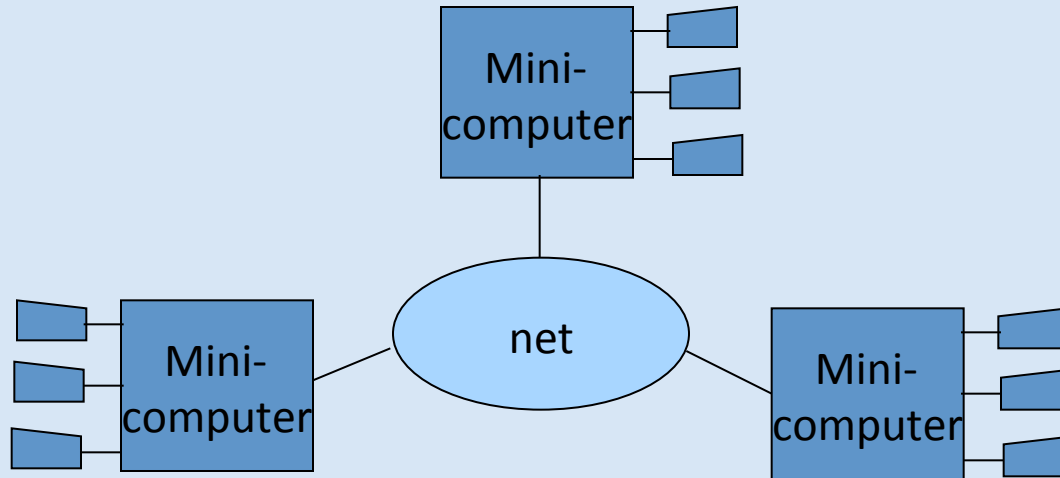
- Nu exista doar un singur punct de control
- Probleme:
 - Mesaje, furate, modificate, copiate, ...
 - Solutie : folosire Criptografie
 - Failures
 - Fault Tolerance solutions

Tipuri de sisteme distribuite

(Munehiro Fukuda – PDP Fundamentals)

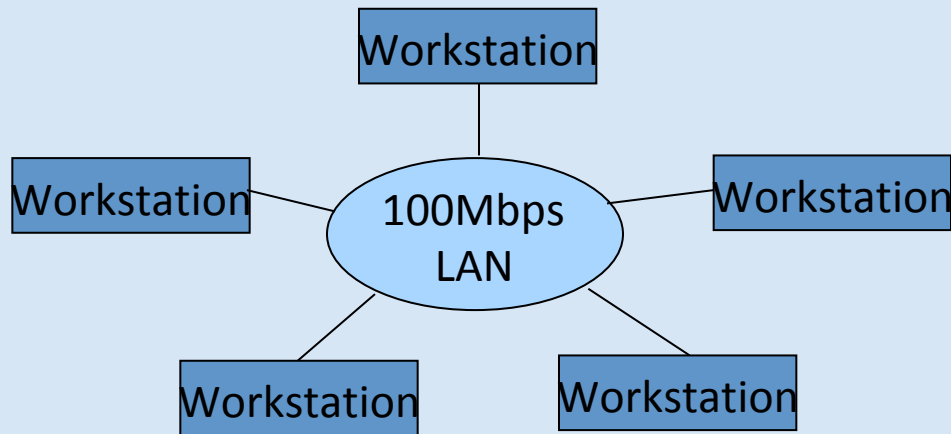
- Modele
 - Minicomputer
 - Workstation
 - Workstation-server
 - Processor-pool
 - Cluster
 - Grid computing

Modelul Minicomputer



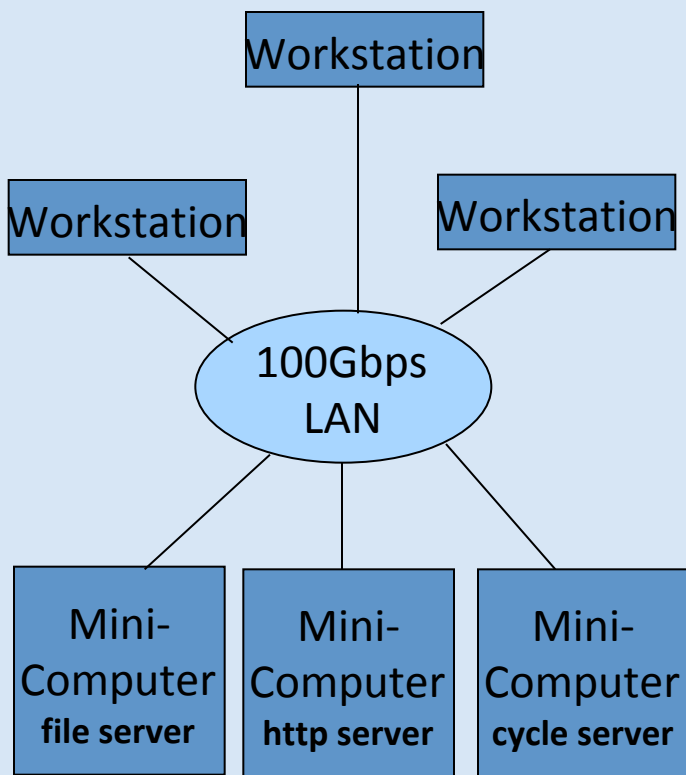
- Extensie a sistemului de Time sharing
 - Userii trebuie sa se logheze pe propriul minicomputer.
 - Apoi se logheaza la alta masina (remote machine) prin programe de tip telnet.
- Partajare resurse
 - DB
 - High-performance devices

