# O superkomputerach

- Marek Grabowski

# Superkomputery dziś

- Klastry obliczeniowe
- Szafy (od zawsze)
  - Bo komputery są duże
- Półki i blade'y (od pewnego czasu)
  - Większe upakowanie mocy obliczeniowej na m^2
  - Łatwiejsze chłodzenie
  - Można coś wyjąć lub włożyć

# Stare Cray'e X1


źródło: Wikipedia

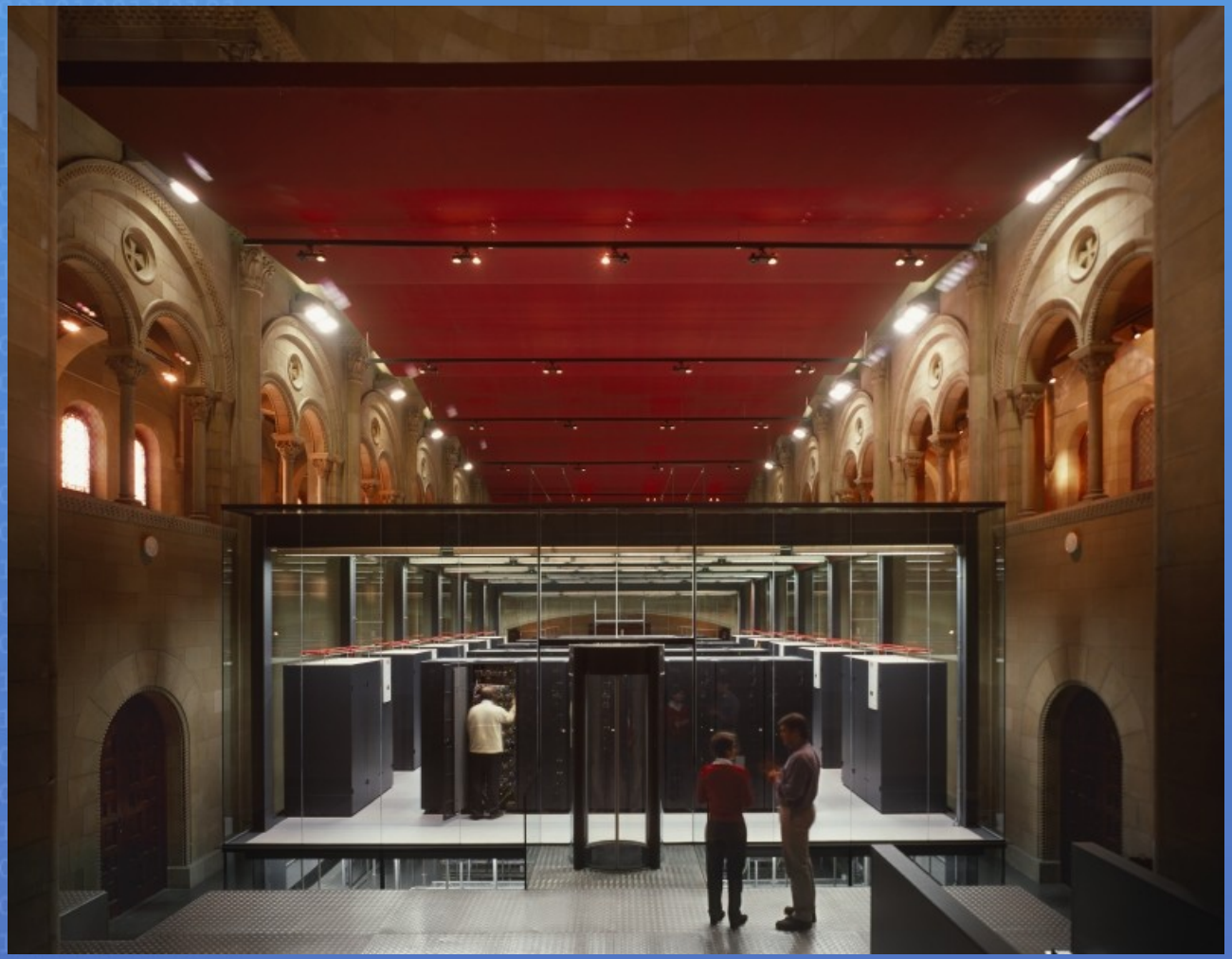
# Nowy Sun Constellation


źródło: SUN

# IBM Roadrunner – Los Alamos:)

źródło: Wikipedia

# MareNostrum



źródło: Wikipedia

# Zastosowania

- Symulacje fizyczne
  - Prognozy pogody
  - Zmiany klimatu
  - Testowanie materiałów
  - Najstarsze, czyli
    - Produkcja bomb atomowych
- Bioinformatyka
  - Badanie białek
  - Testowanie leków
- inne

# Architektura ogólnie

- Dużo „niezależnych“ jednostek obliczeniowych
- Potrzebne szybkie łącza
  - Infiniband, Myrinet, 100GB Ethernet(?)
- Potrzeba DUŻO energii i dobrego chłodzenia
  - EarthSimulator miał własną elektrownię..
- TOP500 i benchmarki
  - Linpack

