Paralelné programovanie

Bc. št. prog. Informatika - 2010/2011

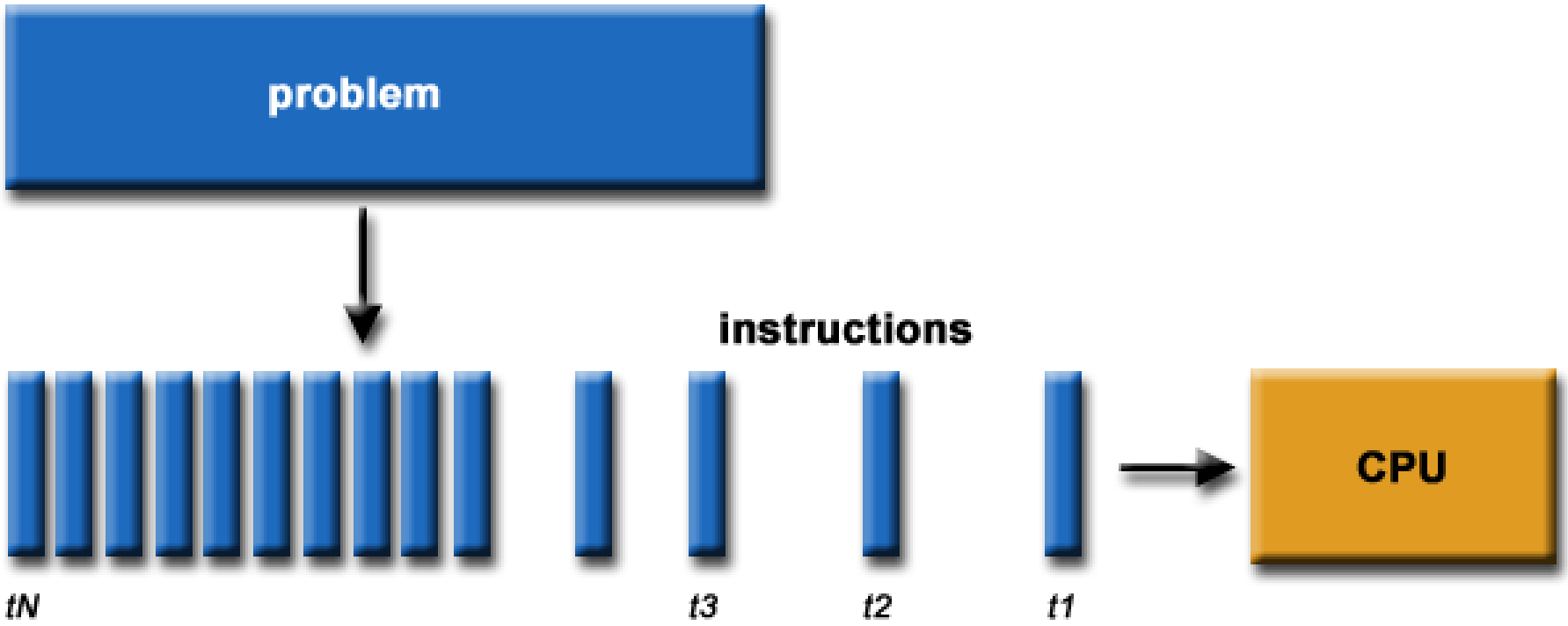
Ing. Michal Čerňanský, PhD.

Fakulta informatiky a
informačných technológií,
STU Bratislava

Paralelné počítanie (Parallel Computing)

- Bežný softvér - sekvenčné sériové spracovanie
- Vykonávané na jednom počítači s jedinou CPU
- Problém je dekomponovaný na menšie celky, až na inštrukcie
- Inštrukcie vykonávané sériovo, jedna po druhej
- Iba jediná inštrukcia je vykonávaná v danom časovom okamžiku

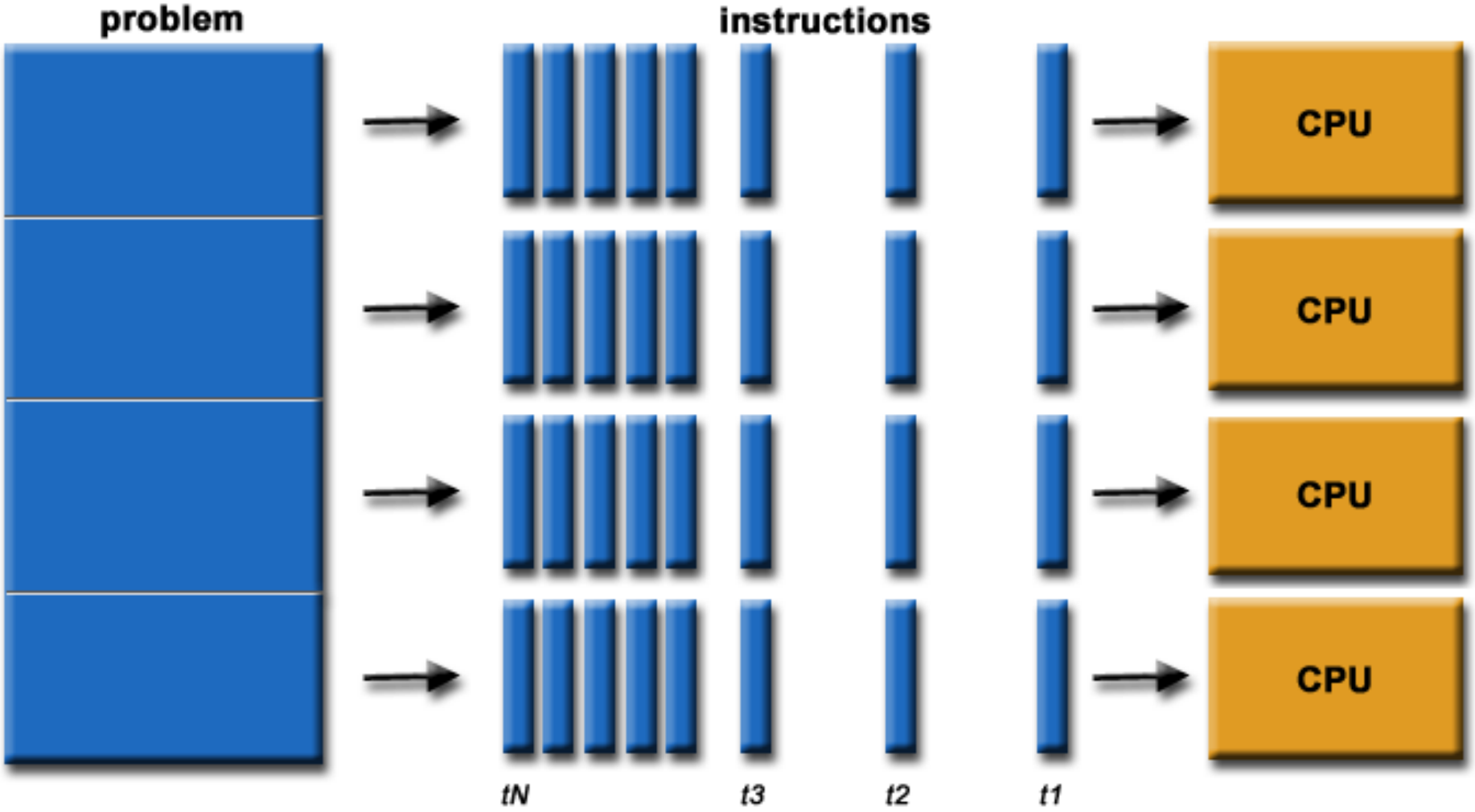
Paralelné počítanie



Paralelné počítanie

- Paralelné počítanie – využívanie viacerých výpočtových zdrojov na vyriešenie problému
- Vykonávané na viacerých CPU
- Problém je dekomponovaný na menšie nezávislé celky ktoré môžu byť vykonávané súbežne
- Celky dekomponované až na inštrukcie
- Inštrukcie z každej časti môžu byť vykonávané súčasne na viacerých CPU

Paralelné počítanie



Paralelné počítanie

- Výpočtový zdroj:
  - Jediný počítač s jediným procesorom s viacerými jadrami
  - Jediný počítač s viacerými procesormi
  - Viaceré počítače prepojené prostredníctvom komunikačnej siete
  - Kombinácia uvedených technológií

Paralelné počítanie

- Charakeristiky problému:
  - Možnosť jeho rozdelenia na diskrétne celky, ktoré je možné vykonávať súbežne
  - Možnosť vykonať viaceré programové inštrukcie v každom časovom okamžiku
  - Môže byť vyriešený s viacerými výpočtovými zdrojmi v kratšom čase ako s jediným výpočtovým zdrojom

Motivácia

- Vysoký výpočtový výkon – HPC (High Perf. Comp.)
  - Vedecké modelovanie a simulácia – vytváranie a skúmanie výpočtových numerických počítačových modelov - všetky vedné disciplíny
- „Bežné počítanie“
  - Spracovanie rozsiahlych dát - obrazu, zvuku
  - Náročný numerický výpočet - šifrovanie
  - „Responsivness“ aplikácie
    - Používateľské rozhranie
    - Vstupno-výstupné operácie

Motivácia

- Zložité vedecké a inžinierske problémy:
  - Bioinformatika, bioveda, biotechnika, genetika
  - Klimatické modelovanie, modelovanie atmosféry, hydrologické modelovanie, modelovanie zeme, modelovanie prostredia
  - Výpočtová chémia, molekulárne vedy
  - Výpočtová fyzika, aplikovaná, nukleárna, časticová fyzika, ...
  - Geológia, Sezmické modelovanie
  - Elektrotechnika, návrh integrovaných obvodov, mikroelektrotechnika
  - Matematika, Informatika
  - Strojárstvo, mechanika, protetika až vesm. lode

Motivácia

- Komerčné problémy:
  - Dátové sklady, dolovanie v dátach
  - Ropný priemysel
  - Webové technológie, vyhľadávače, webové služby
  - Medicínske vizualizácie a diagnostika
  - Farmaceutický priemysel, vývoj liekov
  - Menežment, kolaborativne riešenie problémov, logistika
  - Finančné a ekonomické modelovanie
  - Pokročilá grafika, virtuálna realita, obohatená realita, herný priemysel
  - Multimediálne technológie, video

Motivácia

- „Bežné “problémy:
  - Computer Vision
  - Computer Graphics
  - Machine Learning
  - Video Encoding
  - Webové služby, „Cloud“ computing
  - Vyhľadávanie na webe
  - Spracovanie textu

Motivácia

- Prečo paralelné počítanie ?
  - Ušetriť čas a peniaze – viac zdrojov, kratší výpočtový čas, šetrenie nákladov
  - Riešenie „veľkých problémov“ - Nemožnosť ich riešenia na jedinom počítači, pamäťová limitácia, veľký počet operácií transakcií v danom čase (PetaFLOPs, PetaBytes)
  - Použitie vzdialených zdrojov (SETI, Folding Home)
  - Limitácie sekvenčného počítania

top500.org

- Top500.org – usporiadaný zoznam superpočítačov
- 1. a 2. miesto jún 2010

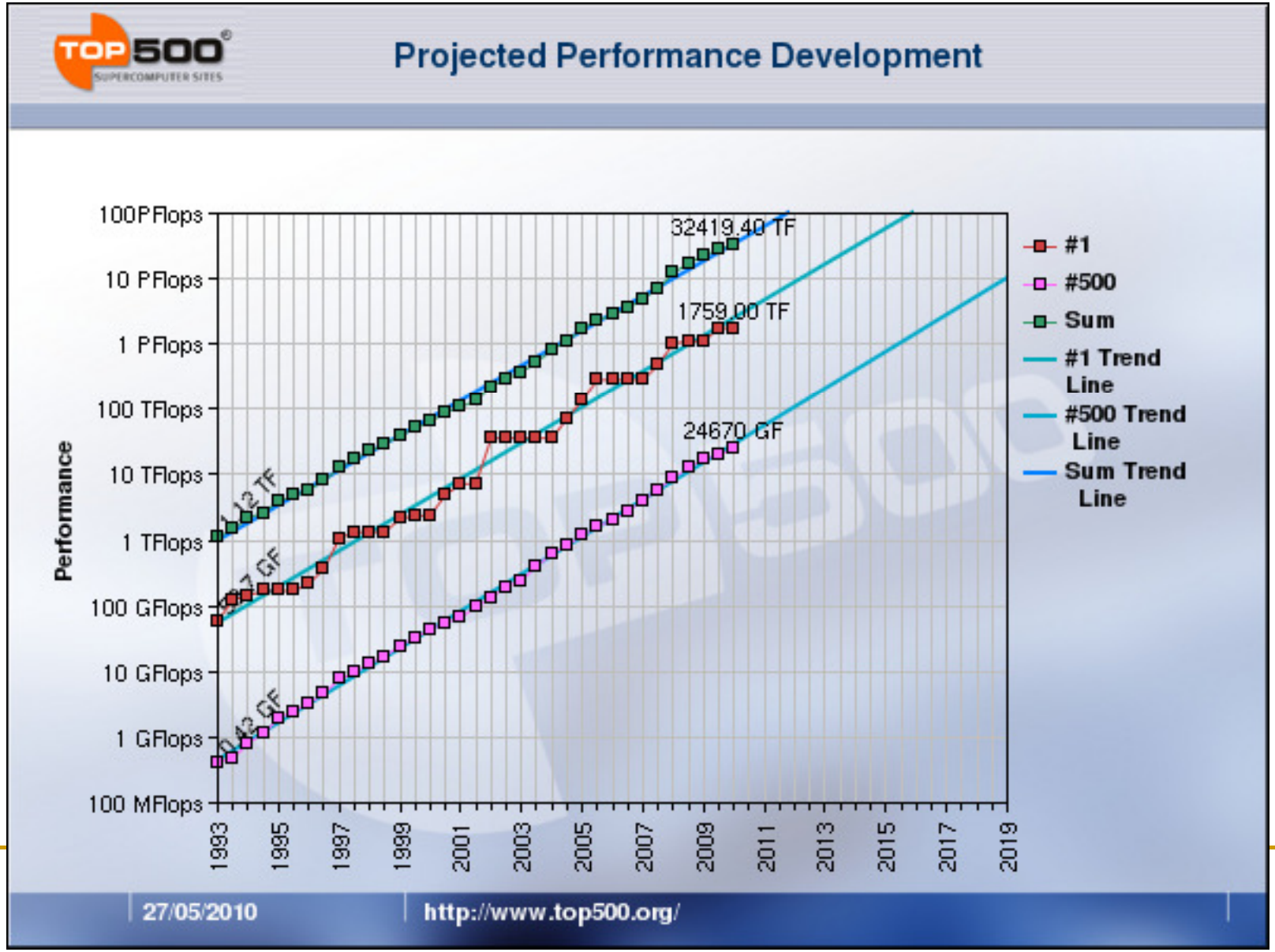
  Jaguar - Cray XT5-HE Opteron Six Core 2.6 GHz – 224162 jadier, 1.759e+06 GFlops

  Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU - 120640 jadier, 1.271e+06 GFlops

top500.org


TOP500
SUPERCOMPUTER SITES
Performance Development
100 PFlops
10 PFlops
1 PFlops
100 TFlops
10 TFlops
1 TFlops
100 GFlops
10 GFlops
1 GFlops
100 MFlops
Performance
32419.40 TF
1759.00 TF
24670 GF
1.12 TF
59.7 GF
0.42 GF
#1
#500
Sum
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
27/05/2010
http://www.top500.org/

top500.org

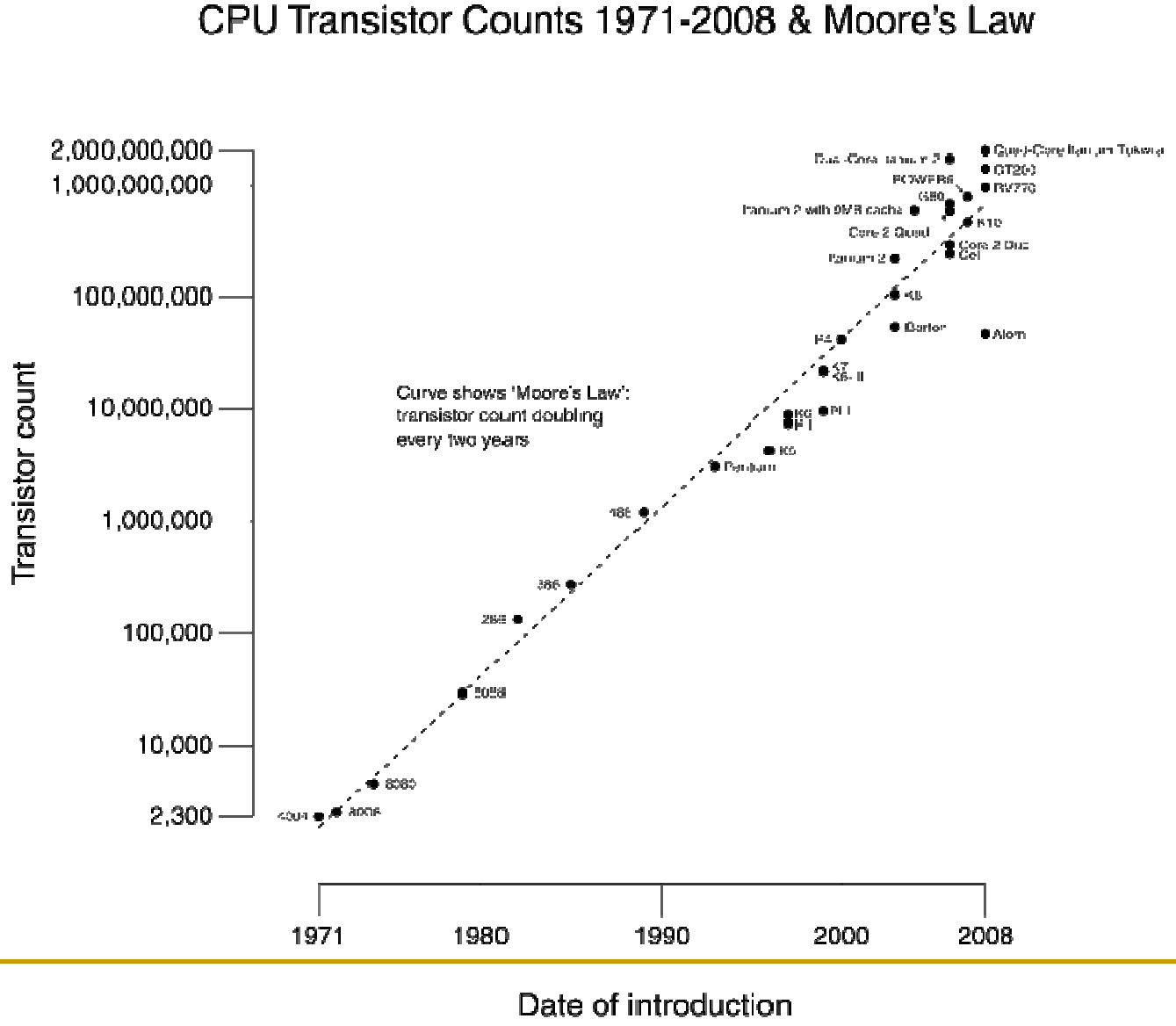

TOP500
SUPERCOMPUTER SITES
Projected Performance Development
Performance
100 PFlops
10 PFlops
1 PFlops
100 TFlops
10 TFlops
1 TFlops
100 GFlops
10 GFlops
1 GFlops
100 MFlops
1993
1995
1997
1999
2001
2003
2005
2007
2009
2011
2013
2015
2017
2019
32419.40 TF
1759.00 TF
24670 GF
1.12 TF
#1
#500
Sum
#1 Trend Line
#500 Trend Line
Sum Trend Line
27/05/2010
http://www.top500.org/

Moorov zákon

The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000.

Gordon Earle Moore (1929) - co-founder and Chairman Emeritus of Intel Corporation and the author of Moore's Law (published in an article 19 April 1965 in Electronics Magazine).

Moorov zákon



Moorov zákon

- Prečo? „exponential behavior of die sizes, finer minimum dimensions, and circuit and device cleverness”.

- V roku 1975 zrevidoval svoj zákon: „There is no room left to squeeze anything out by being clever. Going forward from here we have to depend on the two size factors - bigger dies and finer dimensions“.

- Zredukoval mieru rastu na dvojnásobok každých 18 mesiacov od r. 1975

Moorov zákon

- Hustota tranzistorov na chipe sa zvyšuje
- Rýchlosť ostáva
- Ako zvyšovať počet operácií za jednotku času?
- **Paralelizmus**
  - Implicitný paralelizmus
  - Explicitný paralelizmus
    - Narastá počet jadier
    - Paralelizmus je daný k dispozícii pre programátorov

Sekvenčné výpočty - limitácia

- Energetická limitácia
  - So zvyšujúcim sa výkonom je potrebné zlepšovať energetickú účinnosť
- Pamäťová limitácia
  - Veľmi pomalá pamäť v porovnaní s rýchlosťou vykonávania inštrukcií
- ILP limitácia
  - Ťažšie a drahé zlepšovanie paralelizmu na úrovni inštrukcií

Energetická limitácia (Power Wall)

- Výkonnosť čipu je limitovaná tepelným vyžarovaním, nie počtom tranzistorov
- Zvyšovanie výkonnosti si vyžaduje aj zlepšovanie energetickej účinnosti
- Tepelné vyžarovanie úmerné frekvencií
- Zvýšenie frekvencie 4000x za posledných 10 rokov
- Odvádzanie tepla sa blíži k fyzikálnym limitom
- Zvyšovanie frekvencie si vyžaduje obrovské a neúnosné náklady na chladenie

Limitácia paralelizmu na úrovni inštrukcí (ILP Wall)

- Vyššie frekvencie vyžadujú dlhšiu linku prúdového spracovania
- „Pipelineing“ sa už nedarí zlepšovať
- Zvyšovanie výkonu prostredníctvom špekulatívneho vykonávania inštrukcií
- HW testovanie vykonania inštrukcií v zlom poradí
- Potreba „uhádnuť“ skoky – zlý odhad = zahodenie výpočtu (skok každých 5 inštrukcií)
- Dátova závislosť môže zabrániť paralelnému vykonaniu inštrukcií
- Paralelizmus na úrovni inštrukcií (ILP -Instruction Level Parallelism)
- ILP logika – zložitá „CPU Execution Unit“, väčšia energetická náročnosť
- Čoraz menšie zlepšenia – ILP Wall

Pamäťová limitácia (Memory Wall)

- Pamäťové oneskorenia mnohorádovo väčšie v porovnaní s dĺžkou procesorového cyklu
- Výkonnosť programu limitovaná schopnosťou procesora preniesť dáta potrebné na výpočet
- Snaha zlepšiť rýchlosť prístupu k údajom a inštrukciám hierarchiou vyrovnávacích pamätí
- Drahé „cache miss“ – stratené CPU cykly

Pamäťová hierarchia

- Využívanie lokálnosti (locality) pri prístupe k údajom
  - Priestorová lokálnosť
  - Časová lokálnosť
- Pamäťová hierarchia využíva lokálnosť

- 1ns Procesor – On Chip Cache
- 10ns Second Level Cache
- 100ns Main Memory
- 10ms Disk Memory
- 10s Tape Memory

Oneskorenia v pamät'ovej hierarchií

Hierarchia	**Cykly CPU**
Register	1
L1 Cache	2-3
L2 Cache	6-12
L3 Cache	14-40
Near Memory	100-300
Far Memory	300-900
Remote Memory	O(10^3)
Message-passing	O(10^3)-O(10^4)

Limitácia - rýchlosť svetla

- 1 Tflop/s procesor

- 1 údaj vybraný za t = 1e10^-12
- Šírenie informácie v = 3e10^8 m/s
- „Vzdialenosť“ s = v . t = 0.33 mm

- Umiestnenie 1 Tb do 0.3 x 0.3 mm plochy znamená 1 atóm na 1 bit

Sekvenčné výpočty - limitácia

- Hustota tranzistorov na čipe sa zvyšuje
- Rýchlosť ostáva
- Zvýšenie výkonu je možne dosiahnúť iba nárastom počtu jadier
- Paralelizmus je daný k dispozícií pre programátorov

Náročnosť paralelného programovania

- Zložitejší návrh algoritmov
- Zložitejší vývoj softvéru
- Zmeny v hardvérových architektúrach

Princípy paralelného programovania

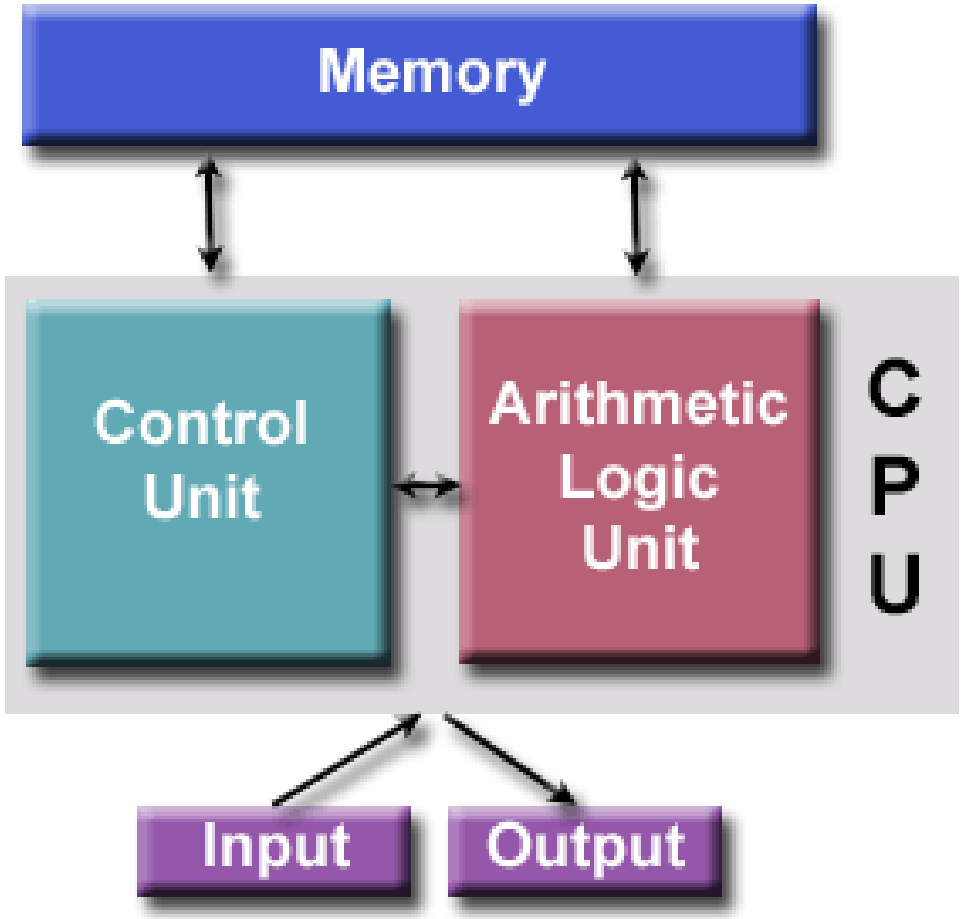
- Vyhľadanie paralelizmu
- Granularita paralelizmu
- Lokálnosť paralelizmu
- Vyrovnávanie záťaže
- Koordinácia a synchronizácia
- Analytické modelovanie výkonnosti

- Paralelné programovanie je principiálne náročnejšie ako sekvenčné programovanie

Architektúry počítačových systémov

- Von Neumannova architektúra
  - Pamäť
  - Riadiaca jednotka (CU)
  - Aritmeticko logická jednotka (ALU)
  - Vstupno výstupná jednotka

Architektúry počítačových systémov



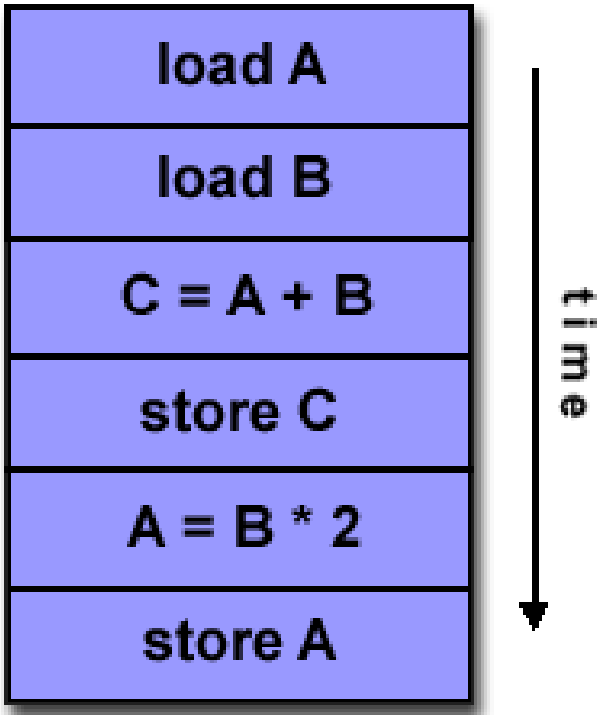
Architektúry počítačových systémov

- Flynnova taxonómia 1966
  - Inštrukcie a dáta
  - Single alebo Multiple
  - SISD – Single Instruction Single Data
  - SIMD – Single Instruction Multiple Data
  - MISD – Multiple Instruction Single Data
  - MIMD – Multiple Instruction Multiple Data

