Obliczenia równoległe

Wojciech Kwedlo
Wydział Informatyki PB
wkwedlo@ii.pb.bialystok.pl
http://aragorn.pb.bialystok.pl/~wkwedlo
pokój 205

Plan wykładu

(1) Wstęp

(2) Programowanie systemów z przesyłaniem komunikatów przy pomocy MPI.

(3) Programowanie systemów ze wspólną pamięcią przy pomocy OpenMP

(4) Techniki projektowania i modelowania wydajności algorytmów równoległych

(5) Wybrane algorytmy równoległe

Literatura

- Niestety tylko po angielsku. Przygotowując wykład opierałem się na książkach:
  - A. Grama, A. Gupta, G. Karypis, V. Kumar, Introduction to Parallel Computing, Addison – Wesley, 2003.
  - B. Wilkinson, M. Allen. Parallel programming: Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall, 1999
  - W. Gropp, E. Lusk, A. Skjellum, Using MPI, Mit Press, 2nd edition, 1999.
  - M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, MPI: The Complete Reference, Mit Press, 1996. **dostępna legalnie w Internecie**.
  - B. Chapman, G. Jost , R. v. d. Pas, Using OpenMP: Portable Shared Memory Parallel Programming, MIT Press, 2007.
  - M.J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw Hill, 2003.

Jednostki i skale w obliczeniach równoległych

- Flop (floating point operation) – operacja zmiennoprzecinkowa. Liczba zmiennoprzcinkowa (double) zajmuje 8 bajtów.
- Flops/s – liczba operacji zmiennopozycyjnych na sekundę

Mega	Mflop/s = 10^6 flop/sec	Mbyte = 2^{20} = 1048576 ~ 10^6 bytes
Giga	Gflop/s = 10^9 flop/sec	Gbyte = 2^{30} ~ 10^9 bytes
Tera	Tflop/s = 10^{12} flop/sec	Tbyte = 2^{40} ~ 10^{12} bytes
Peta	Pflop/s = 10^{15} flop/sec	Pbyte = 2^{50} ~ 10^{15} bytes
Exa	Eflop/s = 10^{18} flop/sec	Ebyte = 2^{60} ~ 10^{18} bytes

Zaliczenie

- Wykład:
  - Egzamin składający się z o.k. 20 pytań testowych.
- Pracownia specjalistyczna:
  - rozgrzewka (zapoznanie się z laboratorium gridowym).
  - 1 lub 2 projekty (jeszcze się zastanawiamy).
- Zaliczenie pracowni specjalistycznej nie jest warunkiem dopuszczenia do egzaminu.

Miary wydajności komputerów

- Peak performance (Peak advertised performence). Maksymalna teoretyczna wydajność, w praktyce bardzo trudna do osiągnięcia. Przykład. Pentium IV 3 Ghz ma przepustowość dwóch operacji zmiennoprzecinkowych w jednym takcie zegara.
  - Peak peformance = 6 Gflops/s
  - Komputer składający się z 1000 takich procesorów 6 Teraflops/s
  - W praktyce osiągnięcie takiej wydajności nie jest możliwe
- Benchmark Linpack „Hello world” obliczeń równoległych
  - Rozwiązanie (eliminacja Gaussa – dekompozycja LU) układu równań Ax=b, gdzie A macierz nxn; x,b wektory n -wymiarowe
- Stosowany przy rankingu komputerów na liście Top500.

System równoległy

- Na wykładzie skupimy się wyłącznie na systemach wieloprocesorowych, zawierających więcej niż jeden procesor.
- Pomijamy równoległość wewnątrz pojedynczego procesora (super skalarność, VLIW) , architektury specjalne (np. sieci systoliczne)
- Idea jest prosta:

Skoro jeden procesor rozwiązuje problem w N godzin, może warto użyć N procesorów, aby rozwiązać problem w (powiedzmy) jedną godzinę

- Pytanie 1: Czy nie można poczekać i kupić N razy szybszy procesor. Innymi słowy dlaczego bardzo szybki komputer musi być komputerem równoległym.
- Pytanie 2: Do jakich zagadnień potrzebujemy tak szybkich komputerów.

Symulacja – trzeci filar nauki

- Tradycyjny paradygmat w nauce i inżynierii.
  - Stwórz teorię lub projekt na papierze.
  - Wykonaj eksperyment lub zbuduj system.
- Ograniczenia
  - zbyt trudne (np. duże tunele aerodynamiczne)
  - zbyt drogie (test zderzeniowy jumbo jeta, odwiert w poszukiwaniu ropy)
  - zbyt powolne (ewolucja galaktyk, klimatu)
  - zbyt niebezpieczne (broń jądrowa, farmaceutyki, wpływ człowieka na klimat)
- Trzecia droga: wykorzystaj bardzo szybki komputer do symulacji zjawiska.
  - Bazując na prawach fizyki i metodach numerycznych
  - Trudne (ang. grand challenge) problemy wymagają naprawdę szybkich komputerów

Przykład pierwszy modelowanie klimatu

- Atmosfera jest dzielona na trójwymiarowe komórki. Obliczenia w każdej komórce wykonywane wielokrotnie (upływ czasu).
- Podzielmy globalną atmosferę ziemską na komórki o wymiarze 1mila x 1mila x 1 mila. Wysokość atmosfery to 10 komórek. Łącznie około $5*10^8$ komórek.
- Niech jedna komórka wymaga 200 flopów (typowa wartość dla równań Naviera-Stokesa). W jednym kroku 10^{11} = 100 Gflops
- Prognoza pogody na 7 dni zakładając krok czasowy 1 minutę przez komputer o szybkości 1 Gflop/s wymaga 10 dni.
- Jeżeli chcemy zakończyć obliczenia w 5 minut potrzebujemy komputera o szybkości 3.4 Tflop/s

- Rzeczywiste modele są dużo bardziej skomplikowane. (np. modelowanie klimatu w skali dziesiątków, setek lat).

