



Sisteme Tolerante la Defecte Introducere

Dr. Ing. Cristian Chilipirea – cristian.chilipirea@gmail.com





Who am I?

Licență Inginer
Calculatoare



Master științific
Sisteme de Calcul Paralel și Distribuit
- cum laude



Doctorat
Crowd Data Analytics As Seen From WiFi
A Critical Review



UNIVERSITY
OF TWENTE.



Who am I?

Student Assistant

- Concurrency & Multithreading



Asistent Universitar

- Algoritmi Paraleli și Distribuți
- **Arhitecturi Paralele**
- Programare Web
- Protocoale de Comunicație
- Programarea Calculatoarelor





Who am I?

Lector

- Structuri de Date și Algoritmi
- Sisteme Tolerante la Defecte
- Arhitecturi Paralele



Senior Technical Program Manager

- Azure Core







Așteptări?

Ce vă așteptați să facem la Sisteme Tolerante la Defecte?

Cu ce tehnologii vă așteptați să lucrăm?

Ce vă așteptați să învățați?

Ce sistem/mecanism de toleranță la defecte cunoașteți?



Regulament

Rezolvarea laboratoarelor este **obligatorie**

- Se rezolvă exclusiv pe platformă.
- Se pot prezenta cu o mică întârziere.

Laboratoarele se rezolvă **individual** și vor fi verificate **anti-plagiat**

Temele se rezolvă **individual** și vor fi verificate **anti-plagiat**

Orice plagiat detectat va duce la cerere exmatriculare



Indicații prevenire plagiat

Este **încurajată** folosirea Internetului. **Orice preluare trebuie corect citată.**

Laborator

- **NU** este permis mutat cod de la un student la altul.
- Este permis uitat scurt unul peste codul celuilalt, arătat un detaliu mic.
- Este încurajată discutarea problemelor.

Teme

- **NU** este permis copiat de cod (de pe net, de la coleg, de la terț).
- **NU** este permis văzut codul unui coleg.
- Se pot discuta probleme punctuale, dar NU spus explicit soluția.

Examen

- **NU** este permisă nici o interacțiune cu alte persoane.



Notare

6 puncte Teme

- **Minim 50%** punctaj pe **fiecare** temă pentru **promovare**

2 puncte Examen parțial

- **Minim 50%** pentru **promovare**

2 puncte Examen final

- **Minim 50%** pentru **promovare**



Objective

Dezvoltarea abilităților pentru:

Proiectarea și implementarea aplicațiilor distribuite

Depanarea unor aplicații distribuite

Demonstrarea corectitudinii și scalabilității unui program distribuit

Proiectarea și implementarea sistemelor de servicii bazate pe containere și orchestrare

Proiectarea și implementarea aplicațiilor Cloud

Dezvoltarea, implementarea și utilizarea tehnicilor pentru obținerea consistenței și rezilienței unui sistem





Cauze Defecte în calculatoare:

Probleme software

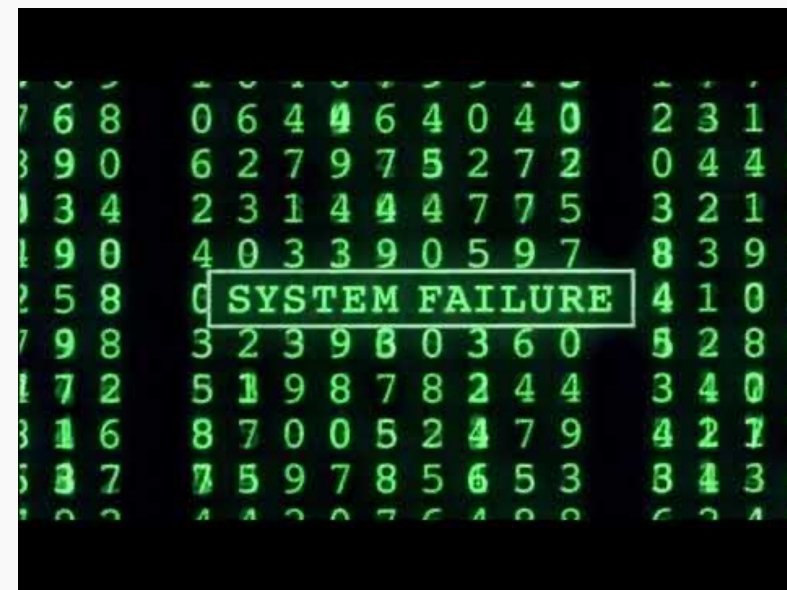
- Presupuneri greșite
- Erori de design, de logică sau de programare (ex: Bug-uri)
- Folosire neașteptată sau necorespunzătoare