  - SPMD – Single Program Multiple Data

Architektúry počítačových systémov

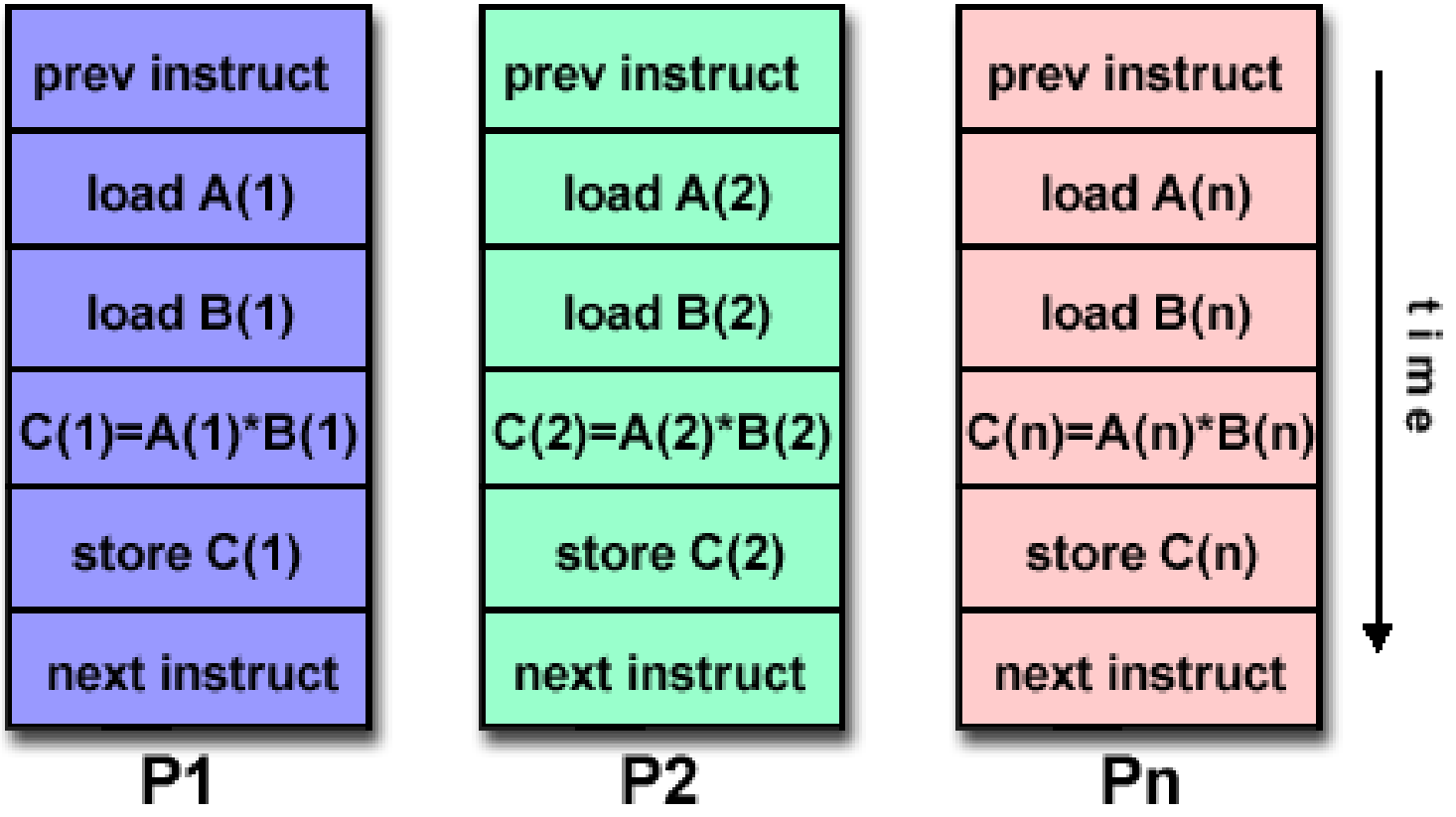
- SISD – Single Instruction Single Data
  - Klasický sekvenčný počítač
  - Jediný prúd inštrukcií v jednom cykle
  - Jediný prúd dát je používaný v jednom cykle
  - Deterministické spracovanie
  - Najbežnejší typ počítača



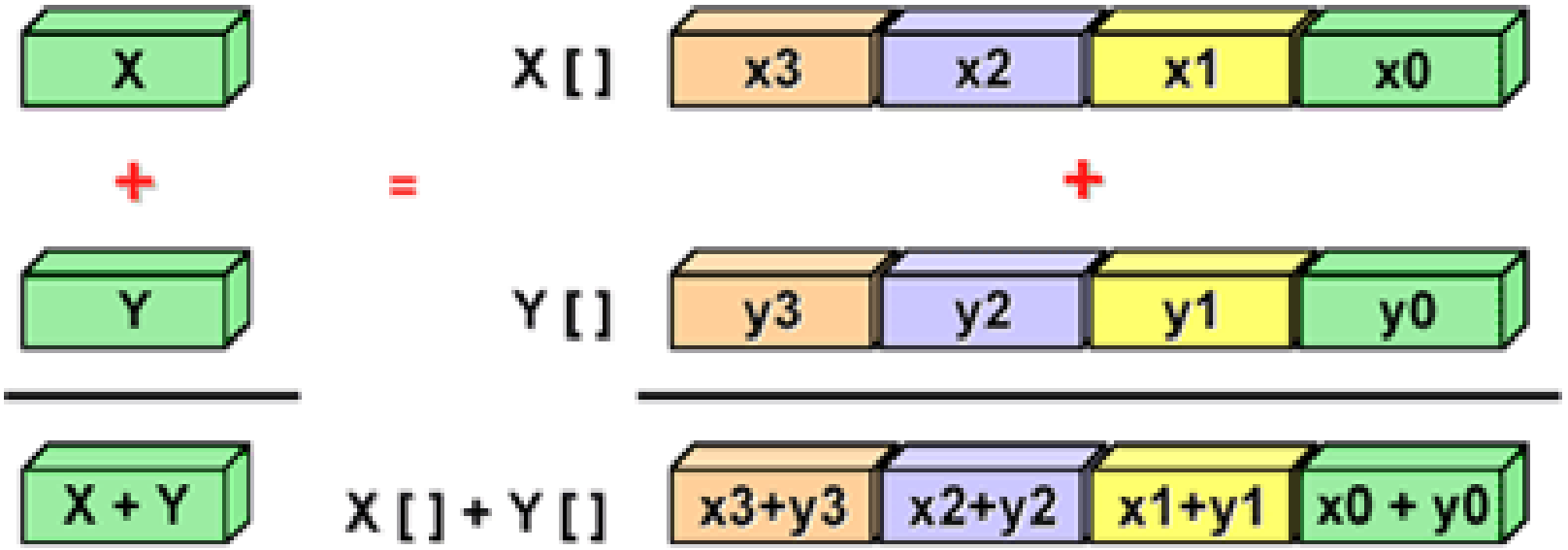
Architektúry počítačových systémov

- SIMD – Single Instruction Multiple Data
  - Paralelný počítač
  - Jediný prúd inštrukcií, každá výpočtová jednotka vykonáva tú istú inštrukciu v danom cykle
  - Viaceré prúdy dát, každá výpočtová jednotka pracuje s rôznymi dátami
  - Väčšina moderných počítačov
    - Vektorové inštrukcie
    - Grafické karty

Architektúry počítačových systémov



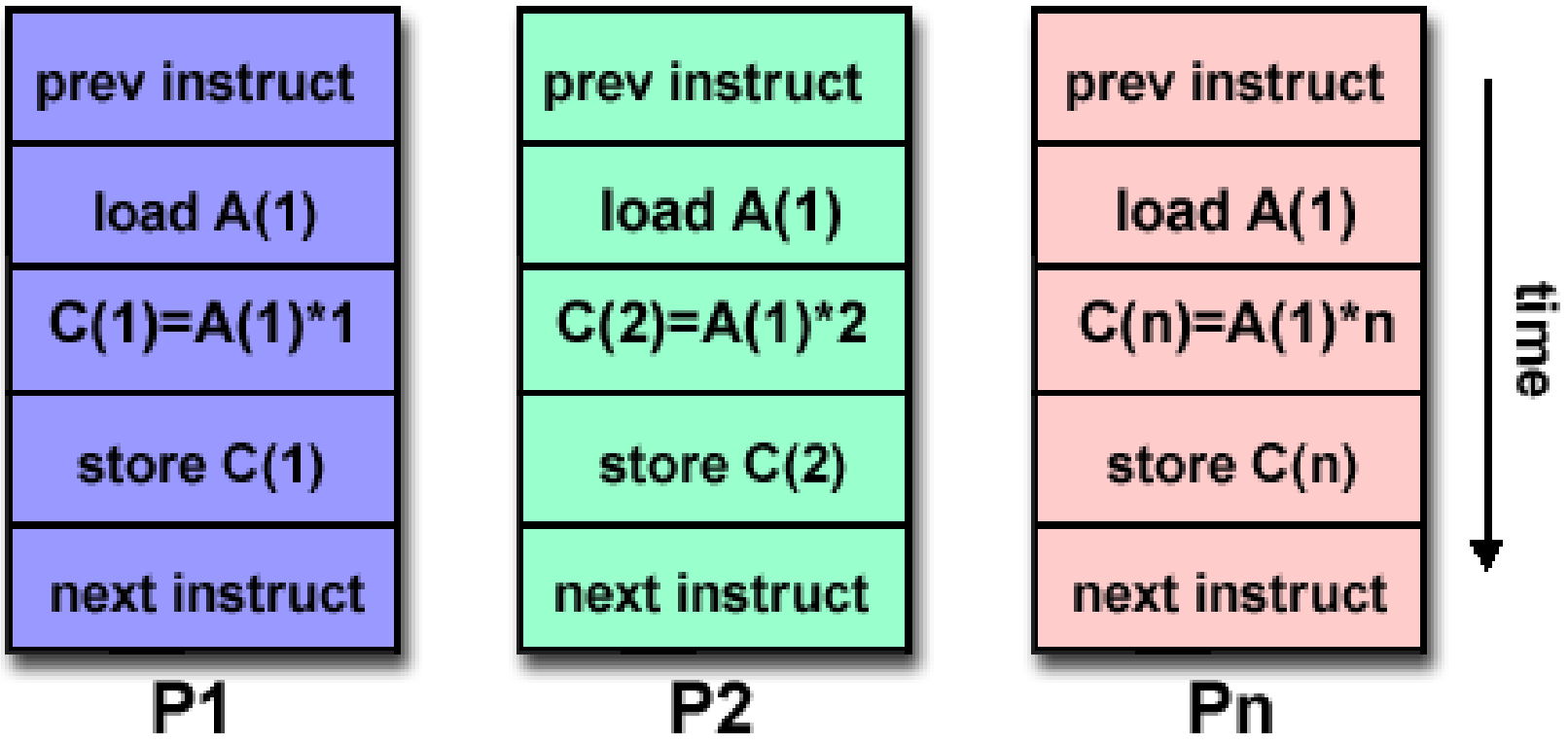
Architektúry počítačových systémov



Architektúry počítačových systémov

- MISD – Multiple Instruction Single Data
  - Paralelný počítač
  - Jediný prúd dát je spracovaný viacerými výpočtovými jednotkami
  - Viaceré prúdy inštrukcií, každý procesor vykonáva rôzne inštrukcie nad tými istými dátami
  - Veľmi zriedkavá architektúra
    - Filtre
    - Redundandné spracovanie

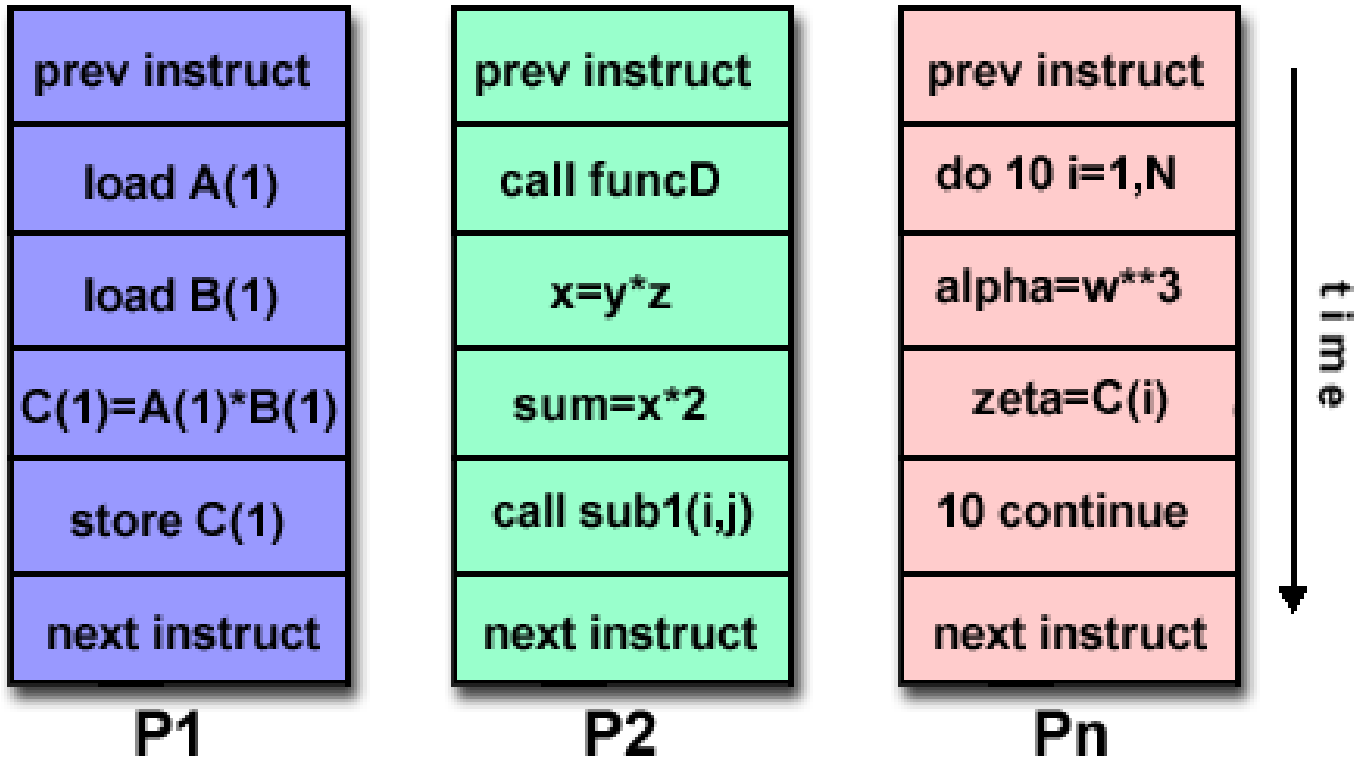
Architektúry počítačových systémov



Architektúry počítačových systémov

- MIMD – Multiple Instruction Multiple Data
  - Bežná architektúra paralelného počítača
  - Viaceré prúdy inštrukcií, každý procesor vykonáva rôzne inštrukcie
  - Viaceré dátové prúdy, každý procesor vykonáva inštrukcie nad rôznymi dátami
  - Veľmi častá architektúra
    - Klasické viacjadrové procesory
    - Počítačové klastre a gridy
    - Superpočítače
    - Moderné grafické procesory