Modelul Workstation



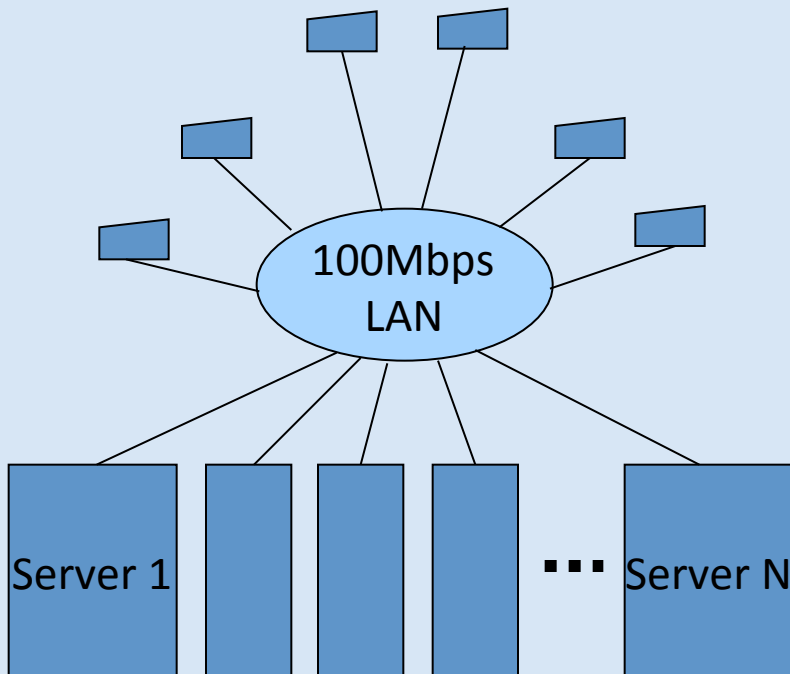
- Migrare Procese
 - Userii se logheaza mai intai pe statia de lucru personala;
 - Daca exista statii “in asteptare” – un job “mare” poate migra la una dintre ele.
- Probleme:
 - Cum se identifica statiile “in asteptare” (idle)?
 - Cum migreaza un job?
 - Ce se intampla daca un alt user se logheaza pe masina folosita?

Modelul Workstation-Server



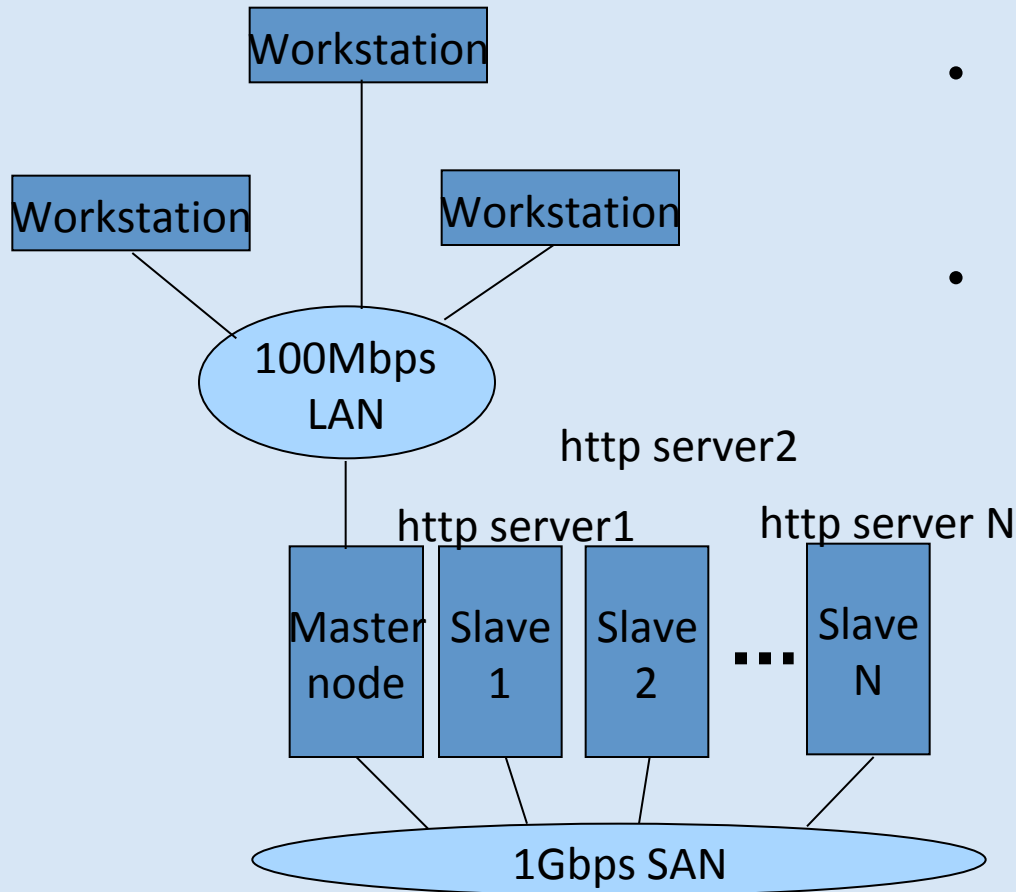
- Statii Client
 - Aplicatiile Grafice/interactive se proceseaza local
 - Pt alte cereri de calcul se trimit cereri la servere.
- Servere (minicomputers)
 - Fiecare server este dedicat unuia sau mai multor tipuri de servicii.
- Model de comunicare : Client-Server model
 - RPC (Remote Procedure Call)
 - RMI (Remote Method Invocation)
 - Un proces client cheama o functie a procesului server.
 - Nu se face migrare de procese
 - Exemplu: [NFS](#)