Przykład drugi: ewolucja galaktyki

- Problem N ciał. Modelowanie polega na obliczeniu sił grawitacji działających na każde z ciał. Po obliczeniu nowej pozycji obliczenia są ponawiane (upływ czasu). Naiwny algorytm ma złożoność O(N*N) (każde ciało przyciąga wszystkie pozostałe), algorytmy zaawansowane O(NlogN).

- Galaktyka zawiera około 10^{11} gwiazd (potrzeba pamięci rzędu TB)

- Zakładając że obliczenie łącznej siły grawitacji działającej na jedną gwiazdę zajmuje 1ms (bardzo optymistyczne dla szeregowego komputera), to potrzebujemy ponad jednego roku na jedną iterację, zakładając algorytm O(NlogN).

Zagadnienia wymagające dużych mocy obliczeniowych

- Badania naukowe
  - Globalne modelowanie klimatu
  - Biologia: genomika, prognozowanie struktury białka
  - Modelowanie zjawisk astrofizycznych (ewolucja galaktyk, czarne dziury)
  - Chemia obliczeniowa
- Inżynieria
  - Projektowanie układów scalonych
  - Dynamika płynów (projektowanie samolotów, rakiet)
  - Symulacja zderzeń
- Biznes
  - Modelowanie finansowe i ekonomiczne
  - Data mining
- Wojsko
  - Symulacje broni jądrowej
  - Kryptografia

Ograniczenia szybkości komputerów szeregowych

- Prawo Moore'a (G. More – współzałożyciel Intela). Liczba tranzystorów i szybkość mikroprocesora podwaja się co 18 miesięcy.
- ***Prawo Moore'a przestało działać !!!***
  - Ograniczenia związane z odprowadzaniem ciepła ze struktury utrudniają dalszy wzrost prędkości zegara. (Moc tracona jest proporcjonalna do prędkości zegara).
  - Dalszy wzrost prędkości zegara prowadzi do innych barier fizycznych. Np. Istnieją tranzystory działające z prędkością 1THz, ale w czasie jednego cyklu światło pokonuje 0.3mm !!!.
  - Predykcja: nastąpi wzrost liczby tranzystorów na strukturze, ale wzrost szybkości zegara będzie bardzo powolny.
- Gdzie zainwestować dodatkowe tranzystory ? Inwestycja w wieksze pamięci cache, więcej jednostek funkcjonalnych przestaje się opłacać. (typowy program nie jest w stanie ich wykorzystać)
- Albo zupełnie nowy paradygmat (kwantowy ?? biologiczny ??) albo wiele procesorów na jednym chipie.

Lista Top500 (www.top500.org)

Rank	Site	Computer	Processors	Year	R_{max}	R_{peak}
1	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution IBM	131072	2005	280600	367000
2	Oak Ridge National Laboratory United States	Jaguar - Cray XT4/XT3 Cray Inc.	23016	2006	101700	119350
3	NNSA/Sandia National Laboratories United States	Red Storm - Sandia/ Cray Red Storm, Opteron 2.4 GHz dual core Cray Inc.	26544	2006	101400	127411
4	IBM Thomas J. Watson Research Center United States	BGW - eServer Blue Gene Solution IBM	40960	2005	91290	114688
5	Stony Brook/BNL, New York Center for Computional Sciences United States	New York Blue - eServer Blue Gene Solution IBM	36864	2007	82161	103219
6	DOE/NNSA/LLNL United States	ASC Purple - eServer pSeries p5 575 1.9 GHz IBM	12208	2006	75760	92781
7	Rensselaer Polytechnic Institute, Computional Center for Nanotechnology Innovations United States	eServer Blue Gene Solution IBM	32768	2007	73032	91750
8	NCSA United States	Abe - PowerEdge 1955, 2.33 GHz, Infiniband Dell	9600	2007	62680	89587.2
9	Barcelona Supercomputing Center Spain	MareNostrum - BladeCenter JS21 Cluster, PPC 970, 2.3 GHz, Myrinet IBM	10240	2006	62630	94208
10	Leibniz Rechenzentrum Germany	HLRB-II - Altix 4700 1.6 GHz SGI	9728	2007	56520	62259.2

