

Paralelismo

Advanced Institute for Artificial Intelligence – AI2

<https://advancedinstitute.ai>

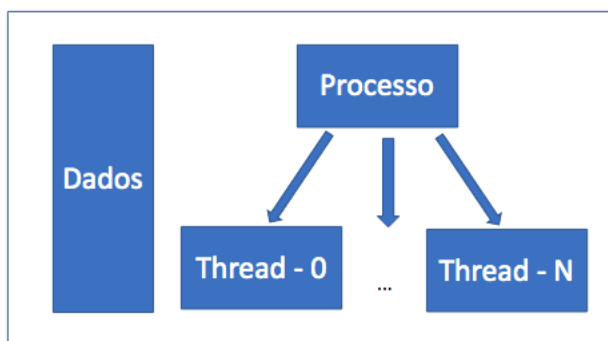
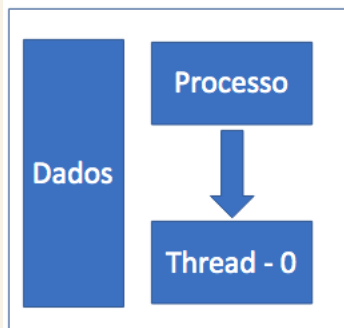
Agenda

- ☐ Paralelismo
- ☐ Thread
- ☐ Níveis de Paralelismo
- ☐ Paralelismo para predição
- ☐ MapReduce - Hadoop
- ☐ Spark

- ❑ Programação concorrente, programação paralela, paralelismo, execução concorrente, programação distribuída, etc
- ❑ Nomes diversos para identificar iniciativas para explorar o uso de mais de um recurso computacional para a mesma aplicação
- ❑ Fundamental para executar aplicações com demandas que não são atendidas pelos recursos fornecidos por um único computador

- ❑ Em um SO qualquer software é identificado como processo A maioria dos SOs permite que um processo crie uma cópia de si mesmo (Thread)
- ❑ Threads são execuções independentes do software que compartilham os mesmos dados da aplicação que a criou
- ❑ Criação de uma thread é muito mais rápido que criar um novo processo
- ❑ Porque criar threads?
 - Atender requisições simultaneas
 - Requisição web
 - Utilizar diversos hardware ligados a um pc

Thread

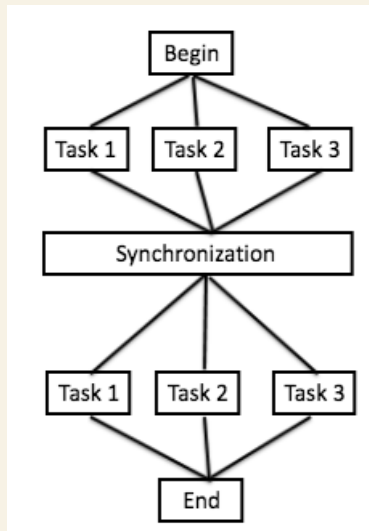


SO – Mapeia Threads para processadores

Hardware – N Processadores

Como explorar esses recursos?

- Para utilizar arquiteturas paralelas é necessário dividir a aplicação em tarefas independentes
 - Para isso podemos dividir o processamento em processos e/ou threads e submeter para execução
 - O SO fará o mapeamento dos processos e threads para os recursos disponíveis
- As tarefas podem possuir algum nível de dependência, de tal forma que, uma tarefa pode necessitar de dados produzidos por outra tarefa antes de entrar em execução
 - Isso deve ser controlado pelo desenvolvedor diretamente



Desafios do paralelismo:

- ☐ No nível do software a paralelização deve considerar as sincronizações entre as tarefas
- ☐ Quanto ao uso do hardware é necessário traçar estratégias para utilizar de modo eficiente cada recurso de otimização de desempenho
- ☐ Obter melhorias de desempenho

Aplicações de Aprendizagem Profunda (Deep Learning) normalmente são computacionalmente intensivas

□ Tarefas computacionalmente intensivas em Deep Learning:

- Busca por hiperparâmetros
- Treino de modelo
- Prototipação de modelos
- Predições em lote

□ Quanto mais rápido uma aplicação de aprendizagem profunda é executada, ainda que com um ganho não tão expressivo, apresenta impacto alto no trabalho dos especialistas desse domínio