Architektúry počítačových systémov



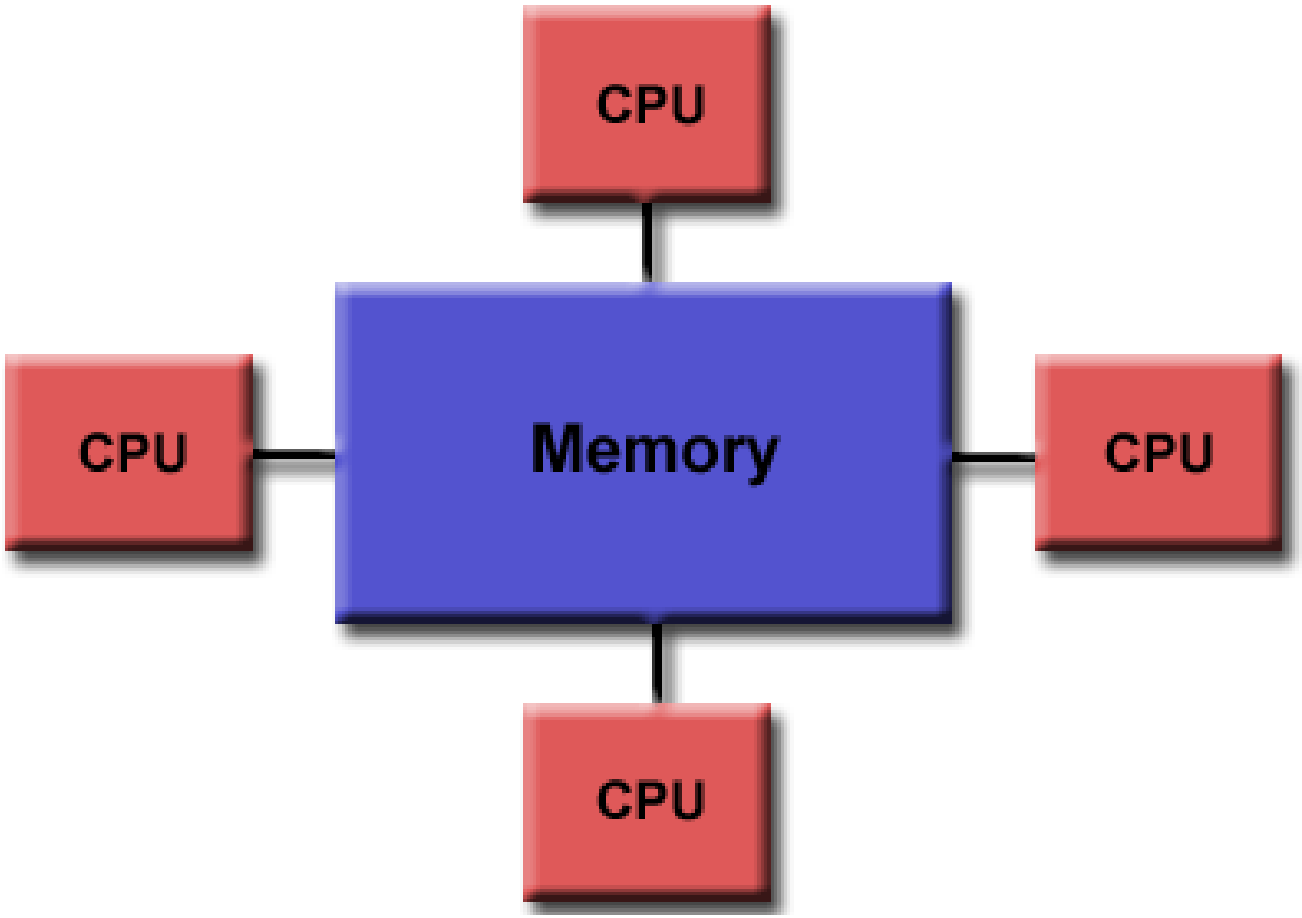
Pamäťové architektúry paralelných počítačových systémov

- Zdieľaná pamäť (Shared Memory)
  - Každý procesor má prístup do pamäte ktorá tvorí globálny pamäťový priestor
  - Každý procesor pracuje samostatne ale zdieľa tie isté pamäťové prostriedky
  - Zmena v pamäti vykonaná jedným procesorom je viditeľná pre všetky procesory
  - Čas prístupu do pamäte
    - Jednotný čas prístupu (UMA)
    - Nerovnomerný prístupový čas (NUMA)

Pamäťové architektúry paralelných počítačových systémov

- Jednotný čas prístupu – Uniform Memory Access (UMA)
  - Symetrické multiprocesory
  - Rovnaké procesory
  - Rovnaký čas prístupu do pamäte
  - CC-UMA – ak jeden procesor vykoná zápis do pamäte dozvedia sa to všetky procesory, koherencia na strane vyrovnávacej pamäte je riešená na HW úrovni

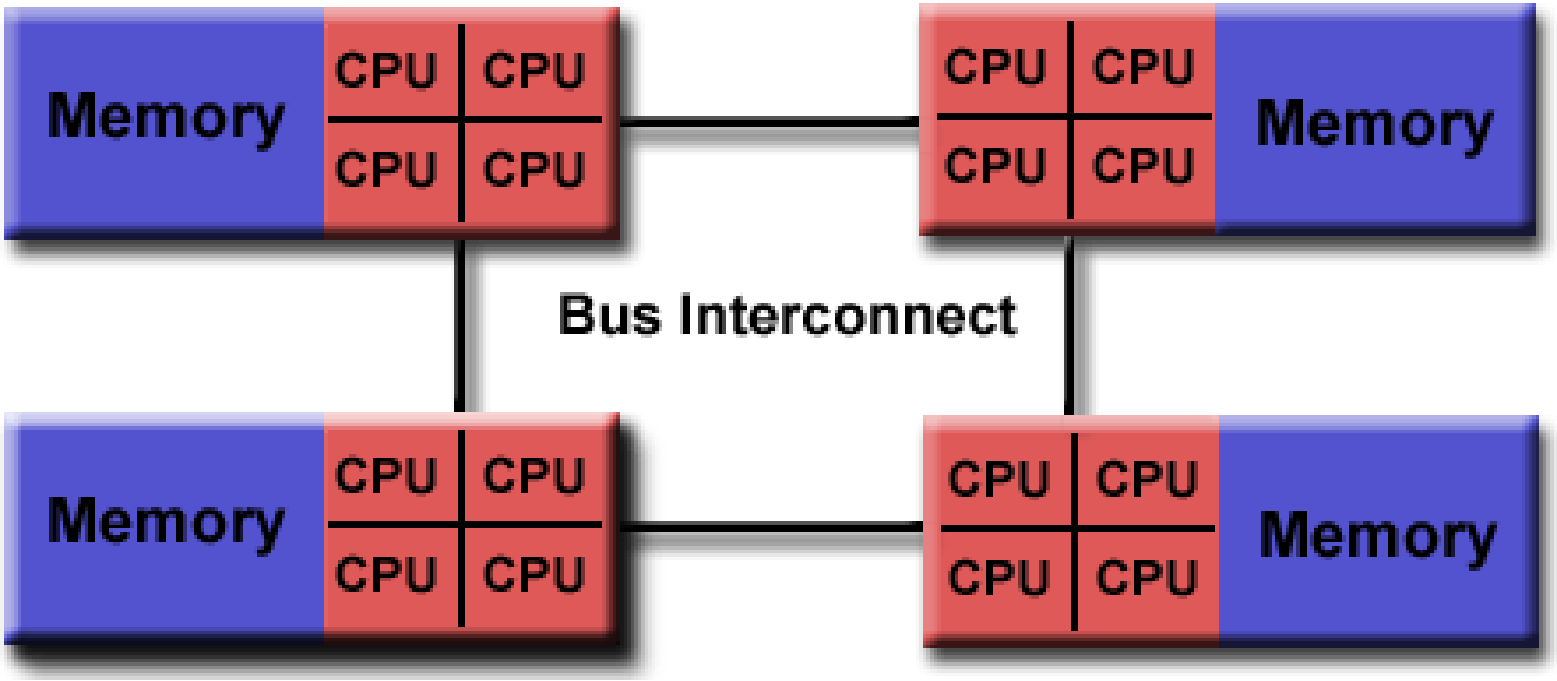
Pamäťové architektúry paralelných počítačových systémov



Pamäťové architektúry paralelných počítačových systémov

- Nerovnomerný čas prístupu – Non-Uniform Memory Access (NUMA)
  - Viaceré prepojené symetrické multiprocesory
  - Procesor má priamy prístup do pamäte ostatných procesorov
  - Čas prístupu do pamäte je rozdielny
  - CC-NUMA – ak sa zachováva koherencia vyrovnávacej pamäte

Pamäťové architektúry paralelných počítačových systémov



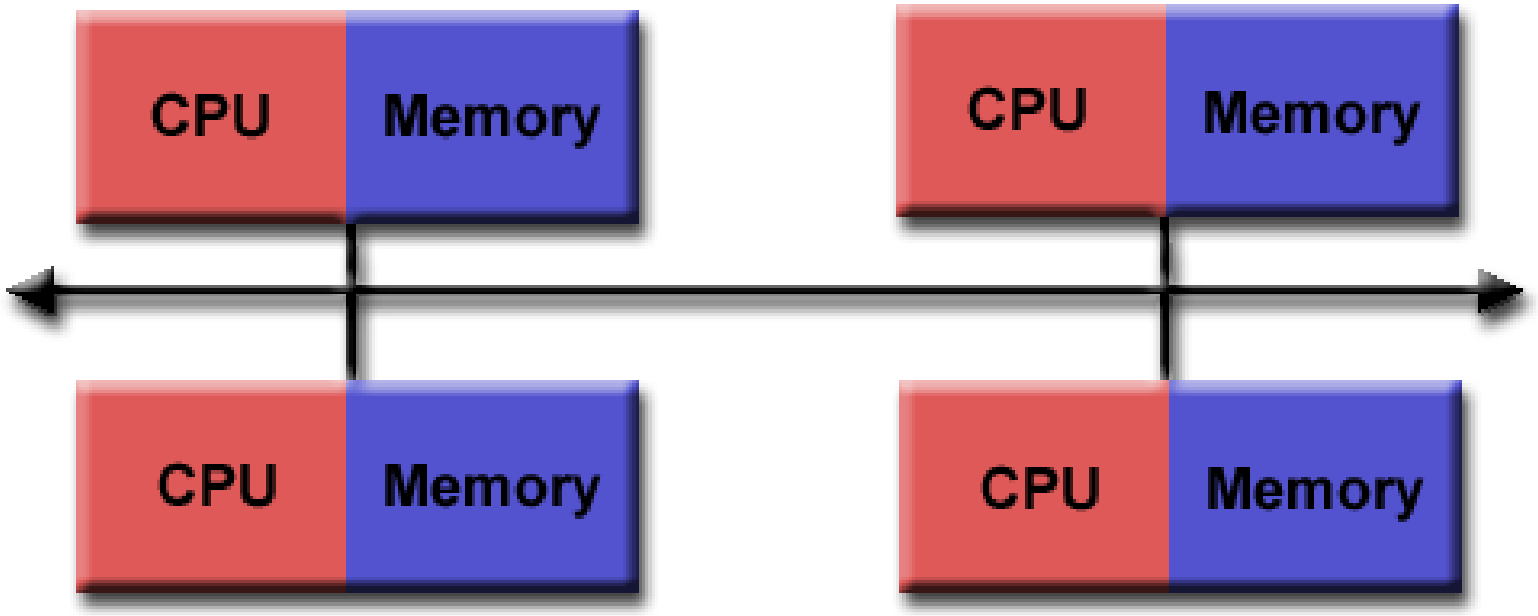
Pamäťové architektúry paralelných počítačových systémov

- Výhody systémov so zdieľanou pamäťou
  - Ľahko programovateľné
  - Zdieľanie dát je rýchle a rovnaké (blízkosť pamäte a procesorov)
- Nevýhody systémov so zdieľanou pamäťou
  - Slabá škálovateľnosť
  - Programátor musí zabezpečiť „správny“ prístup do pamäte

Pamäťové architektúry paralelných počítačových systémov

- Distribuovaná pamäť (Distributed Memory)
  - Komunikačná sieť pri prístupe do pamätí medzi procesormi
  - Každý procesor má prístup do svojej pamäte, ktorá sa nemapuje pre ostatné procesory do globálneho pamäťového priestoru
  - Každý procesor pracuje samostatne, zmena v lokálnej pamäti vykonaná jedným procesorom nie je viditeľná pre ostatné procesory
  - Prístup k údajom iného procesora je riešený na programátorskej úrovni, rovnako ako synchronizácia

Pamäťové architektúry paralelných počítačových systémov



Pamäťové architektúry paralelných počítačových systémov

- Výhody systémov s distribuovanou pamäťou
  - Dobrá škálovateľnosť
  - Rýchly prístup do lokálnej pamäte, nie je potreba zabezpečovať koherenciu vyrovnávacích pamätí
  - Ekonomická výhodnosť
- Nevýhody systémov s distrib. pamäťou
  - Programátor musí zabezpečiť mnohé detaily spojené s komunikáciou medzi procesmi
  - Ťažkosti spojené s mapovaním údajov do distribuovaného pamäťového priestoru
  - Rôzne časy prístupu k údajom

Pamäťové architektúry paralelných počítačových systémov

- Hybridné architektúry
  - Najväčšie a najrýchlejšie superpočítače
  - Súčasné počítačové klastre
  - Systémy so zdieľanou pamäťou – CC SMP
  - Distribuovaná pamäť – sieť viacerých SMP, potreba sieťovej komunikácie

Paralelné programátorské modely

- Vlákna (Threads)
- Zasielanie správ (Message Passing)
- Dátovo paralelné modely
- Hybridné

- Paralelné programátorské modely – abstrakcia nad HW a pamäťovými architektúrami

Paralelné programátorské modely

- Model vlákien
  - Proces môže mať viaceré nezávislé cesty vykonávania inštrukcií
  - Systémy so zdieľanou pamäťou
  - Implementované ako knižnica podprogramov a/alebo direktívy pre kompilátor
  - Množstvo implementácií
    - Pthreads
    - OpenMP

Paralelné programátorské modely

- Model zasielania správ
  - Skupina úloh používa každá svoju lokálnu pamäť pri výpočte
  - Úlohy si vymieňajú údaje zasielaním a prijímaním správ
  - Zvyčajne sa vyžaduje kooperácia
  - Implementované ako knižnica podprogramov
  - MPI - Message Passing Interface

Paralelné programátorské modely

- Dátovo paralelné modely
  - Paralelné spracovanie sa zameriava na vykonávanie operácií nad dátovou množinou
  - Skupina úloh pracuje kolektívne nad rovnakou dátovou množinou, každá na inej časti množiny
  - Úlohy spravidla vykonávajú rovnaké operácie nad svojou časťou dátovej množiny
  - Implementácie – jazyky Fortran 90, 95, HPF

Paralelné programátorské modely

- Hybridné modely
  - OpenMP + MPI
  - Pthreads + MPI
  - HPF + MPI
- SPMD – Single Program Multiple Data
  - Vytvorený nad ostatnými modelmi
  - Úlohy nemusia vykonať kompletný program
- MPMD – Multiple Program Multiple Data
  - Vytvorený nad ostatnými modelmi
  - Viaceré rôzne programy , viaceré rôzne dáta

Návrh paralelných programov

- Automatická vs. Manuálna paralelizácia
- Identifikácia a implementácia paralelizmu – úloha programátora
- Náročný, iteratívny proces nachylný na chyby
- Automatická paralelizácia - „svätý Grál“
  - Plnoautomatická - kompilátor analyzuje kód, hlavne cykly
  - Riadená programátorom – direktívy pre kompilátor
  - Zlé výsledky