Miary skalowalności algorytmu równoległego

- Zakładamy stały rozmiar danych N
- T(1) czas obliczeń dla najlepszego algorytmu sekwencyjnego
- T(p) czas obliczeń dla algorytmu równoległego przy pomocy p procesorów
- ***Przyspieszenie*** (ang. speedup) S(p)

$$S(p)=\frac{T(1)}{T(p)}$$

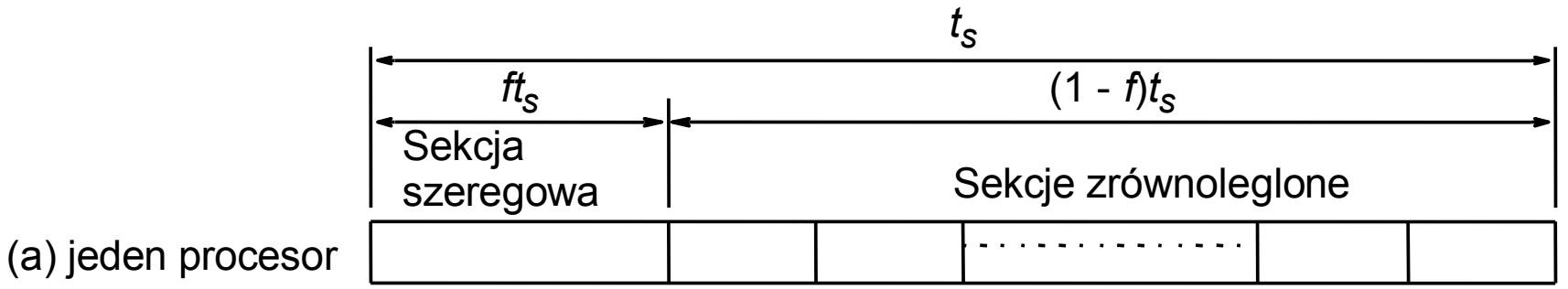
- Idealnie S(p)=p (***przyspieszenie liniowe***), w niektórych sytuacjach (rzadko) S(p)>p (przyspieszenie ***superliniowe*** ang. superlinear). W praktycznych algorytmach S(p)<p.
- Wydajność E(p)

$$E(p)=\frac{S(p)}{p}=\frac{T(1)}{p*T(p)}$$

- Wydajność jest stosunkiem osiągniętego przyspieszenia do idealnego przyspieszenia liniowego. Wydajność równa 1 oznacza przyspieszenie liniowe.