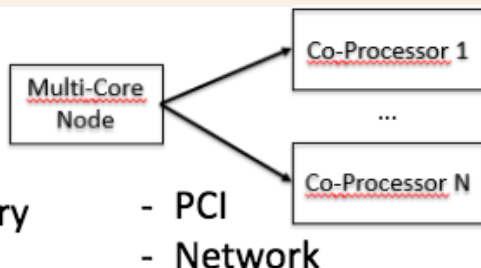
- ❑ Um computador paralelo é um sistema de computador que usa vários elementos de processamento simultaneamente de maneira cooperativa para resolver um problema computacional
- ❑ O processamento paralelo inclui técnicas e tecnologias que permitem calcular em paralelo
 - Hardware, redes, sistemas operacionais, bibliotecas paralelas, linguagens, compiladores, algoritmos, ferramentas,...
- ❑ O paralelismo é natural
 - Problemas de computação diferem em nível / tipo de paralelismo

Arquiteturas computacionais oferecem:

- ❑ múltiplos nós
- ❑ múltiplos processadores
- ❑ múltiplos co-processadores

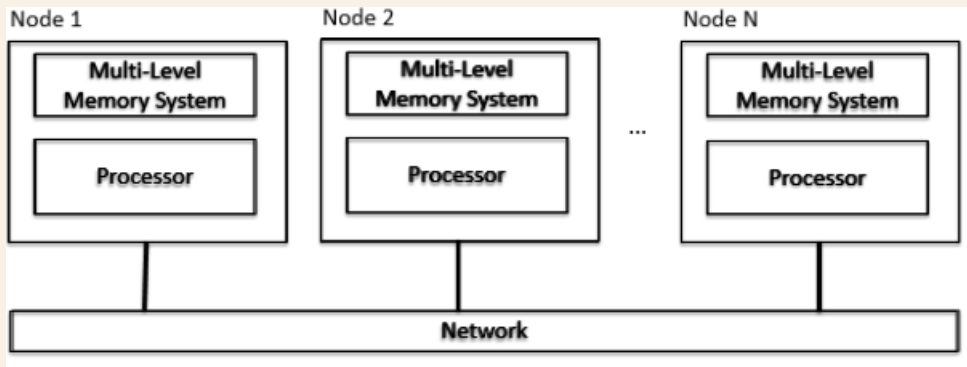
Multi-Core Architecture

- Multi Processors
- Multiple Cores
- **NUMA** (Non-Uniform Memory Access)



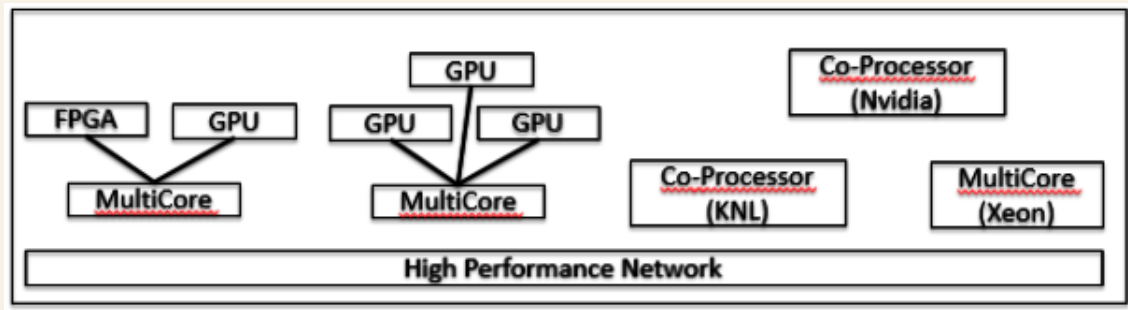
Introdução a HPC

Uma forma de montar uma arquitetura computacional paralela é unificar diversos processadores idênticos



Paralelismo

Outra forma de montar uma arquitetura computacional paralela é unificar diversos processador que podem ser diferentes entre si (Arquiteturas Heterogêneas)



Níveis de paralelismo

☐ Paralelismo no nível de instrução

- Mecanismos do processador para aumentar o desempenho

☐ Paralelismo no nível de dados(Vetorização)

