UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ

Introdução a Computação Paralela

Prof. João Marcelo Uchôa de Alencar

joao.marcelo@ufc.br

Todo Computador é Paralelo

- Dentro de um computador moderno, várias atividades já ocorrem em paralelo:
  - Um controlador de E/S carrega dados na memória para um processo A enquanto a CPU principal executa um processo B.
  - Dentro da CPU temos o ciclo de recuperar uma instrução, encontrar os operandos na memória, executar a instrução e armazenar o resultado.
    - Qualquer etapa de acesso a memória principal introduz atrasos.
    - O uso de *pipelines* permite a execução de várias instruções ao mesmo tempo, em diferentes etapas e unidades.
- A maioria dos processadores são *multicore*.
- Servidores costumam ter vários processadores.
- Placas gráficas são capazes de executar uma quantidade massiva de *threads* em paralelo.

Motivação

- A tendência é cada vez mais paralelismo:
  - Por limites físicos do silício, não há como aumentar a frequência das CPUs e memórias indefinidamente.
  - Quanto maior a frequência de um processador, maior o consumo energético devido ao calor gerado.
  - As razões acima já levaram ao aprimoramento do *software* para executar em paralelo.
- Outras opções para o progresso da computação, como *computação quântica*, ainda estão em estágios iniciais de desenvolvimento.

AMD Ryzen™ Threadripper™ 3990X

Nº de núcleos de CPU: 64

Nº de threads: 128

Clock básico: 2.9GHz

Clock de Max Boost: Até 4.3GHz

Cachê L1 total: 4MB

Cachê L2 total: 32MB

Cachê L3 total: 256MB

Desbloqueado: Sim

CMOS: TSMC 7nm FinFET

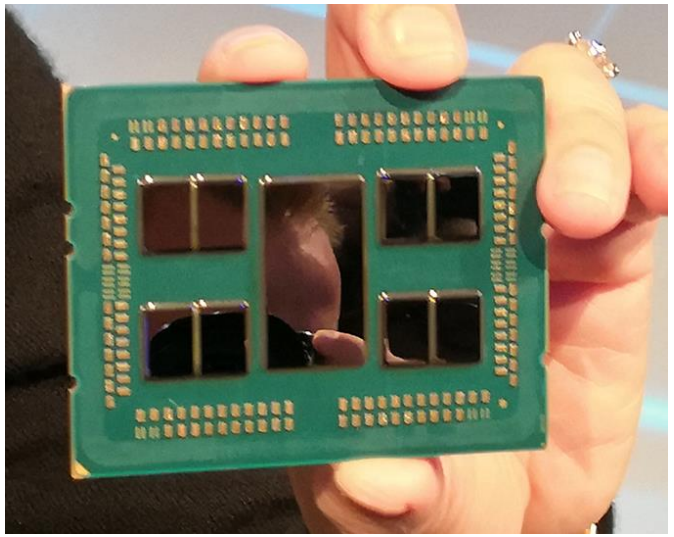
Package: sTRX4

Versão do PCI Express: PCIe 4.0

TDP / TDP Padrão: 280W

Temps máx: 95°C

***Suporte de SO**
Edição Windows 10 - 64-Bit
RHEL x86 64-Bit
*O suporte ao sistema operacional (SO) poderá variar de acordo com o fabricante.



Tipos de Paralelismo

- O paralelismo começou a ser explorado em supercomputadores.
- Com o progresso tecnológico, na atualidade até dispositivos móveis são sistemas paralelos.
- Não importa a escala, os sistemas são classificados em três tipos:
  - Sistemas de memória compartilhada.
  - Sistemas distribuídos.
  - Placas aceleradoras.
- Na prática, um mesmo sistema pode apresentar características de mais de um tipo.
- A classificação é útil para analisarmos os sistemas.