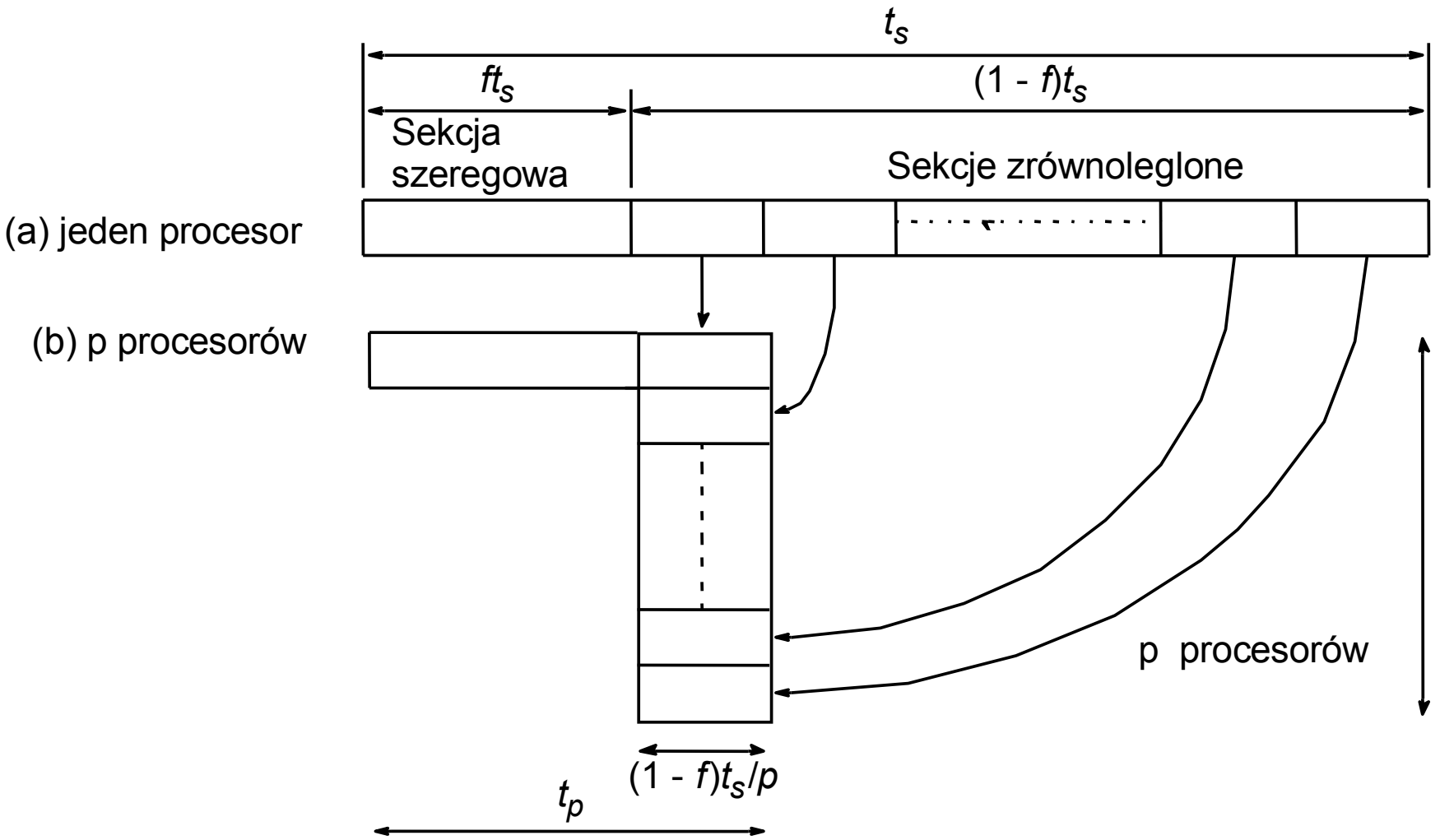
Prawo Amdahla

- G. Amdahl pracownik IBM, założyciel firmy Amdahl.
- Zakładamy że pewna część programu w ogóle nie ulega przyspieszeniu przez równoległe przetwarzanie danych. Pozostała część jest przyspieszona p-krotnie na p procesorach.
- t_s – czas sekwencyjnego wykonania programu. Niech f jest stosunkiem czasu fragmentu nie dającego się przyspieszyć (równego $f*t_s$) do całkowitego czasu t_s.



Prawo Amdahla

- t_p – czas obliczeń na p procesorach.



Prawo Amdahla – przyspieszenie

- Przyspieszenie *S(p)* dane jest wzorem:

$$S(p) = \frac{t_s}{f t_s + (1-f)t_s/p} = \frac{p}{1+(p-1)f}$$

zwanym prawem Amdahla.

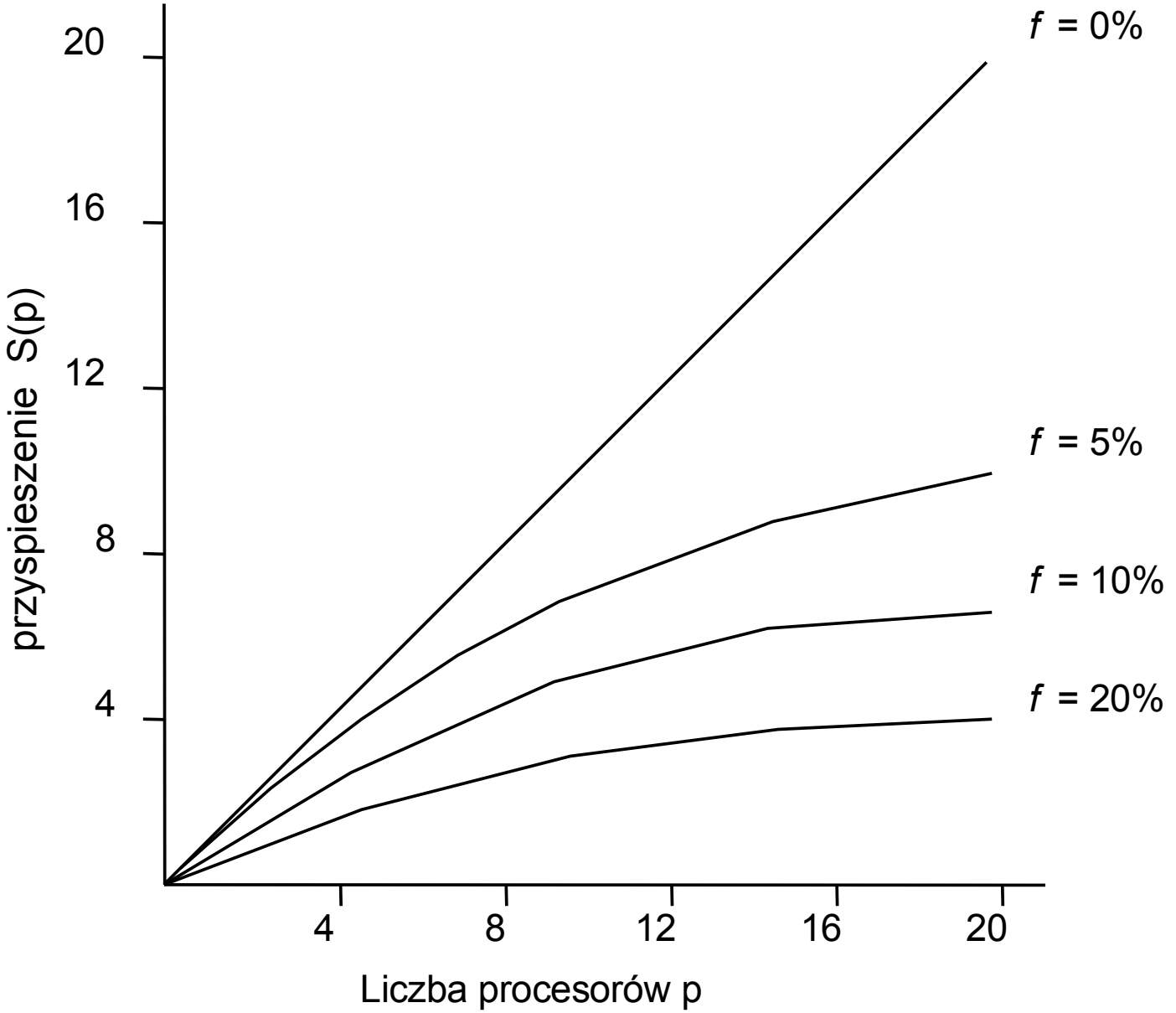
- Uwaga:

$$\lim S(p) = 1/f$$

A zatem dysponując nawet nieskończoną liczbą procesorów, nie przekroczymy przyspieszenia 1/f