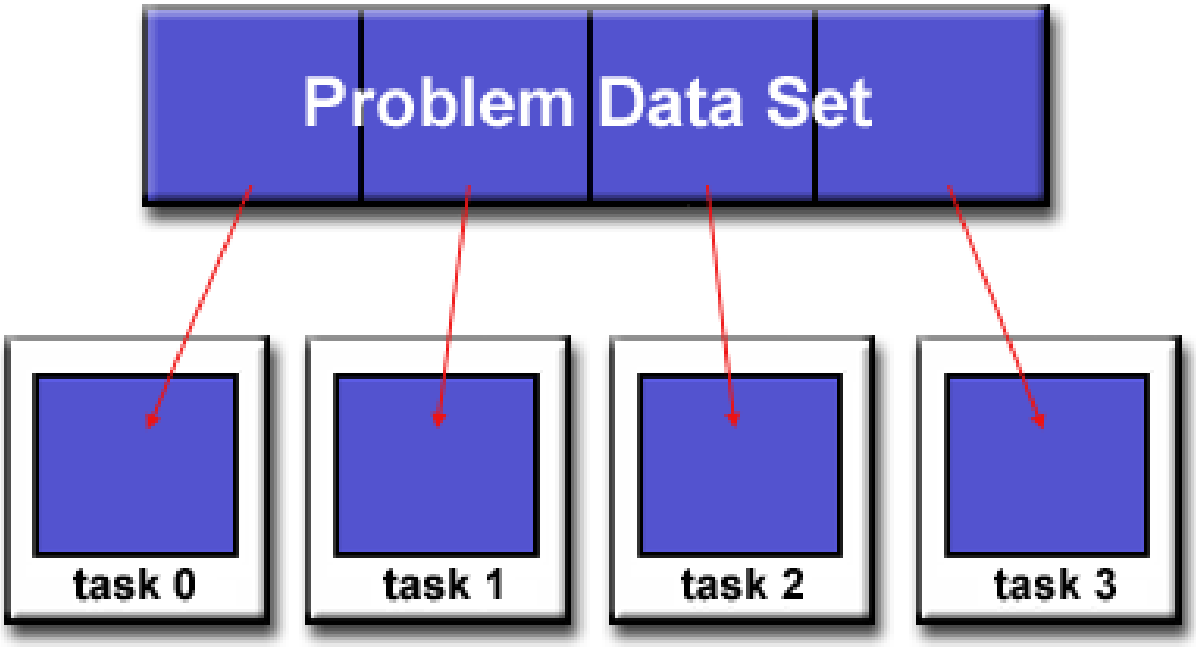
Návrh paralelných programov

- Pochopenie problému, posúdiť možnosti paralelizácia
- Spočítanie energie pre rôzne stavy molekuly
- Fibbonacci (1,1,2,3,5,8,13,21, ...)
- Identifikácia častí programu, kde sa vykonáva najviac „práce“ – profilovanie
- Identifikácia úzkeho hrdla – napr. IO
- Identifikácia inhibítorov pralelizmu (dátová závislosť)

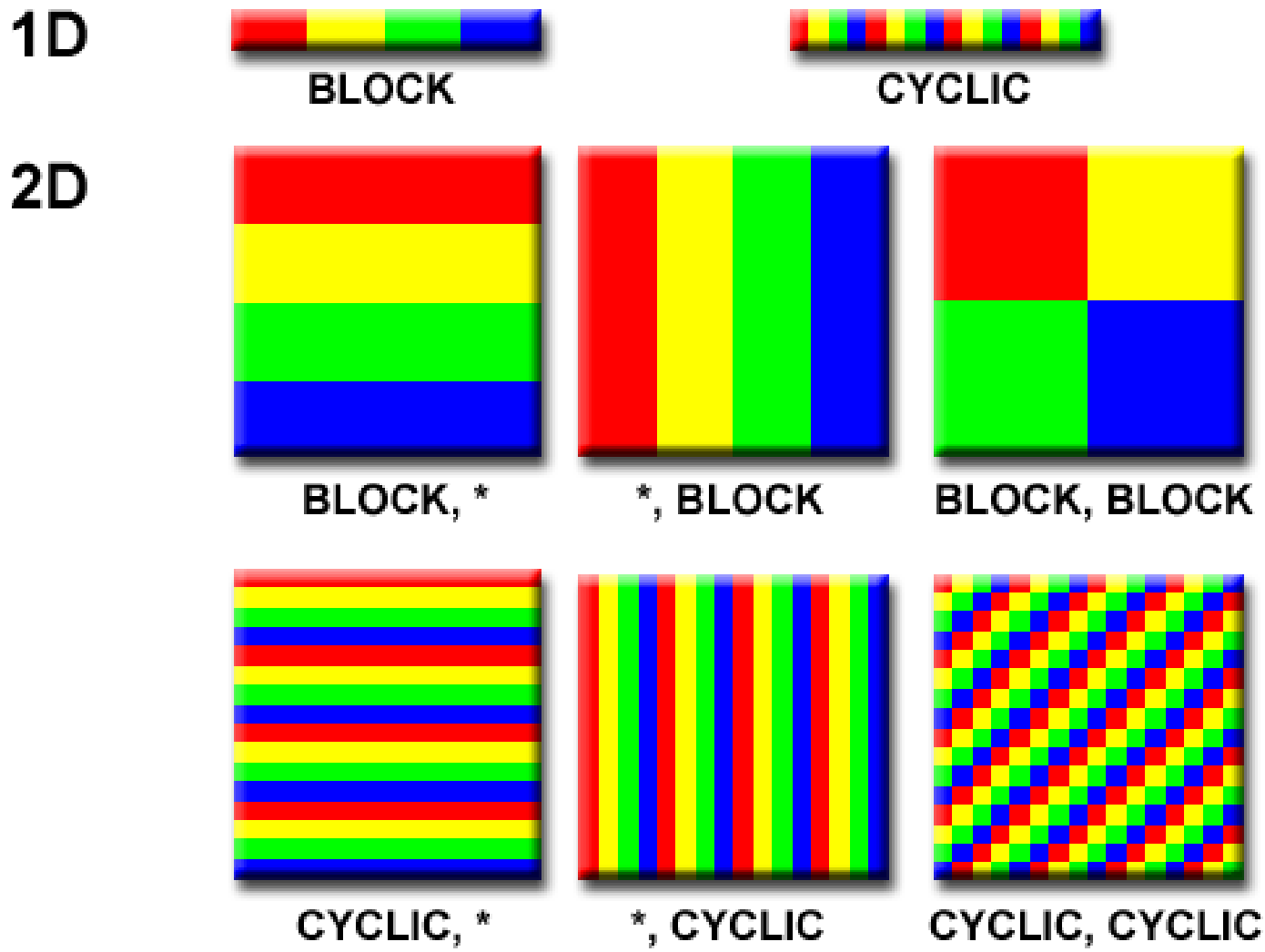
Návrh paralelných programov

- Dekompozícia problému (Partitioning)
  - Rozdelenie na diskrétne časti vykonateľné nezávisle
- Doménová dekompoizícia
  - Rozdelenie na základe dát, paralelné vykonanie úloh nad časťou celkových dát
- Funkcionálna dekompozícia
  - Rozdelenie na základe vykonávaných výpočtov, každá úloha vykonáva časť celkových potrebných výpočtov

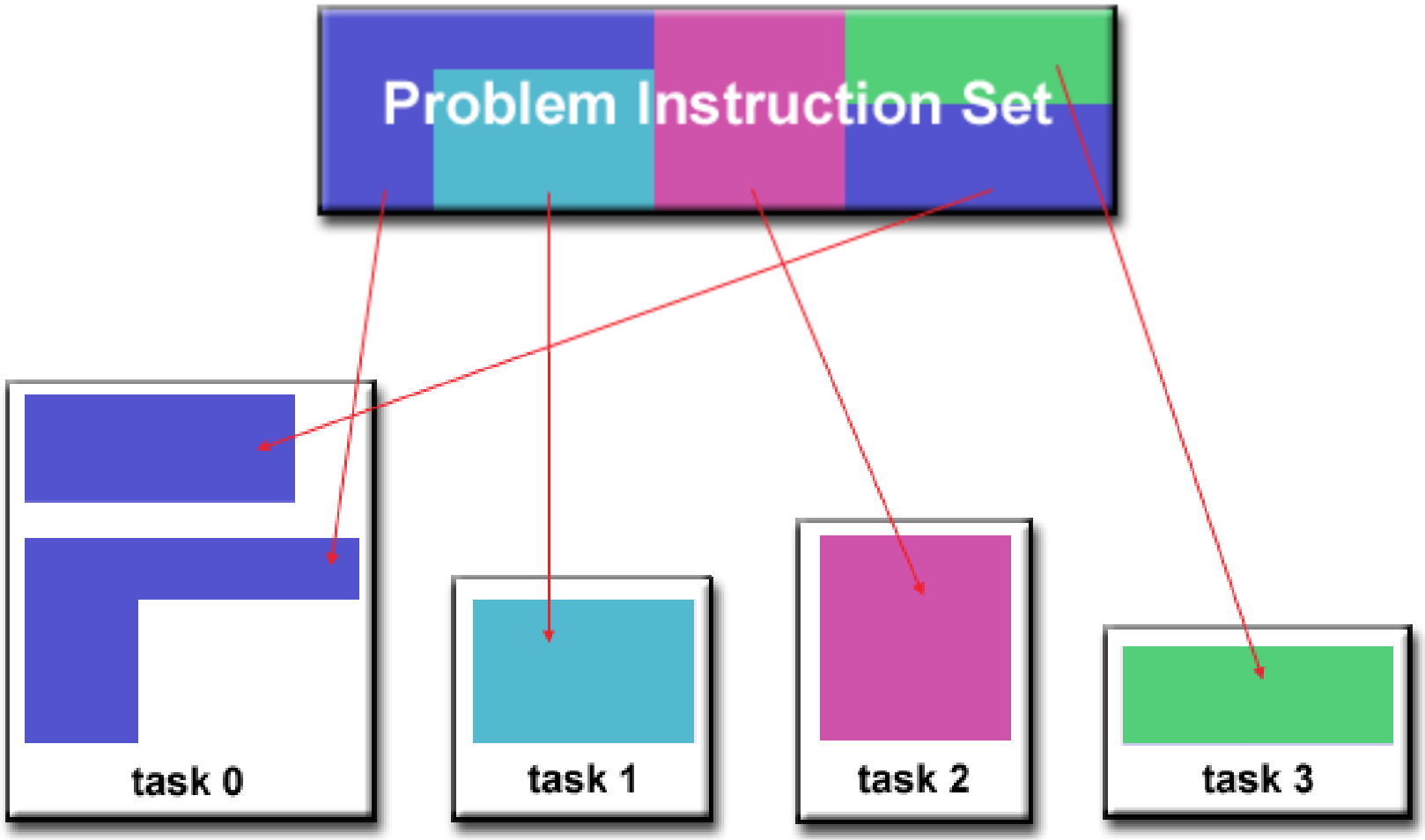
Návrh paralelných programov



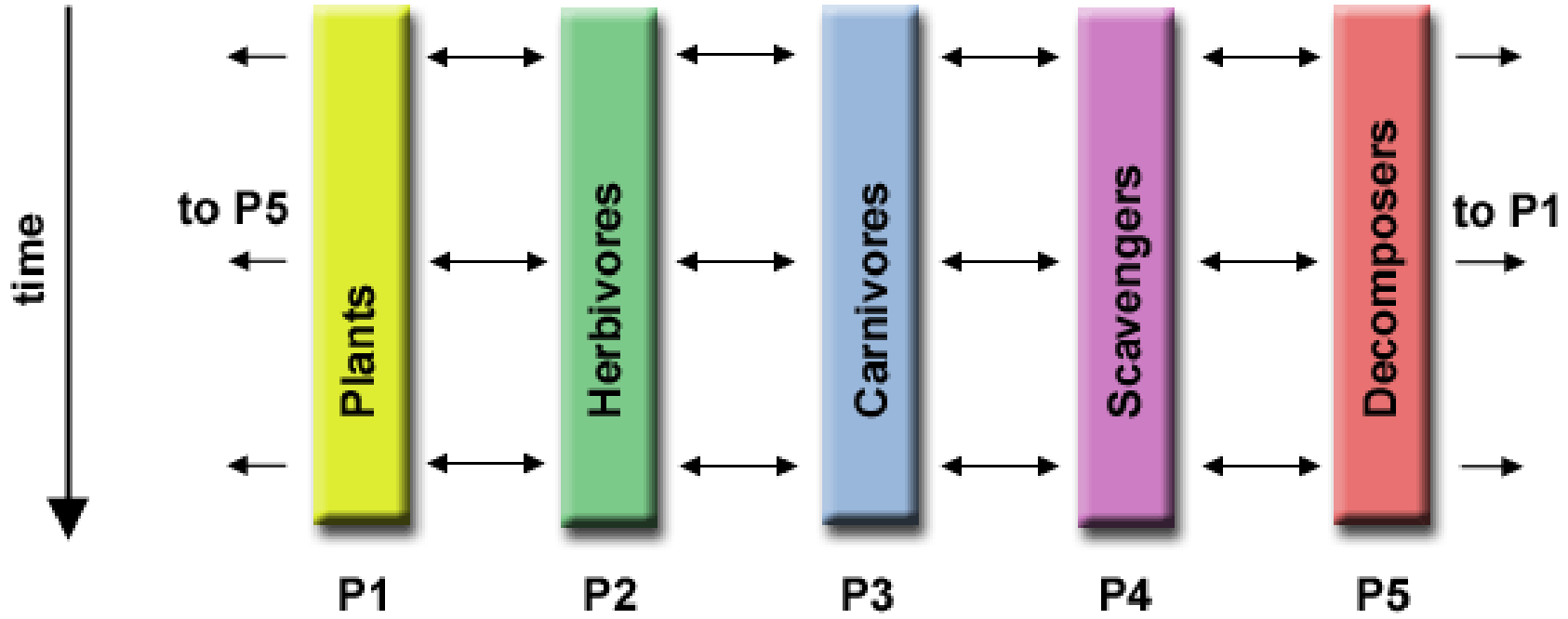
Návrh paralelných programov



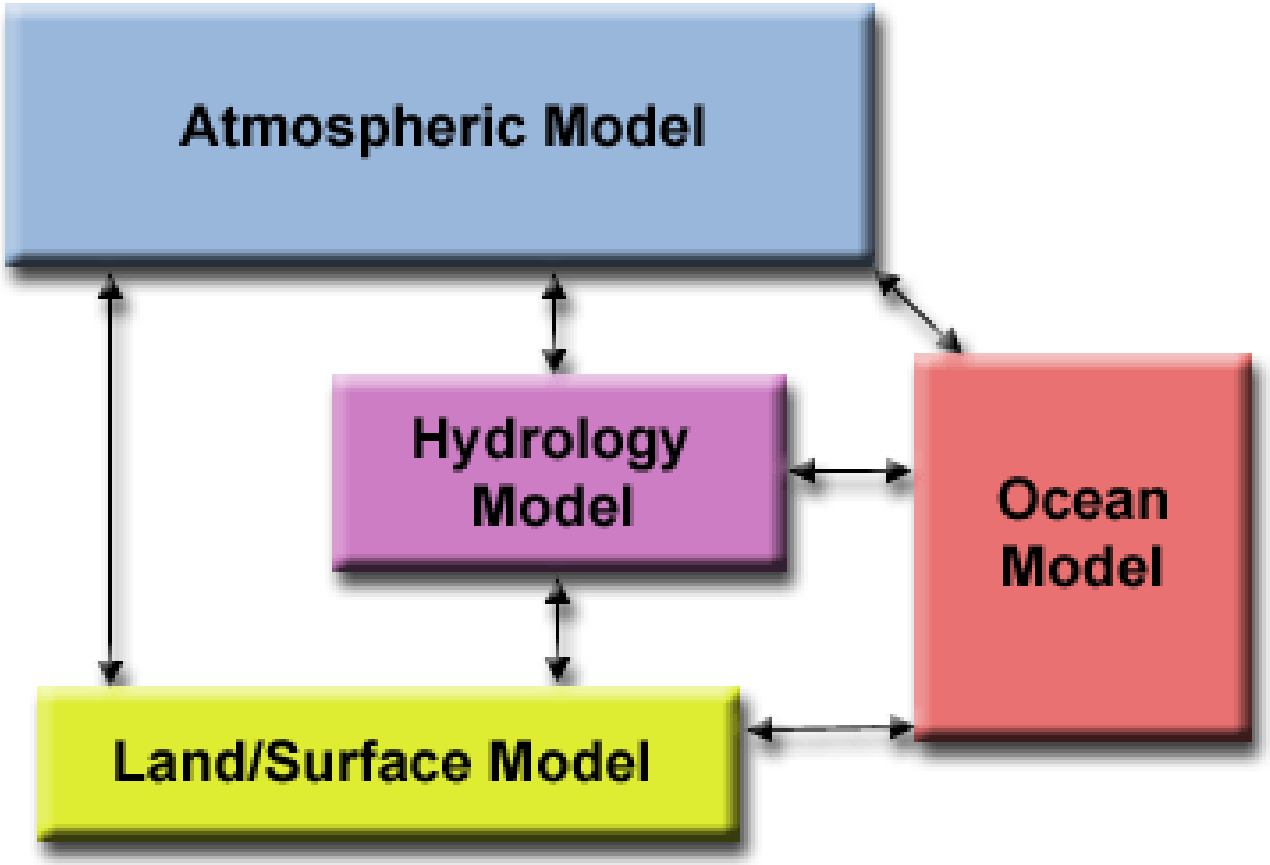
Návrh paralelných programov



Návrh paralelných programov



Návrh paralelných programov



Návrh paralelných programov

- Komunikácia
- Úlohy nepotrebujú komuniovať – „Embarrassingly Parallel Problems“ („Trápny“ resp. priamočiari paralelizmus)
- Väčšinou úlohy potrebujú komunikovať
  - Potreba zohľadniť viaceré faktory