Modelul *Processor-Pool*



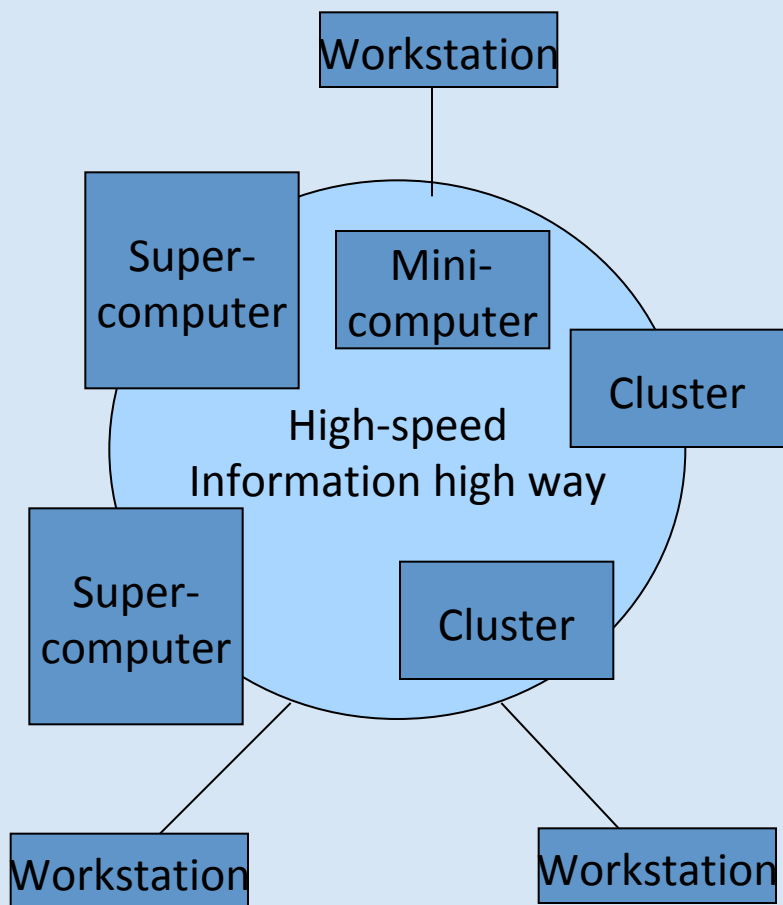
- Clientii:
 - Se logheaza la un terminal
 - Toate serviciile sunt gestionate de catre servere.
- Servere:
 - Pt fiecare user se aloca nr necesar de procesoare din *pool*.
- Utilizare buna dar interactivitate slaba.

Cluster Model



- Consta in mai multe PC/workstations conectate la o retea de tip *high-speed*.
- Focus pe performanta.

Grid Computing



- Scop
 - Colectarea puteri de calcul a mai multor supercomputere si clustere dispersate geografic
- *Distributed Supercomputing*
 - Pt problemw foarte mari. Dificile. (CPUintensive, memory intensive).
- High-Throughput Computing
 - Folosirea multor resurse care nu sunt folosite
- On-Demand Computing
 - Resurse la distanta integrate in calculul local
- Data-intensive Computing
 - distributed data
- Collaborative Computing
 - Suport pt comunicare intre mai multe parti