Prawo Amdahla – maksymalne przyspieszenie

- dla *f=0 S(p)=p*



Prawo Amdahla - wnioski

- Na pierwszy rzut oka wnioski wydają się pesymistyczne: maksymalne przyspieszenie jest ograniczone z góry.
- W rzeczywistych problemach skalowalność jest ***gorsza*** od wynikającej z prawa Amdahla.
  - Koszty komunikacji procesorów (rosną wraz ze wzrostem ich liczby)
  - Niezrównoważenie obciążenia (ang. load imbalance)
  - Potrzeba duplikacji obliczeń na różnych procesorach.
- Istotnym założeniem w prawie Amdahla jest stały rozmiar problemu.
- To założenie ***nie jest spełnione***. Szybkie komputery buduje się po to aby rozwiązywać coraz bardziej skomplikowane problemy (np. budować złożone modele symulacyjne, grać w bardziej złożone gry)
- Dlatego też dziedzina obliczeń równoległych rozwija się dynamicznie..

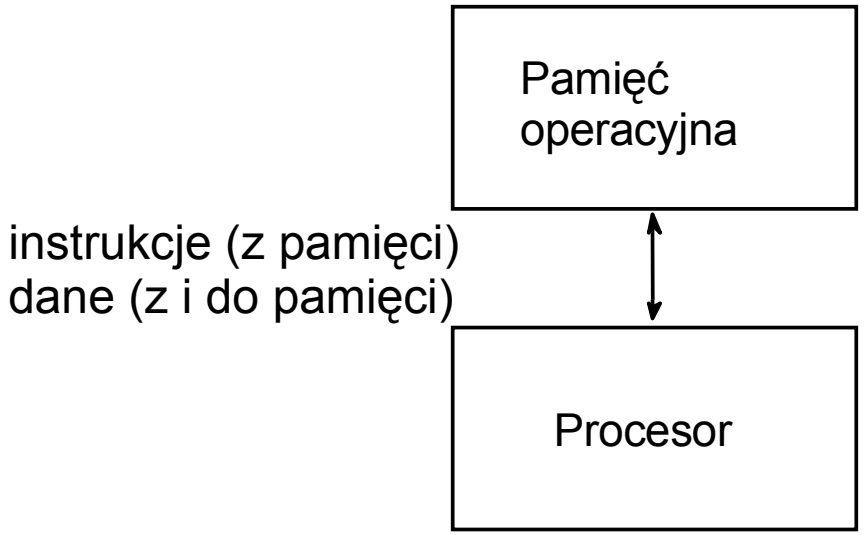
Klasyfikacja Flynn'a (1966)

- Flynn oparł swoją klasyfikację na liczbie przetwarzanych strumieni danych i rozkazów. Zdefiniował cztery kategorie SISD, MISD, SIMD, MIMD. Warto się z nimi zapoznać, bo występują w literaturze.
- SISD (single instruction single data) – klasyczny komputer sekwencyjny.
- MISD (multiple instruction single data) – nie występuje, niektórzy zaliczają architektury oparte na przetwarzaniu potokowym.
- SIMD (single instruction multiple data) – głównie wektorowe rozszerzenia procesorów
  - 3DNOW, SSE, SSE2 – Intel i AMD
  - AltiVec – Motorola/IBM/Apple (już bez Apple)
- MIMD (multiple instruction multiple data) wszystkie obecne maszyny wieloprocesorowe wpadają do tej kategorii, co ogranicza użyteczność klasyfikacji. Jednak z grubsza możemy podzielić je na:
  - ***Systemy wieloprocesorowe ze wspólną pamięcią***
  - ***Systemy wieloprocesorowe z pamięcią rozproszoną***

System ze wspólną (ang. shared) pamięcią

Konwencjonalny komputer

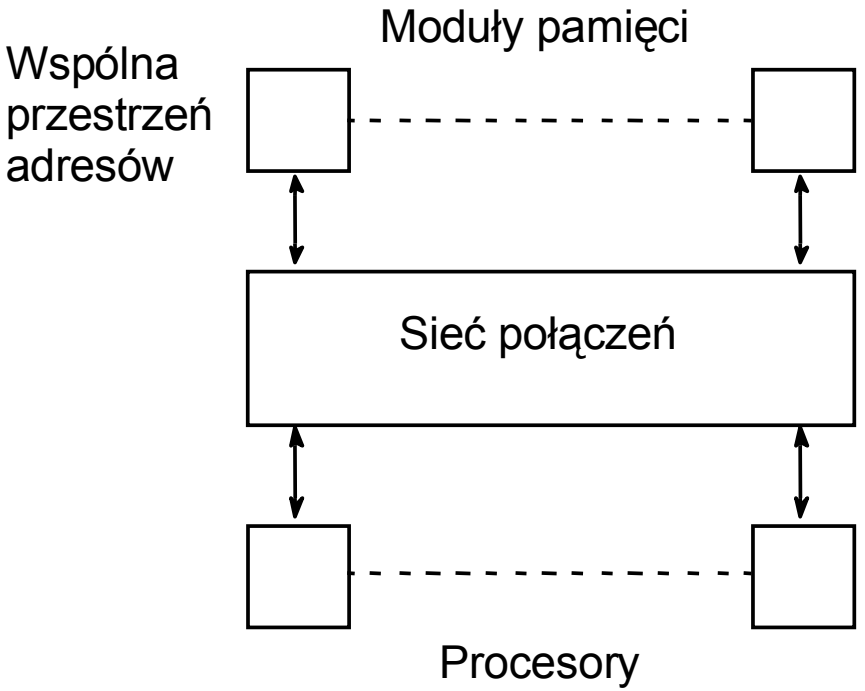
- Składa się z procesora wykonującego program przechowywany w pamięci operacyjnej.