Depășirea resurselor disponibile

Hardware

- Rezistență limitată în timp
- Supraîncălzire
- Supratensiune

Multe, multe, multe.... Multe altele.





Mai mult decât toleranță la defecte

Trebuie să:

Acceptăm defectele

Așteptăm defectele

Fim îngrijorați când nu identificăm defecte

Fim îngrijorați dacă nu am avut un defecte de prea mult timp

“A pessimist is never disappointed”



Mai mult decât toleranță la defecte

*“Everything fails,
All the time”*



Werner Vogels –Amazon CTO



Mai mult decât toleranță la defecte

*“Wear your failure
as a badge of
honour”*



Sundar Pichai – Alphabet CEO



Mai mult decât toleranță la defecte

*“Microsoft is
always two years
away from failure”*



Bill Gates – Founder of Microsoft



Un singur sistem mereu se va defecta

Soluția?

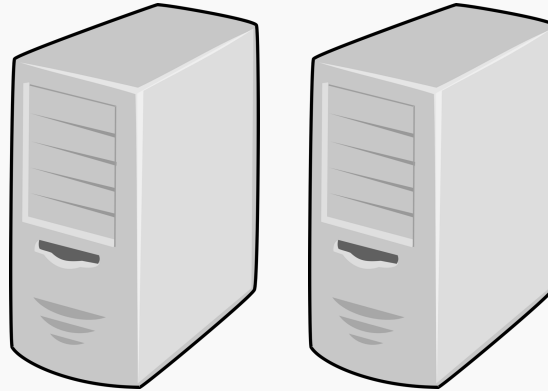




Un singur sistem mereu se va defecta

Soluția?

2 sisteme





Un singur sistem mereu se va defecta

Soluția?

2 sisteme



Dar dacă unul se defectează? (am reveni la cazul precedent)



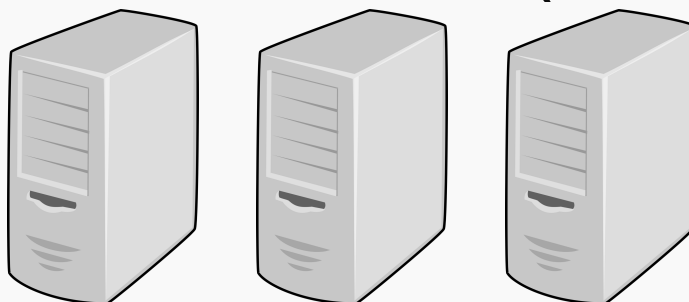
Un singur sistem mereu se va defecta

Soluția?

2 sisteme

Dar dacă unul se defectează? (am reveni la cazul precedent)

3 sisteme





Un singur sistem mereu se va defecta

Soluția?

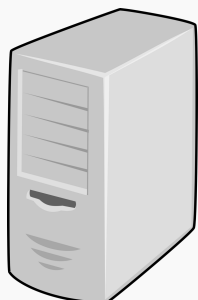
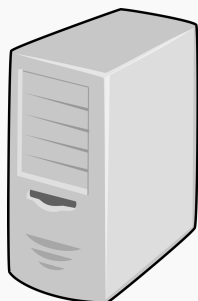
2 sisteme



Dar dacă unul se defectează? (am reveni la cazul precedent)

3 sisteme

Deci 5 sisteme, să fim siguri.