Návrh paralelných programov

- Cena komunikácie
  - „Nadbytočnosť“ (Overhead) spojený s komunikáciou
  - Výpočtové cykly nepočítajú ale realizujú komunikáciu
  - Častá potreba synchronizácie – čakanie
  - Súťaženie pri komunikácií – saturácia dostupného prenosového pásma na sieti

Návrh paralelných programov

- Latencia vs. šírka prenosové pásma (Bandwidth)
  - Latencia – čas prenosu 0 bajtov
  - Šírka pásma – množstvo dát, ktoré môže byť prenesené za jednotku času
  - Veľa malých správ – latencia dominuje nad šírkou pásma v nadbytočnosti spôsobenou komunikáciou – malá efektívna šírka pásma
  - Spojenie malých správ do väčšej správy

Návrh paralelných programov

- Viditeľnosť komunikácie
- Zasielanie správ – komunikácia je viditeľná
- Dátovo paralelné modely – komunikácia je pred programátorom skrytá

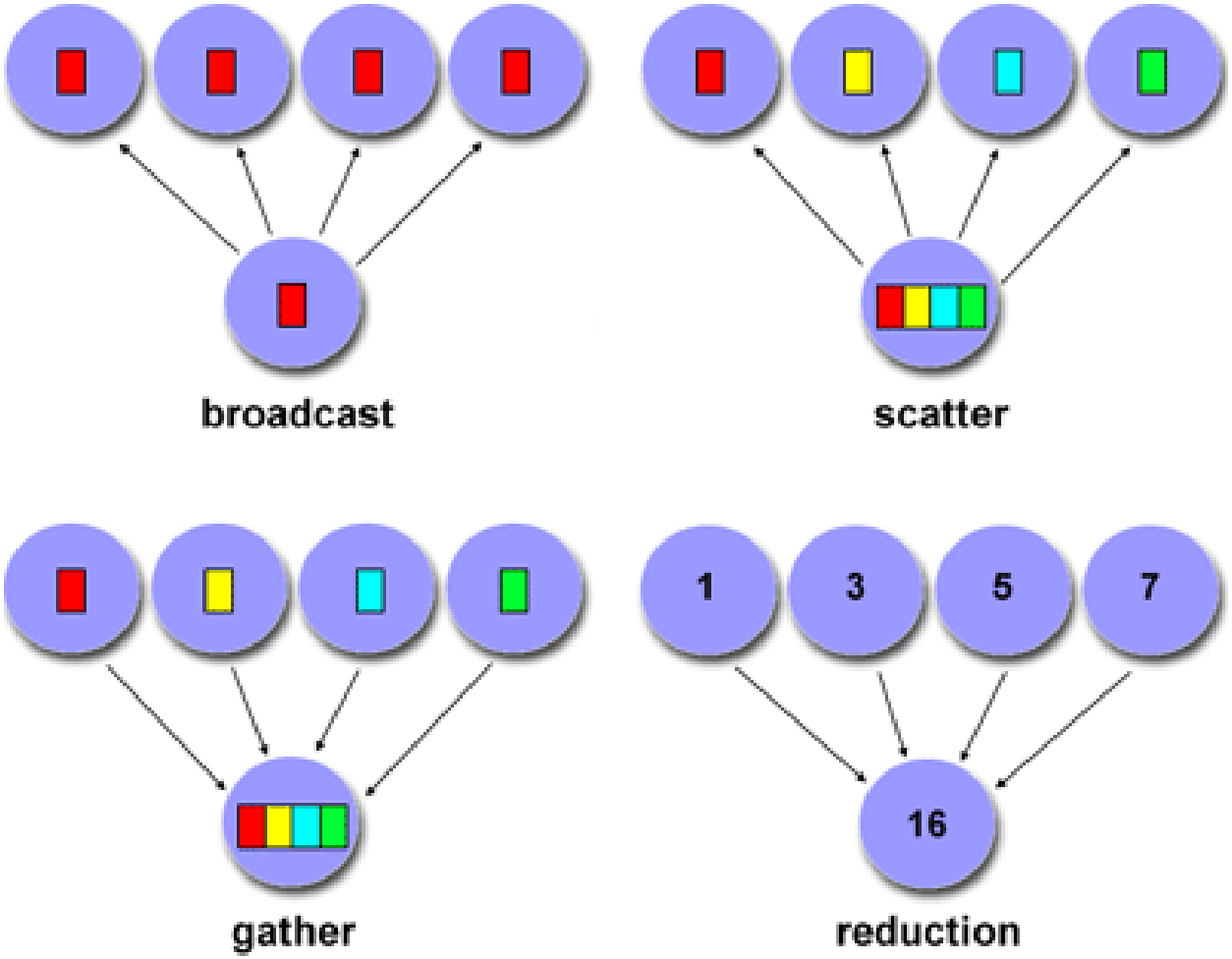
Návrh paralelných programov

- Synchrónna vs. asynchrónna komunikácia
- Synchrónna komunikácia – „handshaking“, blokujúca, čaká sa na ukončenie komunikácie
- Asynchrónna komunikácia – prenášanie dát medzi úlohami nezávisle – neblokujúca, možnosť prekladať výpočet s komunikáciou

Návrh paralelných programov

- Rozsah komunikácie
- Komunikácia bod – bod (Point-to-point)
  - Dve úlohy sa zúčastňujú komunikácie
  - Sender/producer -> Receiver/Consumer
- Skupinová komunikácia
  - Viac ako dve úlohy, často určené ako členy skupiny

Návrh paralelných programov



Návrh paralelných programov

- Efektívnosť komunikácie
- Správny výber konkrétneho paralelného programátorského modelu
- Správny výber typu komunikácie (synchrónna asynchrónna)
- Viaceré sieťové prepojenia – potreba vybrať vhodné

Návrh paralelných programov

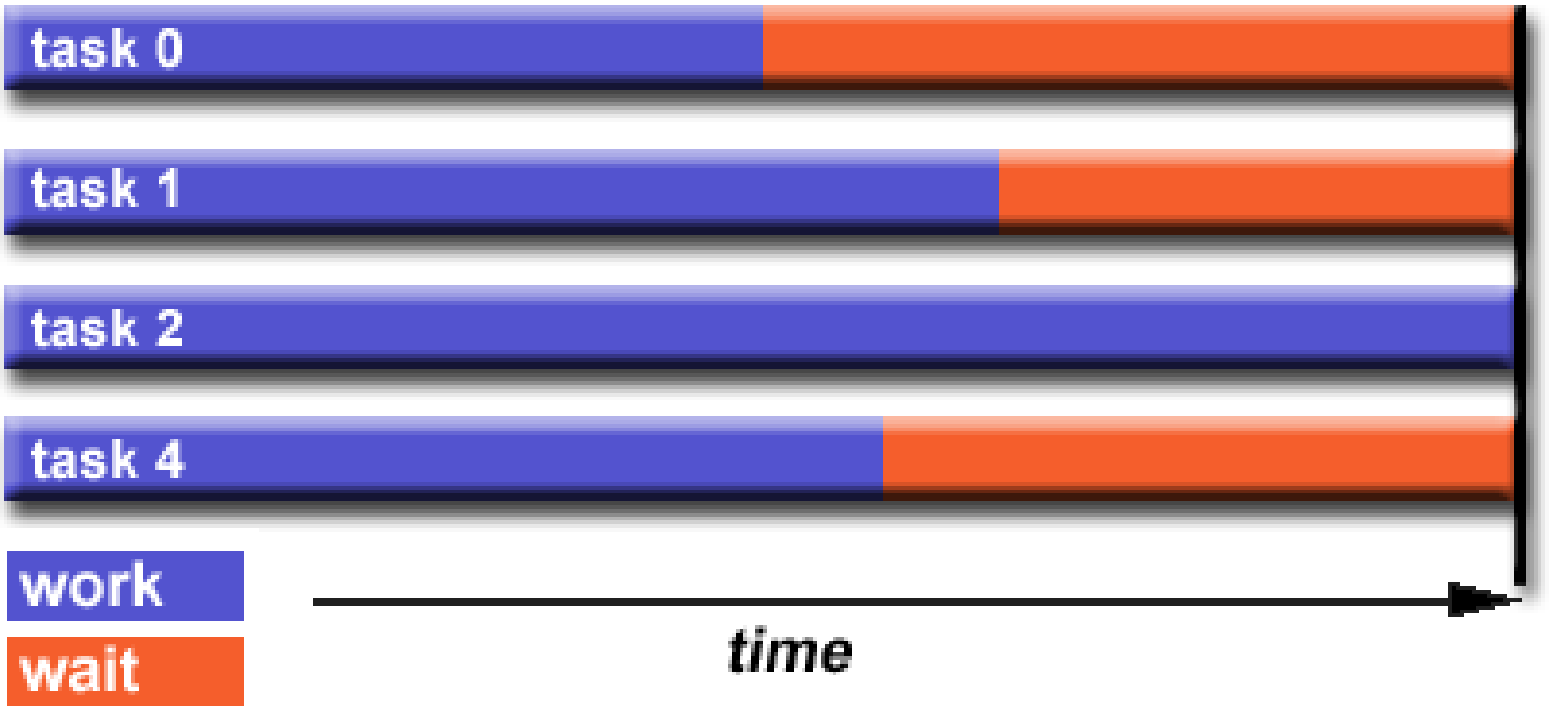
- Synchronizácia
- Barierová synchronizácia – spravidla všetky úlohy, všetky čakajú na poslednú úlohu
- Zámok, Semafór – týka sa viacerých úloh, serializácia prístupu k zdroju (často chránené dáta, alebo kód)
- Synchrónna komunikácia vyžaduje koordináciu

Návrh paralelných programov

- Dátová závislosť
- Poradie vykonania príkazov ovplyvní výsledok programu
- Viacnásobné použitie jednej dátovej lokácie
- Dátová závislosť v cykle
  - for (i=start;i<end;i++) a[i] = a[i-1] * 2;
- Dátová závislosť mimo cyklu
  - Task 1: x=2; .. y = 2*x;
  - Task 2: x=4; .. y = 3*x;
- Kumunikácia a synchrnonizácia

Návrh paralelných programov

- Vyrovnávanie záťaže (Load Balancing)
- Rovnomerné rozdeľovanie práce
- Všetky úlohy pracujú v každom čase
- Minimalizácia času nečinnosti (Idle time)
- Najpomalšia úloha môže určovať celkový výkon výpočtu pri barierovej synchronizácií


task 0
task 1
task 2
task 4
work
wait
time

Návrh paralelných programov

- Vyrovnávanie záťaže
- Rovnomerné rozdelenie práce pre jednotlivé úlohy
  - Polia, matice – rovnomerné rozdelenie prvkov, ak ich spracovanie trvá rovnaký čas
  - Cykly – rovnomerné rozdelenie iterácií
  - Heterogénne počítačové systémy – výkonnostná analýza a kompenzácia nerovnomernosti
- Dynamické pridelovanie práce

Návrh paralelných programov

- Dynamické prideľovanie práce
- Typické problémy
  - Riedke polia, matice
  - Adaptívne mriežkové metódy
  - Simulácie častíc, ktoré môžu migrovať
- Plánovač so zásobníkom úloh (Task pool)
  - Po skončení práce si úloha vyzdvihne ďalšiu prácu
- Potreba implementovať algoritmus vyhľadávania a riešenia nerovnom. záťaže

Návrh paralelných programov

- Granularita (Zrnitosť)
- Pomer veľkosti výpočtu a komunikácie
- Výpočet je spravidla popretkávaný komunikáciou
- Jemnozrnný paralelizmus (Fine-grain)
- Hrubozrnný paralelizmus (Coarse-grain)

Návrh paralelných programov

- Jemnozrnný paralelizmus (Fine-grain)
  - Malé výpočtové úseky medzi komunikáciou
  - Slabý pomer výpočet/komunikácia
  - Jednoduchšie vyrovnávanie záťaže
- Hrubozrnný paralelizmus (Coarse-grain)
  - Veľké výpočtové úseky medzi komunikáciou
  - Vysoký pomer výpočet/komunikácia
  - Ťažšie implementovateľné vyrovnávanie záťaže
- Vhodná úroveň granularity závisí od algoritmu a HW

Obmedzenia a náročnosť paralelného spracovania

- Amdahlov zákon
- Zložitejší celý cyklus vývoja SW
- Prenositeľnosť (Portability)
- Náročnosť na výpočtové prostriedky, zdroje
- Škálovateľnosť

Ahmdalov zákon

- Gene Amdahl 1967
- Zrýchlenie, ktoré je možné dosiahnuť paralelizáciou je limitované:

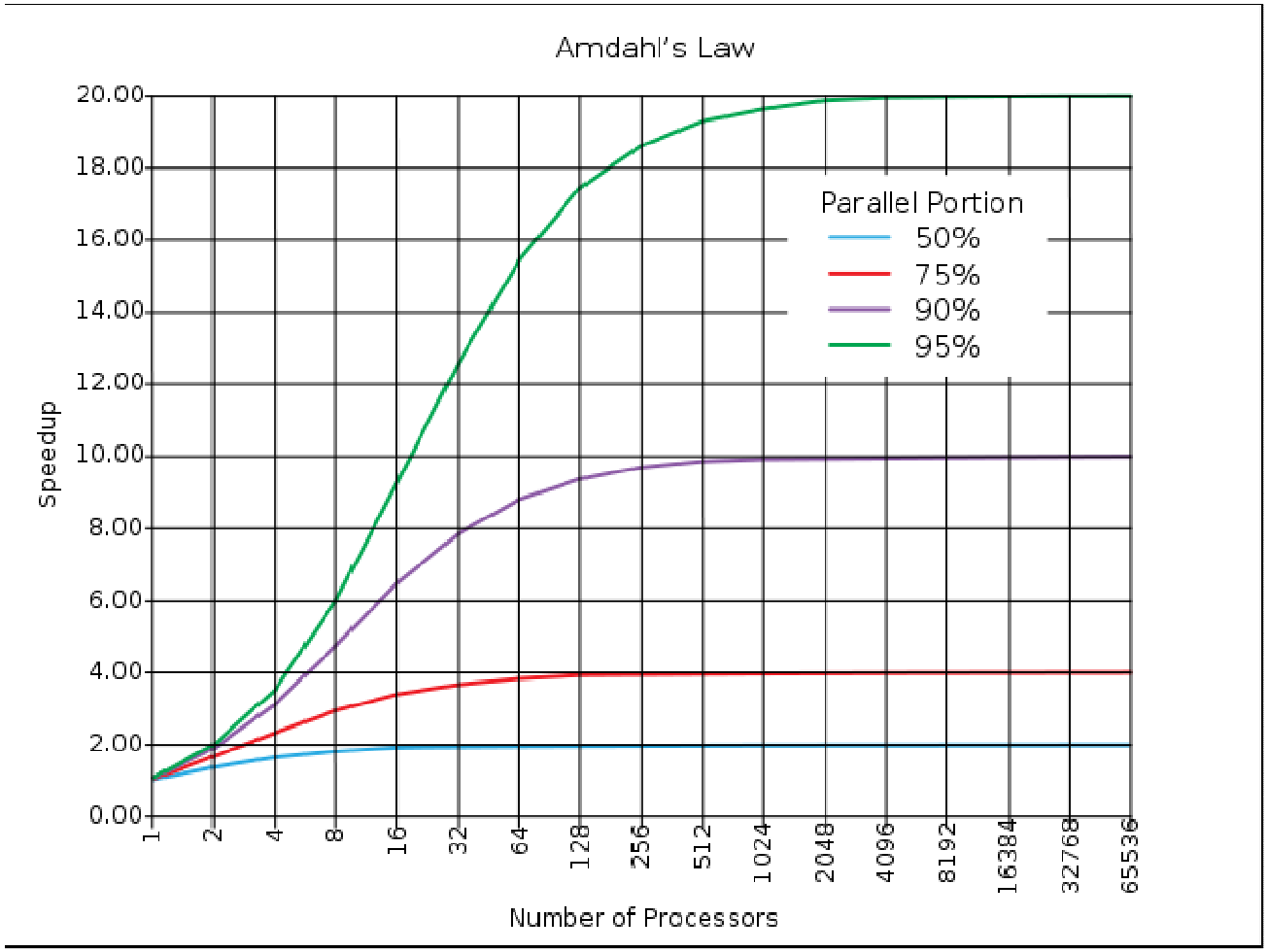
- V každom programe je sekvenčná časť, ktorá môže limitovať maximálne možné zrýchlenie paralelného programu

Ahmdalov zákon

- Fs – sekvenčná časť programu
- Fp – paralelná časť programu Fp = 1 - Fs
- N – počet procesorov
- Tn – čas behu programu na N procesoroch
- S – zrýchlenie programu na N procesoroch

- Tn = (Fp / N + Fs) T1
- S = 1 / (Fp / N + Fs)

Ahmdalov zákon



Obmedzenia a náročnosť paralelného spracovania

- Zložitejší celý cyklus vývoja SW
  - Návrh
  - Implementácia
  - Ladenie
    - Hľadanie chýb
    - Urýchľovanie
  - Správa

Obmedzenia a náročnosť paralelného spracovania

- Prenositeľnosť
  - Štandardizované API
  - Zaužívané paralelné programátorské modely
  - Rozdiely vo výkone konkrétnych implementácií
  - HW architektúry majú rozdielne vlastnosti, rýchly vývoj v oblasti
  - Dôležitosť OS

Obmedzenia a náročnosť paralelného spracovania

- Náročnosť na výpočtové prostriedky, zdroje
  - Väčší počet výpočtových elementov
  - Pamäťová náročnosť môže byť vyššia pre paralelné programy
  - Energetická náročnosť

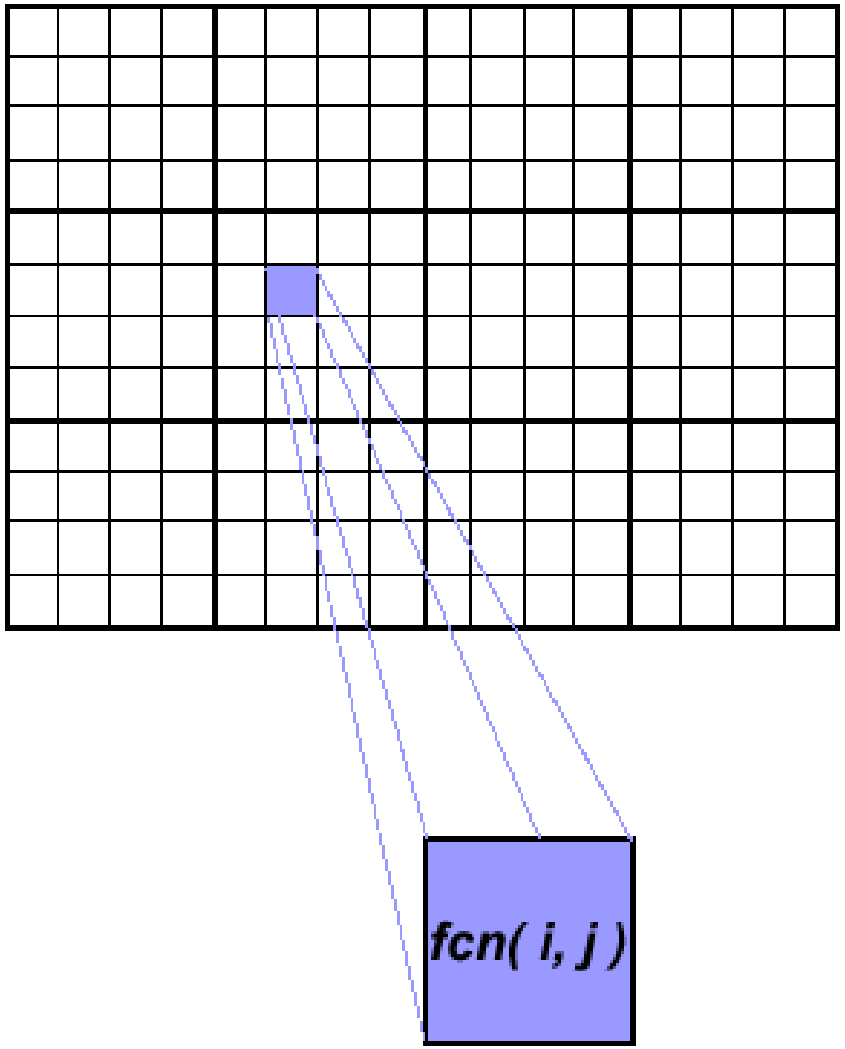
Obmedzenia a náročnosť paralelného spracovania

- Škálovateľnosť
  - Pridávanie výpočtových elementov spravidla nestačí
  - Algoritmy môžu obsahovať prirodzené obmedzenia na škálovateľnosť
  - HW obmedzenia
    - Šírka pásma pamäť – procesor
    - Šírka pásma komunikačnej siete
    - Veľkosť pamäte na jednom výpočtovom uzle

Príklady paralelného spracovania

- Spracovanie 2D poľa
  - Spracovanie prvku poľa nezávislé od ostatných prvkov
  - „Priamočiari“ („trápny“) paralelizmus
  - Spracovanie prvku primerane výpočtovo náročné
  - “ Sekvenčné spracovanie – jeden prvok po druhom:

Príklady paralelného spracovania

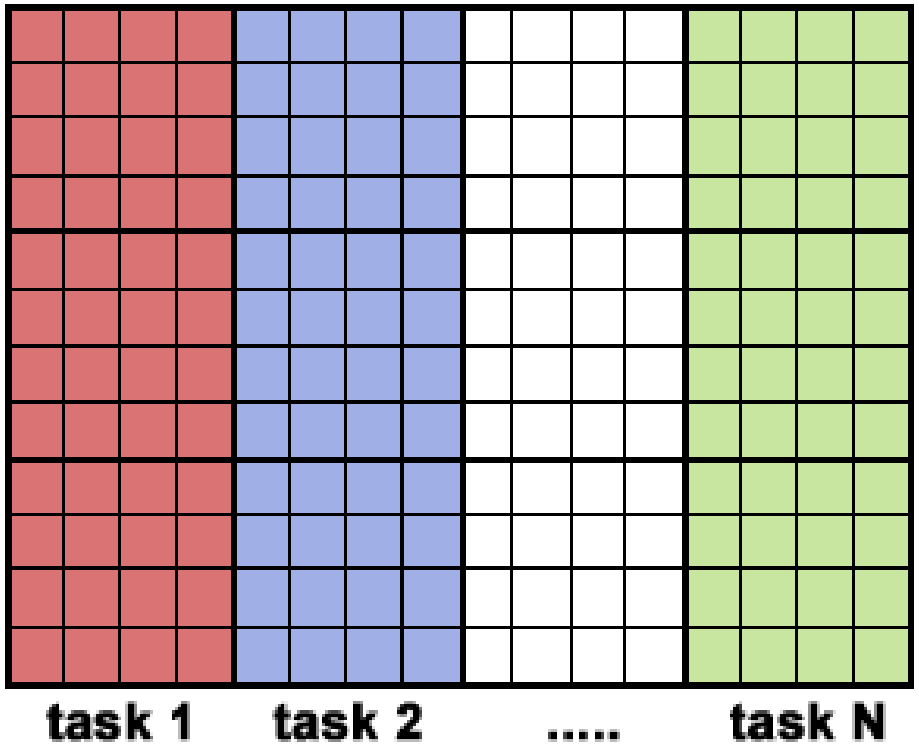



```
for j = 1 to n
 for i = 1 to n
  a[i,j] = fcn(a[i,j])
 end
end
```

Príklady paralelného spracovania

- Každý procesor časť poľa
- Nezávislé spracovanie prvkov poľa – žiadna komunikácia
- Rozdelenie na procesory tak, aby veľkosť kroku v cykle bola 1 – maximalizácia využitia vyrovnávacej pamäte
- Každá úloha spracuje časť cyklu nad dátami, ktoré „vlastní“

Príklady paralelného spracovania




```
for j = mystart to myend
 for i = 1 to n
  a[i,j] = fcn(a[i,j])
 end
end
```

Príklady paralelného spracovania

- “Master – worker” paradigma
- “Master“
  - „Vlastní“ pole dát
  - Zasiela prácu
  - Príma výsledky
- “Worker“
  - Príjma prácu
  - Realizuje výpočet
  - Zasiela výsledky

Príklady paralelného spracovania

```
if MASTER then
  initialize the array
  send each WORKER info on part of array it owns
  send each WORKER its portion of initial array
  receive from each WORKER results
end
if WORKER then
  receive from MASTER info on part of array it owns
  receive from MASTER portion of initial array
  calculate my portion of array
  for j = myfirstcolumn to mylastcolumn
    for i = 1 to n
      a[i,j] = fcn(a[i,j])
  send MASTER results endif
end
```

Príklady paralelného spracovania

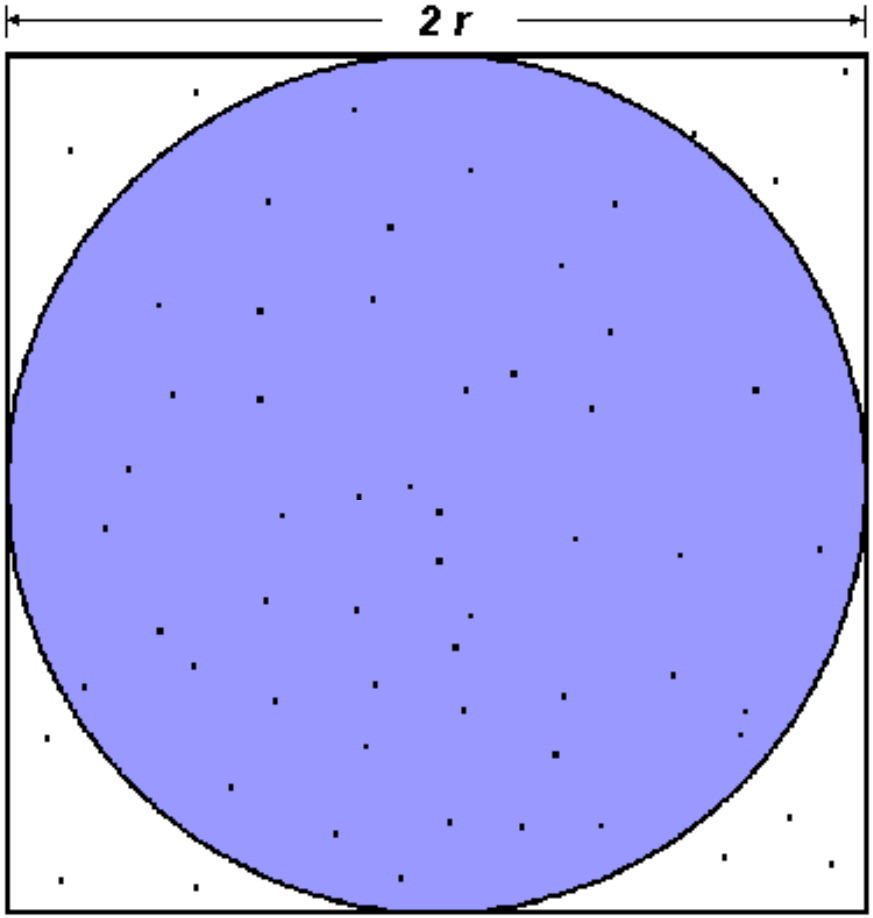
- “Pool of tasks” paradigma
- “Master“
  - Vlastní zásobník („bazén“) úloh
  - Zasiela prácu, keď je požadovaná
  - Zbiera výsledky
- “Worker“
  - Požaduje prácu
  - Realizuje výpočet
  - Zasiela výsledky
- „Worker“ – obdržaná časť práce určovaná až za behu
- Dynamické vyrovnávanie záťaže

Príklady paralelného spracovania

- Výpočet PI
  - Kruh vo štvorci
  - Náhodné generovanie bodov vo štvorci
  - Určiť body ktoré sú aj v kruhu
  - PI ~ 4 Nk / N

  - „Priamočiari“ („trápny“) paralelizmus
  - Minimálna komunikácia a IO operácie

Príklady paralelného spracovania



$A_S = (2r)^2 = 4r^2$

$A_C = \pi r^2$

$\pi = 4 \times \frac{A_C}{A_S}$

```
npoints = 10000
circle_count = 0
for j = 1 to npoints
  xcor = rnd(-1,1)
  ycor = rnd(-1,1)
  if incircle(xcor, ycor)
    circle_count++
end
PI =4.0*circle_count/npoints
```

Príklady paralelného spracovania

- Rozloženie cyklus na časti vykonateľné paralelne
- Každá úloha vykonáva svoju časť cyklu
- Žiadna komunikácia – nie sú potrebné informácie od ostatných úloh
- Master – Worker paradigma