- Każdy bajt pamięci głównej ma swój adres. 2^D adresów, gdzie D jest długością słowa w bitach.

System wieloprocesorowy ze wspólną pamięcią

- Każdy procesor ma dostęp do dowolnego modułu pamięci
- Bajt pamięci widoczny przez wszystkie procesory pod tym samym adresem
- Pamięci cache (indywidualne dla poszczególnych procesorów) komplikują sytuację.
- Procesory wielordzeniowe (ang. multi-core) należą do tej klasy

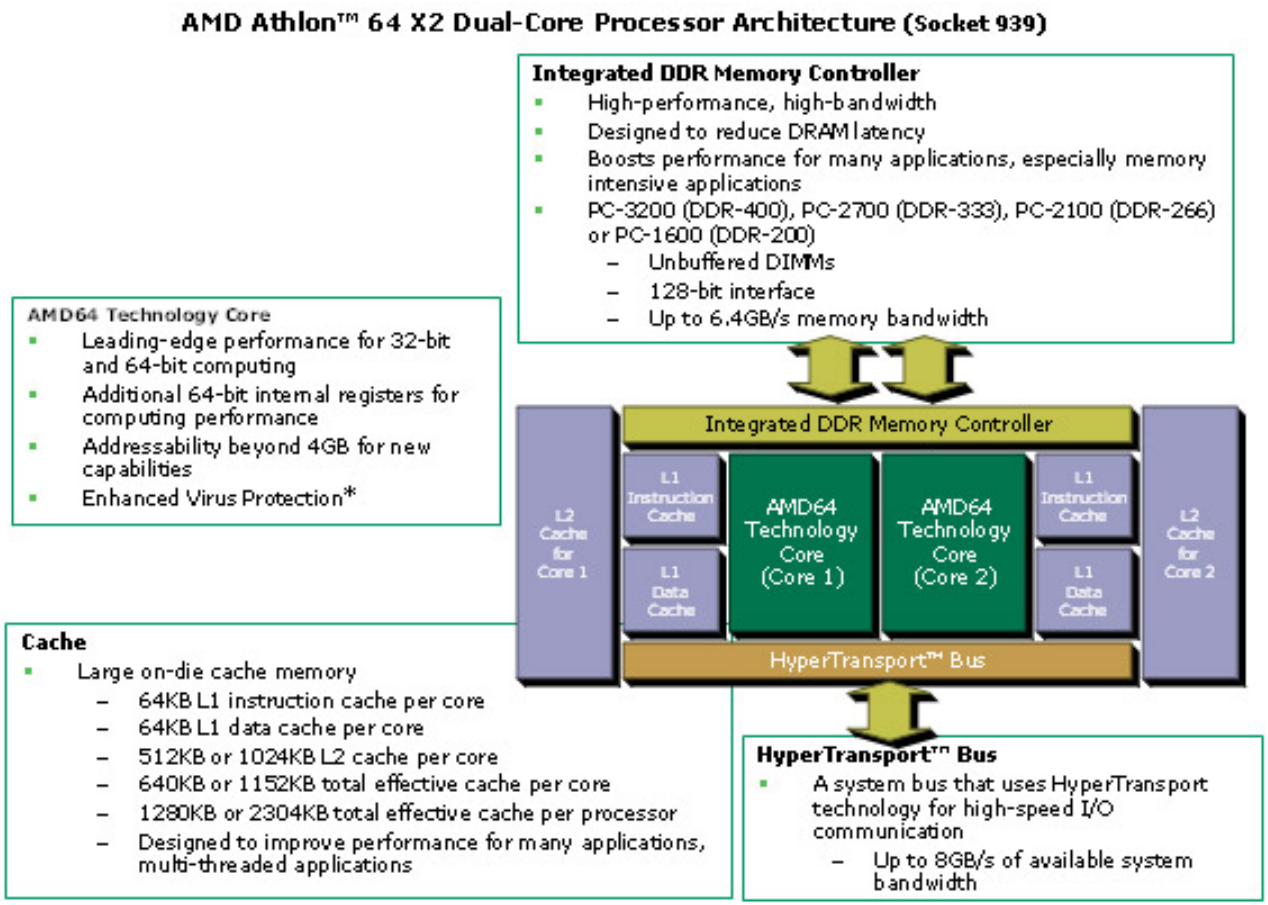


System wieloprocesorowy ze wspólną pamięcią - programowanie

- Generalnie stosuje się mechanizmy programowania wielowątkowego. System operacyjny (Linuks, Windows) „dba” o to, aby uruchomić różne wątki na różnych procesorach.
- Jeżeli uda się dobrze podzielić zadanie pomiędzy wątki=> uzyskamy skrócenie czasu obliczeń.
- Standardy dla wątków (niskopoziomowe):
  - POSIX threads => znane z systemów operacyjnych
  - Windows threads => pomijamy
  - Java threads => być może znane z programowania obiektowego; raczej nie nadaje się do obliczeń o wysokiej wydajności
- Nasz standard: ***OpenMP (www.openmp.org)***
  - wysokopoziomowy, wygodny (powiedzmy wygodniejsz niż PThreads i WinThreads).
  - w postaci dyrektyw #pragma omp dla kompilatora. Wymaga odpowiedniego kompilatora (np. gcc 4.2 albo icc – Intel C/C++ compiller)
  - gcc 4.2 do pobrania z internetu i zainstalowania na Linuksie (kompilacja ze źródeł trochę trwa; **gcc.gnu.org**)