Deci...

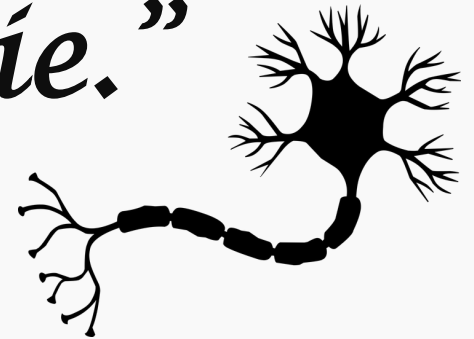


Deci... Programare Distribuită



Programare distribuită

“Studierea unui neuron se numește neuroștiință. Studierea a doi neuroni se numește psihologie.”



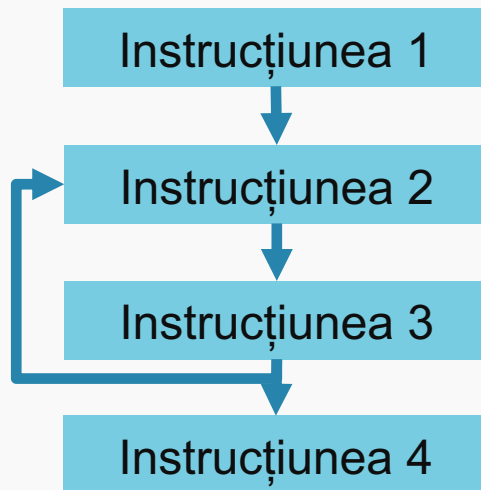
În cazul nostru, programarea distribuită reprezintă programarea a cel puțin două sisteme de calcul pentru rezolvarea unei probleme.



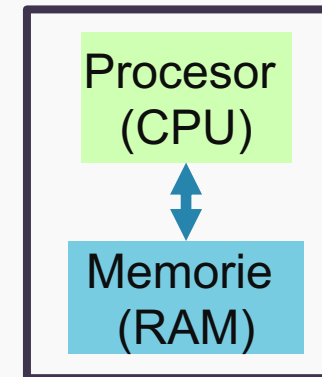
Calcul Paralel vs Distribuit vs Secvențial



Calcul Secvențial

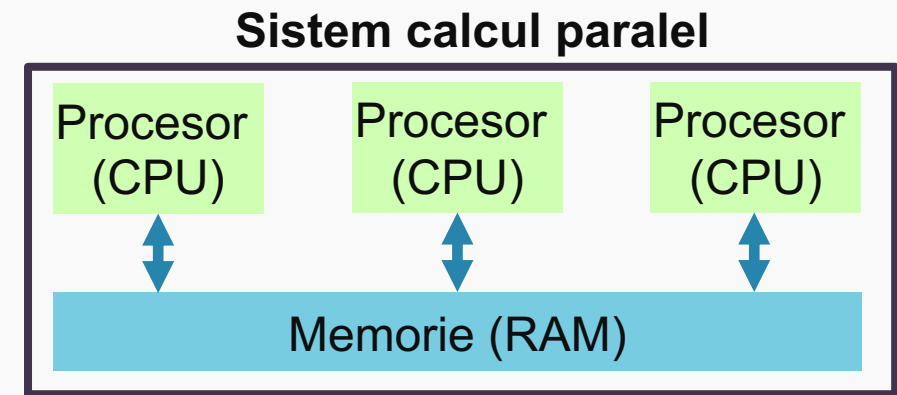
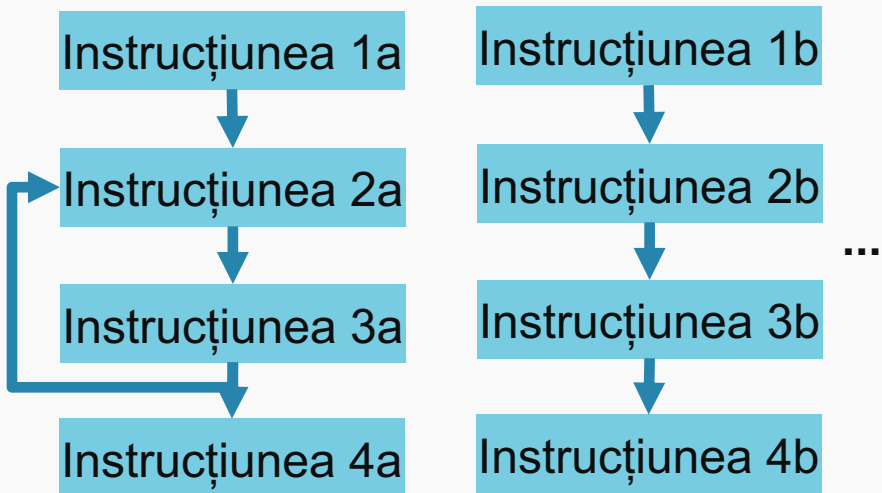