# Trochę szczegółów

- Procesory
  - Intel
  - AMD
  - SUN Spark
  - IBM Cell
- Akceleratory
  - Nvidia
  - ATI
  - ClearSpeed

# Co mamy

- Dużo procesorów, każdy z własnym cachem
  - dość szybie połączenia między procesorami
  - problemy ze spójnością cache'a
- Dużo półek/bladów każda z własną pamięcią
  - znośna prędkość odczytu z pamięci
  - NUMA
  - stosunkowo wolna komunikacja między półkami
- Dużo miejsca na dyskach
  - o prędkości lepiej nic nie mówić

# Co chcemy

- Wyżyłować kod tak jak się da
  - przy obliczeniach idących w godziny, dni, lub tygodnie stała MA znaczenie
- Maksymalnie wykorzystywać cach'e
- Jak najlepiej używać pamięci (NUMA)
- Rozpraszać obliczenia na wiele procesorów
- Łatwo pisać współbieżne programy

# Wracamy na ziemię

- CHCEMY to co było na poprzednim slajdzie. Może za kilka lat chociaż częściowo będzie to możliwe...
- Prawo Amdahla na maksymalne przyspieszenie programu
  - $\frac{1}{(1-P)+\frac{P}{S}}$
  - P - część programu dająca się zrównoleglić
  - S – przyspieszenie części równoległej

# Smutna rzeczywistość

- Java raczej się nie nadaje
  - Obiektowość ogólnie jest zła
- Biblioteki obliczeniowe są napisane w Fortranie
  - szczęśliwi ci, którzy nigdy nie widzieli tego języka
- Automatyczne wykorzystywanie rozmieszczenia pamięci to bajka
- Kompilatory czasami działają
  - gcc wcale nie jest fajne

# Jak się pisze programy

- Dwie biblioteki: MPI i OpenMP
  - dużo różnych, różnie działających kompilatorów
  - wersje do C i Fortranu
- DARPA sponsoruje prace nad innymi językami
  - SUN Fortress
  - IBM X10
  - Cray Chapel
- Jest źle, może będzie lepiej

# MPI

- Message Passing Interface
  - jak sama nazwa wskazuje polega na przekazywaniu komunikatów
- Zakładamy, że każdy proces ma prywatną pamięć
- Program jest powielany na wszystkie procesy (coś a'la SIMD)
- Chyba jest na jednym z labów na Sieciach...

# MPI

- Na początku inicjalizujemy MPI
- Każdy proces dostaje swój unikalny numer
- Potem, w chwilach kiedy chcemy się skomunikować używamy
  - MPI_Send
  - MPI_Bcast
  - MPI_Recv
- Na koniec wymeldowujemy się z systemu

# MPI

- Są różne wersje standardowych poleceń
  - blokujące/nieblokujące
  - probe'y
- Można łączyć procesy w grupy (np. dla broadcasta)
- Różne mechanizmy synchronizacji
  - bariery
- Programy trzeba uruchamiać poleceniem mpiexec

# MPI

- Popularne
- Jak się dobrze napisze program, to szybko działa
- Pełna kontrola nad tym co się dzieje
- Łatwo kontrolować skalowalność
- Strasznie się w tym pisze
- Kompilatory czasem wariują

# OpenMP

- Procesy mają wspólną pamięć
- Mają też obszar pamięci prywatnej
- Mechanizmy synchronizacji
  - bariery
  - redukcje
- Ma postać dyrektyw dodawanych do kodu
- Ogólnie polega na forkowaniu procesu na czas wykonania części równoległej
- Także SIMD

# OpenMP

- Bierzemy program sekwencyjny
- W miejscach, które chcemy wykonywać równolegle dodajemy dyrektywę typu
  - #pragma omp parallel
- W dyrektywie deklarujemy, które zmienne chcemy mieć prywatne (np. akumulator), a które dzielone (np. licznik pętli)
- Można dodawać różne bajery, jak warunkowe zrównoleglanie

# OpenMP

- Każdy z procesów wykonujących część równoległą dostaje unikalny numer
- Po zakończeniu zrównoleglonego kawałka programu, możemy chcieć wykonać redukcję, czyli np. jakąś zmienną ze wszystkich procesów
- Możemy używać barier, żeby szybsze procesy czekały na wolniejsze

# OpenMP

- Wolniejsze niż dobrze napisane MPI
- Automatycznie zarządzana komunikacja i synchronizacja
- Łatwiejsze w użyciu niż MPI
- Podobno dobrze się skaluje (podobno)
- IMHO jeszcze nie dopracowane – jakieś problemy z zagnieżdżeniami itp.
- Kompilator też czasami potrafi robić dziwne rzeczy

# Na zakończenie

- HPC to jeszcze dziki zachód na początku kolonizacji
  - właściwie nic nie jest zrobione dostatecznie dobrze
  - ciągle pojawiają się nowe problemy – jak NUMA
  - automatyzacja czegokolwiek kuleje
  - różne firmy i rządowe instytucje bardzo chętnie wpuszczają w to pieniądze:)
  - bardzo brakuje ludzi

# Dziękuję za uwagę