Procesory wielordzeniowe

- Szczególnym przypadkiem systemu ze wspólną pamięcią jest system z procesorem ***wielordzeniowym*** (ang. multicore). Procesor wielordzeniowy to układ scalony zawierający kilka niezależnych procesorów.

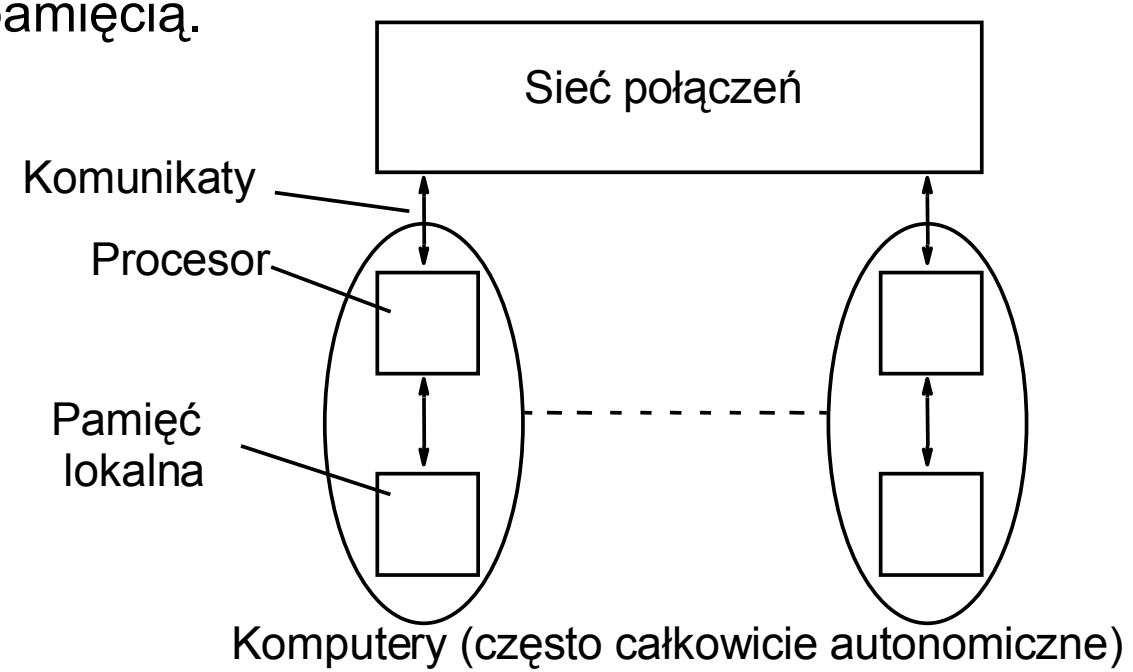


- Niedługo wszystkie komputery biurkowe będą systemami równoległymi. Zaprogramowanie dobrej gry 3D będzie wymagało wiadomości z tej dziedziny. Jest to kolejna powód, dla którego ***warto zainwestować w ten przedmiot.***

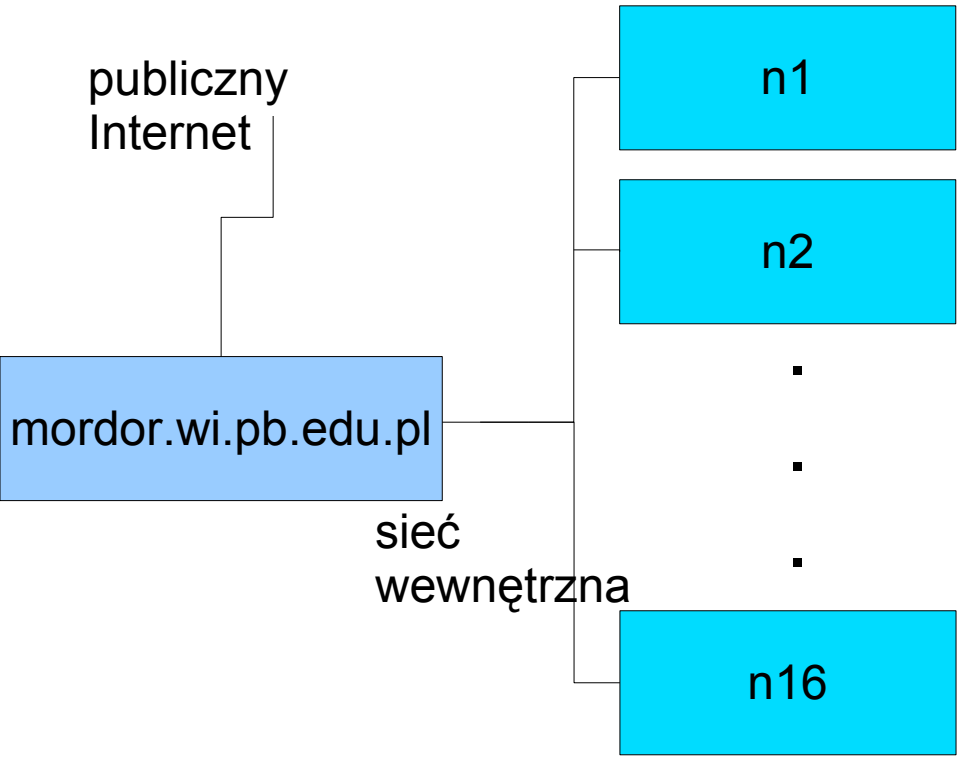
System z pamięcią rozproszoną (ang. distributed)