Sistem calcul secvențial



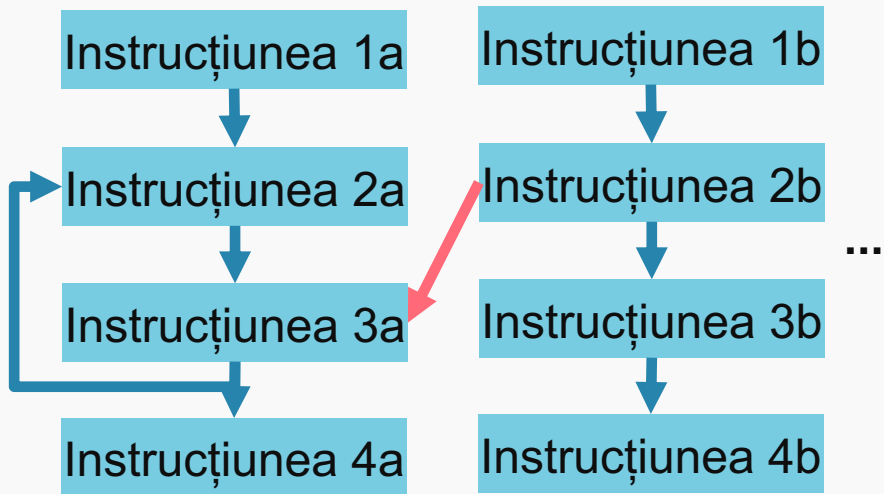


Calcul Paralel

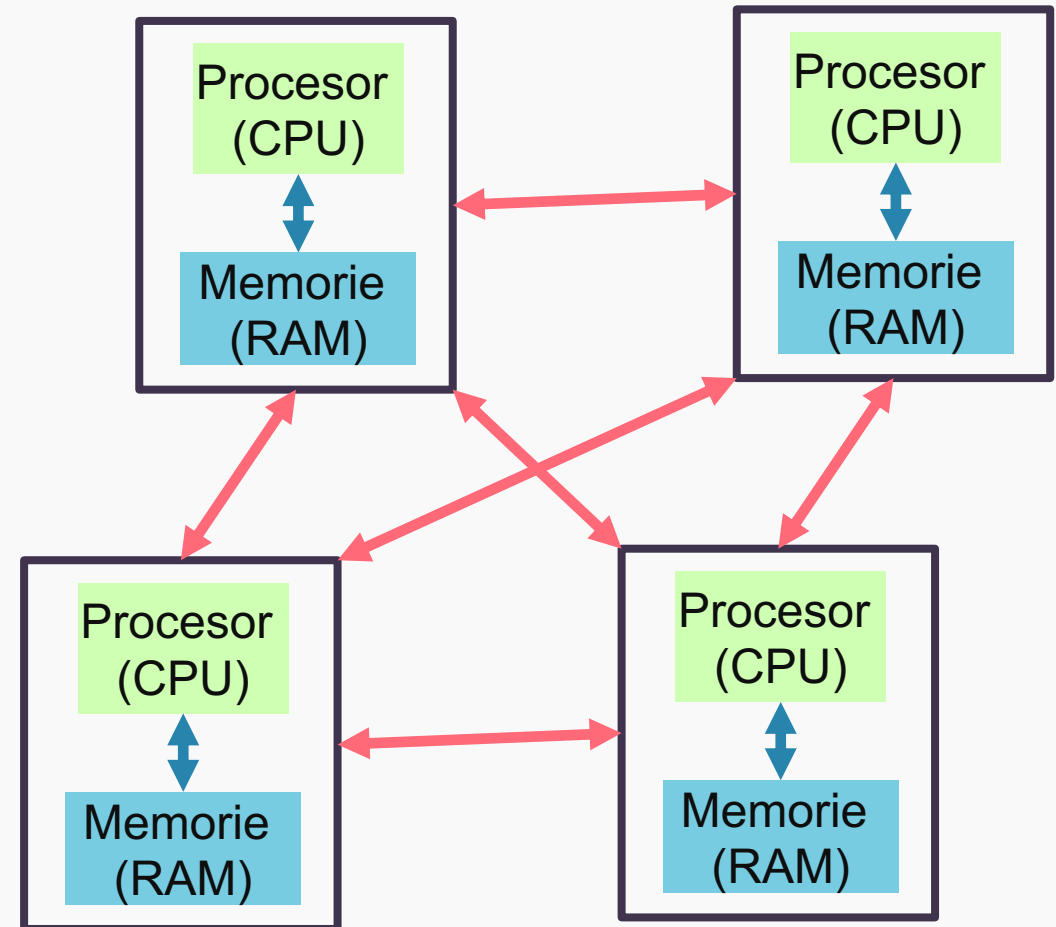




Calcul Distribuit



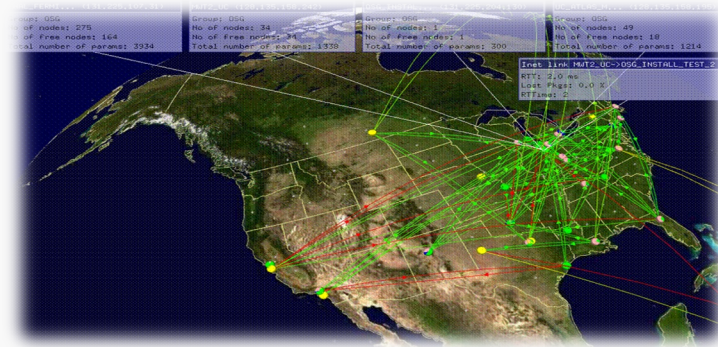
Sistem calcul distribuit





Resurse fizice

- Procesor – multi-core – 48 core-uri
(64 la AMD)
- Cluster
- Grid/Cloud





Supercomputers (top500.org)

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	LUMI - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	Leonardo - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610
5	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096



Frontier





Fugaku





Lumi





Leonardo





Summit





BOINC computing power

Totals

24-hour average: 13.790 PetaFLOPS.

Active: 43,637 volunteers, 159,606 computers.

Daily change: +28 volunteers, +1001 computers.



Tehnologiile pe care le vom folosi



Open MPI:
Open Source High Performance Computing



kubernetes



MPI

Framework care facilitează

- Pornirea programelor distribuite (processe pe același sistem sau pe sisteme diferite, dar strâns conectate – ideal aceeași rețea)
- Conectarea proceselor unui program distribuit (accept, bind, connect)
- Simplificarea identificării (identificatori în loc de IP, port)
- Simplificarea comunicării (oferă funcții gen Send/Recv, Broadcast)
- Asigură comunicarea corectă pe sisteme cu arhitecturi de calcul diferite (little/big endian problems)