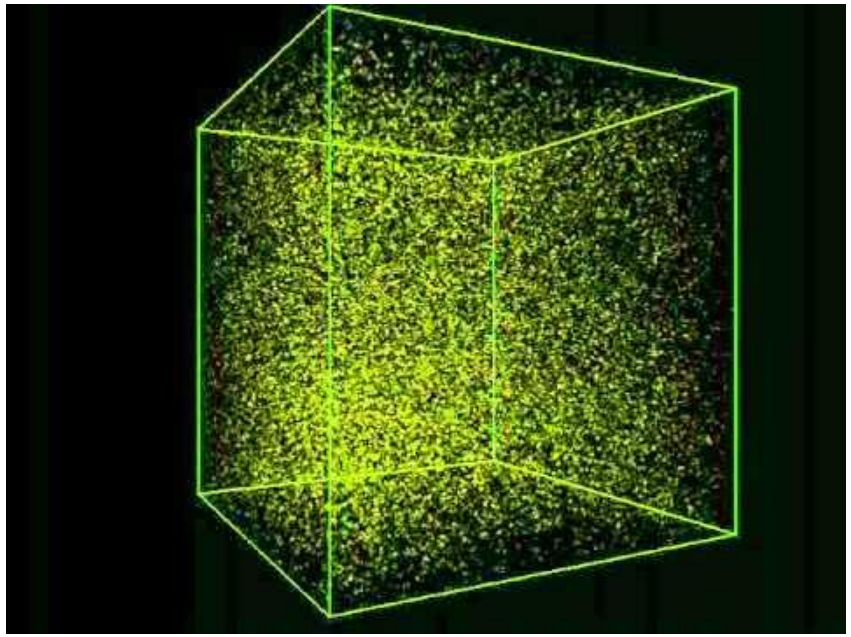
Programação Paralela

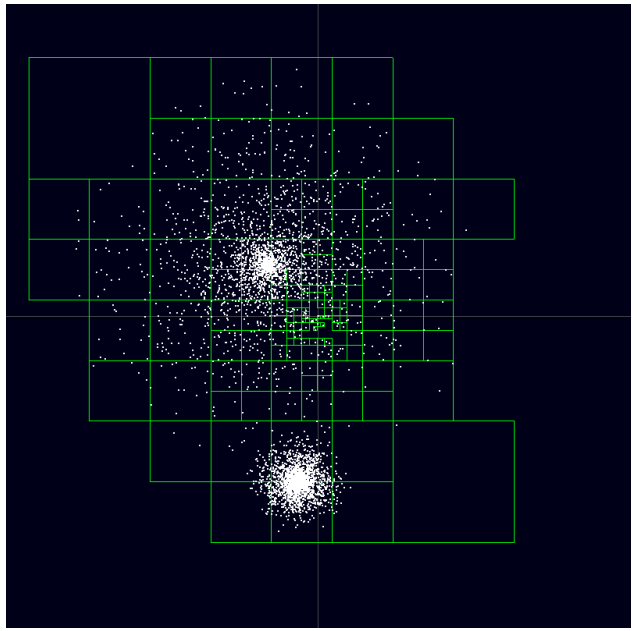
- Apesar da diferença de escala entre os sistemas, os princípios e técnicas de programação paralela são similares.
- O projeto dos algoritmos paralelos é mais desafiador do que soluções seriais.
- Abordagens para programação paralela:
  - *Threads*
  - Troca de mensagens
  - Fluxos (*streams, kernels*)
- Vamos abordar essas soluções no decorrer da disciplina.

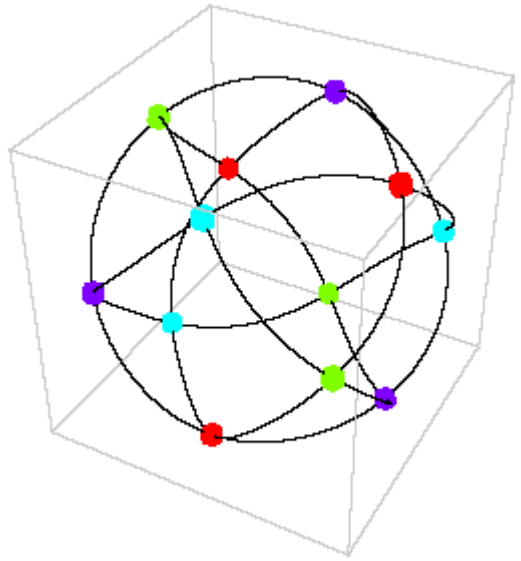
Usando Paralelismos para Resolver Problemas

O problema clássico *n-body*

Dada a posição e o momento de cada membro de um grupo de corpos em um instante inicial, calcular suas posições e velocidades para todas as instâncias futuras.







Usando Paralelismo para Resolver Problemas

- Os objetos se atraem entre si através de forças conhecidas:
  - Versão clássica: gravitação de Newton.
  - Versão moderna: relatividade geral de Einstein.
- Cada objeto n tem dois tipos de equações diferenciais:
  - Posição nos eixos: *(x(t), y(t), z(t))*
  - Momento nos eixos: *($mv_x(t)$, $mv_y(t)$, $mv_z(t)$)*
- A resolução dessas equações conduz a um sistema de equações lineares. Sabemos que:
  - *n = 2*, permite solução analítica.
  - *n > 2,* dependendo da configuração inicial, há soluções analíticas.
  - Na prática, não há solução analítica.
- Precisamos usar métodos numéricos computacionais para chegar a uma solução.

Usando Paralelismo para Resolver Problemas

- Soluções seriais numéricas tem complexidade O(n^2). Já não é das melhores, ainda há um fator oculto que tem impacto não desprezível.
- Algoritmos mais avançados como Barnes-Hut apresentam complexidade O($n \log n$).
- Para valores muitos grandes de n, o tempo necessário em um sistema serial é proibitivo.

Usando Paralelismo para Resolver Problemas

- A solução é desenvolver métodos numéricos para resolver o problema em sistemas paralelos.
- Questões a serem tratadas:
  - Como dividir um método numérico em subtarefas?
  - Em qual processador/núcleo cada subtarefa deve executar?
  - Como cada subtarefa irá colabora com as outras?
- Vamos usar os modelos e *frameworks* de programação paralela para criar código capaz de gerenciar todas as questões acima.
- Para o caso do *n-body*, existe uma solução paralela baseada no Barnes-Hut que suporta até 1000000 elementos, executando de forma ótima em 64 processadores.

Conclusão

- Dúvidas?