System z przesyłaniem komunikatów (ang. message passing)

- Również zwany systemem ***z przesyłaniem komunikatów*** (ang. message passing).
- Często klastry (ang. cluster) niezależnych systemów komputerowych.
- Każdy procesor ma dostęp wyłącznie do swojej lokalnej pamięci.
- Współpraca z innymi procesorami odbywa się poprzez wymianę komunikatów
- Sieć połączeń: dziś co najmniej Gigabit Ethernet (1Gb/s), często Infiniband (10 Gb/s lub 20 Gb/s) lub wyspecjalizowane rozwiązania.
- Możliwe rozwiązanie hybrydowe: każdy komputer jest systemem wieloprocesorowym ze wspólną pamięcią.

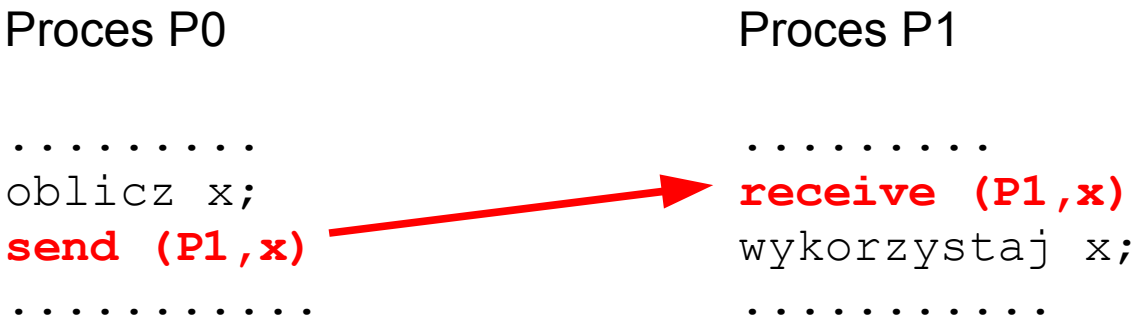


Architektura klastra obliczeniowego Mordor



- Węzeł zarządzający (mordor)
  - 2xXeon 2.800 MHz, 2 GB RAM
  - macierz RAID 1
  - sieciowy system plików (NFS)
  - system kolejkowy
  - monitoring klastra (Ganglia)
  - logowanie użytkowników
- Węzły obliczeniowe: (n1,n2, ..., n16)
  - 2xXeon 2.8GHz, 2 GB RAM
  - ***komputer ze wspólną pamięcią !!!***
  - 64-bitowy Linux
- Sieć wewnętrzna
  - Gigabit Ethernet (protokół TCP/IP)
  - Infiniband 4x (10 Gb/s, tylko dla aplikacji równoległych MPI)

System z przesyłaniem komunikatów - programowanie



- W programie umieszcza się wywołania powodujące wysyłanie i odbiór danych do/z innych procesów.
- Komunikacja jest jawna – upraszcza to projektowania wydajnych algorytmów (łatwiej jest minimalizować komunikację)
- Na zajęciach: standard MPI (ang. message passing interface) wzbogacający standardowy język programowania (C/C++/Fortran) o funkcje służące do przesyłania komunikatów.
- Darmowe implementacje można pobrać ze stron:

Model programowania a model maszyny

- Należy odróżnić model programowania od modelu maszyny
- Maszynę ze wspólną pamięcią jest bardzo łatwo programować przy pomocy modelu z przesyłaniem komunikatów.
- Wystarczy przeznaczyć część pamięci na bufor, w którym przechowywane są komunikaty.
- Dlatego też klastry maszyn ze wspólną pamięcią programowane są przy pomocy przesyłania komunikatów, chociaż możliwe jest programowanie hybrydowe MPI + OpenMP.
- Sytuacja odwrotna, w której maszyna z przesyłaniem komunikatów jest programowana przy pomocy modelu ze wspólną pamięcią jest bardzo trudna (niemożliwa bez specjalnego hardware'u) do realizacji.

Podsumowanie

- Programowanie maszyn równoległych jest trudną sztuką.
- Należy znaleźć dostatecznie dużo równoległości w algorytmie (prawo Amdahla)
- Należy zapewnić koordynację, komunikację i synchronizację (to ostatnie zagadnienie znane z systemów operacyjnych) – wszystko to kosztuje.
- Należy zadbać o maksymalnie lokalny wzorzec odwołań do pamięci – aby zmaksymalizować wykorzystanie pamięci cache.
- Należy zadbać aby wszystkie procesory były zajęte rozwiązując problem (zagadnienie równoważenia obciążenia – load balancing)