Príklady paralelného spracovania

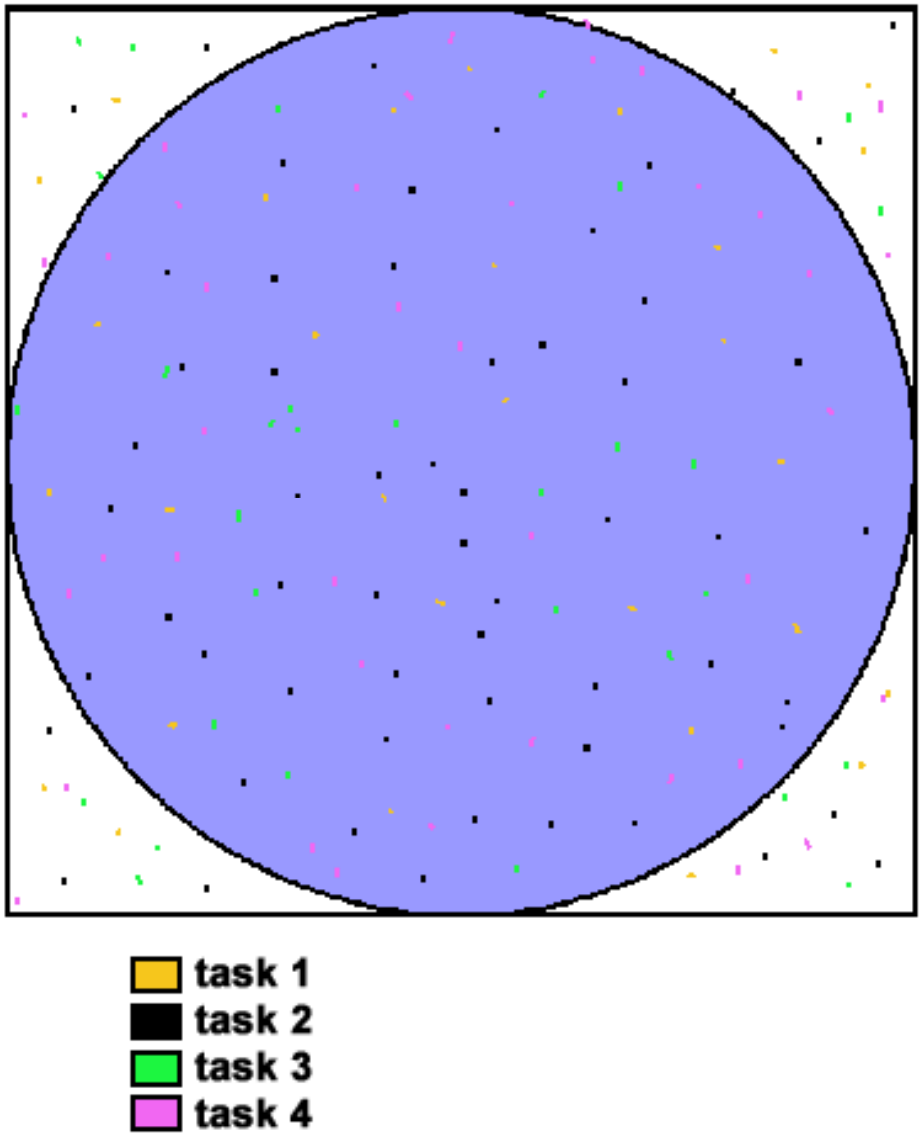
```
npoints = 10000
circle_count = 0
my_circle_count = 0
my_npoints = npoints / number of tasks

for j = 1 to my_npoints
  xcor = rnd(-1,1) and ycor = rnd(-1,1)
  if incircle(xcor, ycor) my_circle_count++

if WORKER send to MASTER my_circle_count

if MASTER
  receive circle_count from WORKERS
  PI =4.0 * circle_count/npoints
```

Príklady paralelného spracovania



Príklady paralelného spracovania

- Šírenie tepla
  - Väčšina problémov paralelného spracovania vyžaduje komunikáciu medzi úlohami
  - Mnohé problémy vyžadujú komunikáciu medzi susednými úlohami
  - Zmeny tepla v čase určené iniciálnymi hodnotami a okrajovými podmienkami
  - Metóda konečných diferencií na numerické riešenie diferenčnej rovnice v štvorcovej mriežke
  - Iniciálna teplota je 0 na okraji a vysoká v strede mriežky, teplota na okraji je udržiavaná na 0
  - Výpočet v bode mriežke závisí od okolitých hodnôt

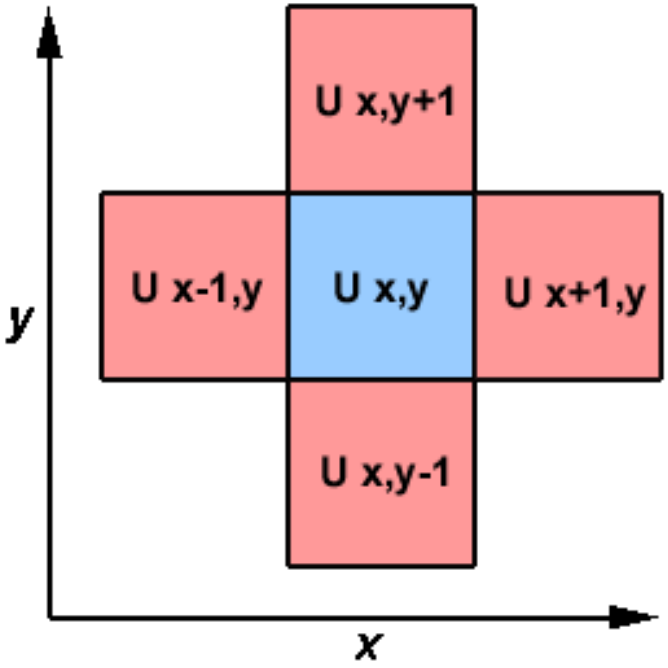
Príklady paralelného spracovania

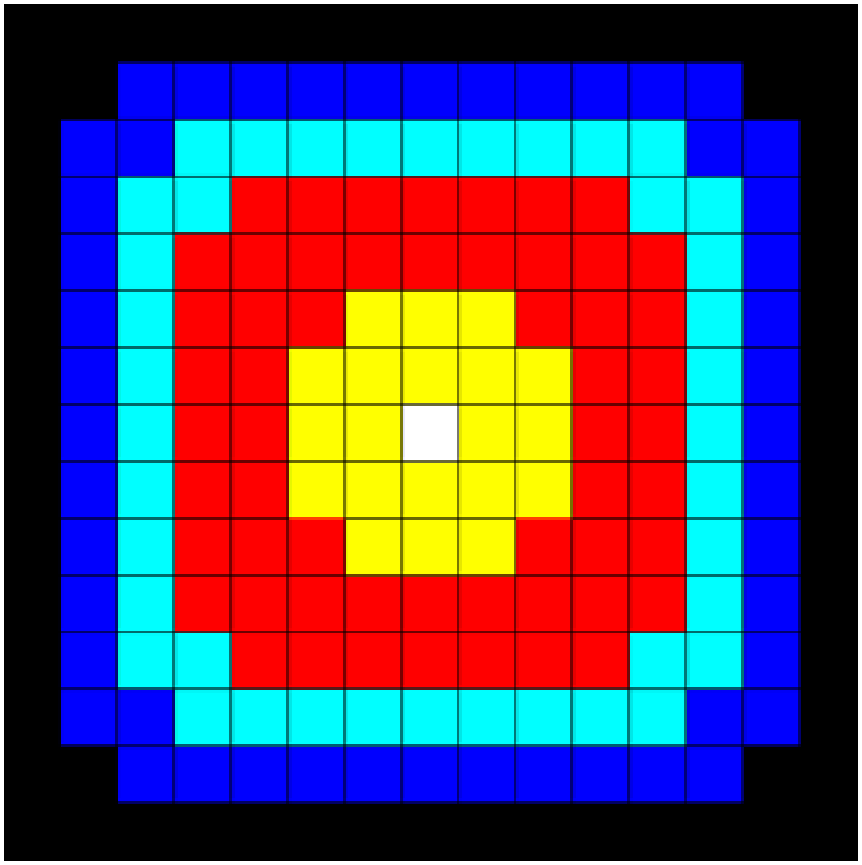
```
for iy = 2 to ny - 1
  for ix = 2 to nx - 1
    u2(ix, iy) = u1(ix, iy) +
      cx * (u1(ix+1,iy) + u1(ix-1,iy) - 2.*u1(ix,iy)) +
      cy * (u1(ix,iy+1) + u1(ix,iy-1) - 2.*u1(ix,iy))
  end
end
```

$$U_{x,y} = U_{x,y}$$

$$+ C_x * (U_{x+1,y} + U_{x-1,y} - 2 * U_{xy})$$

$$+ C_y * (U_{x,y+1} + U_{x,y-1} - 2 * U_{x,y})$$





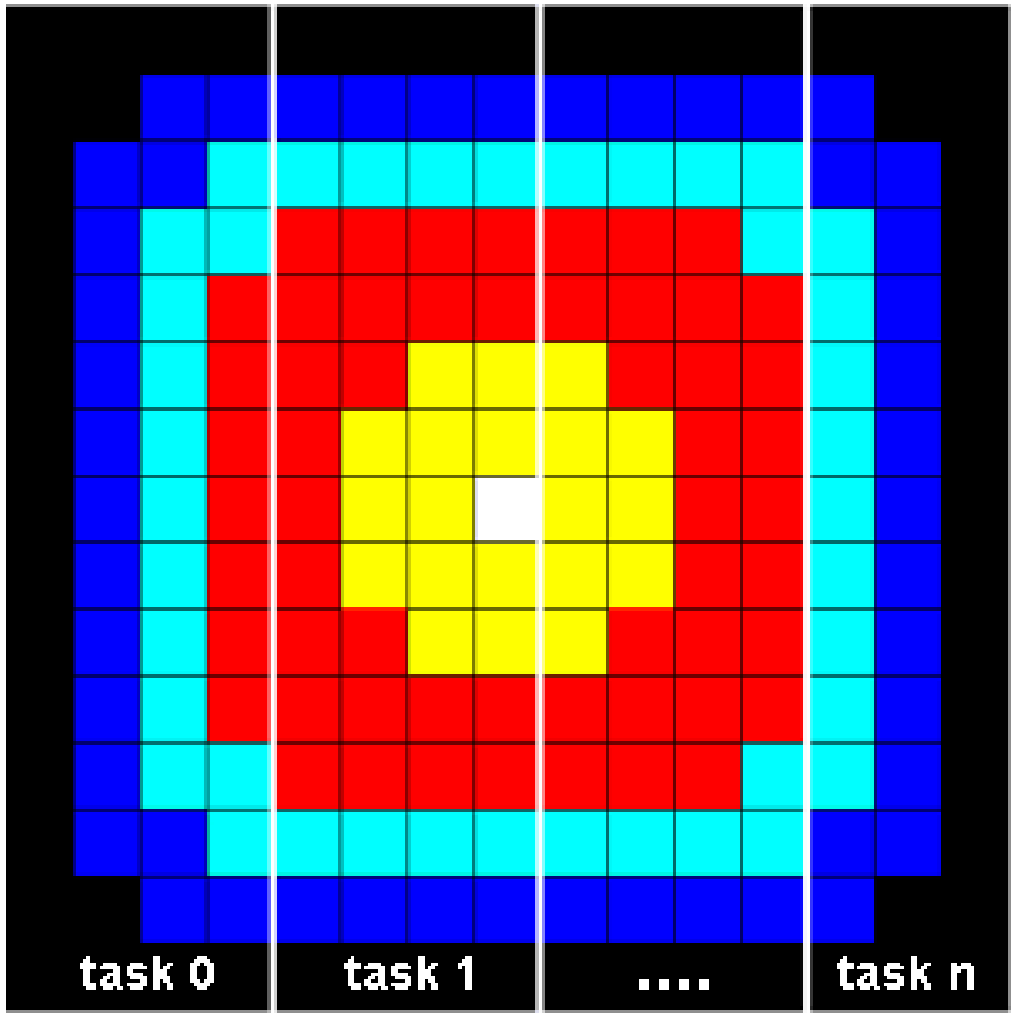
Príklady paralelného spracovania

- 2D pole je rozdelené a časti poľa distribuované jednotlivým úlohám, každá úloha vlastní časť poľa
- Určenie dátovej závislosti
- Vnútorné prvky častí poľa nezávisia od ostatných úloh
- Okrajové prvky častí poľa závisia od častí poľa susedných úloh, potrebná komunikácia

Príklady paralelného spracovania

- Master – worker paradigma
- Master
  - Rozposlanie iniciálnych informácií
  - Sledovanie konvergencie
  - Zber výsledkov
- Worker
  - Výpočet vo svojej časti poľa
  - Komnunikácia so susedmi

Príklady paralelného spracovania



Príklady paralelného spracovania

```
if MASTER then
    initialize array
    send each WORKER starting info and subarray
    until all WORKERS converge
        gather from all WORKERS convergence data
        broadcast to all WORKERS convergence signal
    end
    receive results from each WORKER
end
If WORKER then
    receive from MASTER starting info and subarray
    until solution converged
        send neighbors my border info
        receive from neighbors their border info
        update my portion of solution array
        determine if my solution has converged
        send MASTER convergence data
        receive from MASTER convergence signal
    end
    send MASTER results
end
```

Príklady paralelného spracovania

- Blokujúca komunikácia – čakanie na ukončenie komunikácie a až potom výpočet
- Odkomunikovanie hodnôt na okrajoch a následne spracovanie vlastnej časti poľa

- Skrátenie času behu použitím neblokujúcej komunikácie – výpočet počas komunikácie
- Každá úloha môže upraviť prvky vo vnútornej časti poľa počas zasielania okrajových dát

Príklady paralelného spracovania

```
if MASTER then
    initialize array
    send each WORKER starting info and subarray
    until all WORKERS converge
        gather from all WORKERS convergence data
        broadcast to all WORKERS convergence signal
    end
    receive results from each WORKER
end
If WORKER then
    receive from MASTER starting info and subarray
    until solution converged
        non-blocking send neighbors my border info
        non-blocking receive neighbors border info
        update interior of my portion of solution array
        wait for non-blocking communication complete
        update border of my portion of solution array
        determine if my solution has converged
        send MASTER convergence data
        receive from MASTER convergence signal
    end
    send MASTER results
end
```

Terminológia

- Úloha (Task)
- Paralelná úloha (Parallel Task)
- Sériové resp. sekvenčné spracovanie
- Paralelné spracovanie
- Prúdové spracovanie (Pipelining)
- Zdieľaná pamäť (Shared Memory)
- Distribuovaná pamäť (Distributed Memory)
- Symetrický multiprocesor (SMP)
- Počítačový klaster
- Viacjadrový procesor (Multicore)

Terminológia

- Komunikácia
- Synchronizácia
- Granularita
- Zrýchlenie
- Paralelná nadbytočnosť (Overhead)
- Masívny paralelizmus
- „Trápny“ paralelizmus (Embarrassingly parallel)
- Škálovateľnosť
- Vysokovýkonné počítanie (High Performance Computing, Supercomputing)

Zdroje

- Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. Introduction to ParallelComputing, 2nd Edition, Addison-Wesley 2003„Introduction to Parallel Computing“ http://www-users.cs.umn.edu/~karypis/parbook/
- Manferdelli, J. "The Many-Core Inflection Point for Mass Market Computer Systems," CTWatch Quarterly, Volume 3, Number 1,February 2007. http://www.ctwatch.org/quarterly/articles/2007/02/the-many-core-inflection-point-for-mass-market-computer-systems/
- Blaise Barney, Lawrence Livermore National Laboratory: Introduction to Parallel Computing. https://computing.llnl.gov/tutorials/parallel_comp/

- Obrázky prevzaté z:
  - Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. Introduction to ParallelComputing, 2nd Edition, Addison-Wesley 2003„Introduction to Parallel Computing“ http://www-users.cs.umn.edu/~karypis/parbook/
  - Blaise Barney, Lawrence Livermore National Laboratory: Introduction to Parallel Computing. https://computing.llnl.gov/tutorials/parallel_comp/