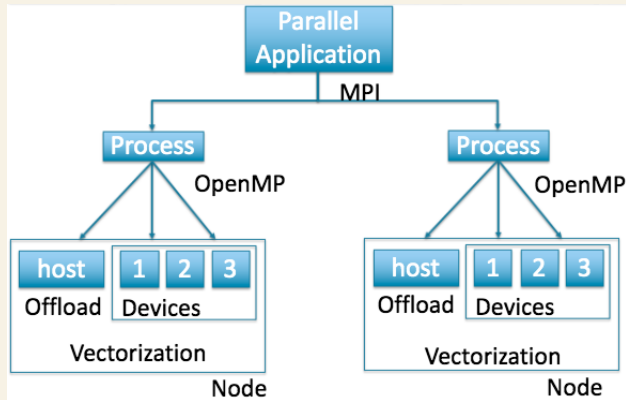
- Execução de mais de instrução idênticas em paralelo para conjuntos distintos de dados, utilizando registradores vetoriais do processador
- Vetorização em geral é combinada com técnicas de acesso eficiente utilizando múltiplos níveis de memória (cache)

- ❑ Paralelismo no nível de tarefa (Thread) OpenMP
 - Múltiplas threads sendo executadas em paralelo
- ❑ Paralelismo no nível de processo
 - Mesmo programa em diferentes computadores coordenado a execução por troca de mensagens pela rede (MPI)
- ❑ Offload
 - Dividindo o processamento entre o processador e um ou mais co-processadores, como GPU por exemplo

Níveis de Paralelismo

Os níveis de paralelismos podem ser usados de modo combinado por uma única aplicação

O paralelismo pode ser controlado por recursos elementares ou por bibliotecas de alto nível



Barreiras para explorar paralelismo em python

- ☐ Python utiliza um componente chamado GIL (Global Interpreter Lock) para proteger chamadas simultaneas a um mesmo objeto na memória
- ☐ Embora evite diversos problemas durante a execução, que podem não ser facilmente detectados apenas olhando o código. O GIL inibe o uso de threads
- ☐ Na prática uma programa multi thread em python executa sequencialmente

Barreiras para explorar paralelismo em python

- ☐ Uma estratégia possível é utilizar multiprocessing, no lugar de multi threading
 - Custo de criação e destruição de processo pode ser alto
- ☐ Realizar chamadas a bibliotecas externas que por si executando em múltiplas threads
- ☐ Implementar o código multi-thread em C e linkar com código python usando cython

Explorando paralelismo para aprendizagem de máquina

- Ferramentas de aprendizagem de máquina em geral exploram paralelismo, em todos os níveis apresentados
- Ambientes de nuvem podem ser explorados para delegar parte da carga de trabalho
- Escolha de recursos adequados para execução de cada modelo, também pode ser explorado
 - Exemplo, utilizar recursos com GPU para treino de redes neurais

Frameworks como Spark, Hadoop entre outros podem ser usados para executar o processo completo de aprendizagem de máquina de modo eficiente

- ❑ Explorando paralelismo, co-processadores, execução em ambiente de nuvem
- ❑ Esses frameworks exploram
 - Execução de modelos
 - Processamento dos dados de modo distribuído
 - Uso de memória RAM para inibir atrasos por transferência em disco

Modelo de paralelismo desenvolvido pela Google para processamento de grandes volumes de dados ¹

- ☐ Baseia-se na idéia de usar apenas duas funções para facilitar a execução paralela
- ☐ Modelo mais rígido de paralelismo, mas facilita a otimização de desempenho
- ☐ A implementação é proprietária mas existe uma implementação pública do modelo Hadoop ²

¹Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: simplified data processing on large clusters. In Proceedings of the 6th conference on Symposium on Operating Systems Design Implementation

²<http://hadoop.apache.org>

- ❑ Essa abordagem permiteo mapear fragmentos de dados de entrada a uma chave identificadora, e entao processar todos os fragmentos que compartilhem a mesma chave.
- ❑ Se a quantidade de dados for grande, pode ser dividido para a execucao de diversas funcoes Map ao mesmo tempo, em paralelo.
- ❑ Podemos aplicar separadamente as funcoes Map e Reduce a um conjunto de dados.
- ❑ Um sistema de arquivos especial chamado HDFS permite mapear diferentes partes de uma mesmo arquivo em diferentes sistemas de memória volátil em diversas unidades de processamento

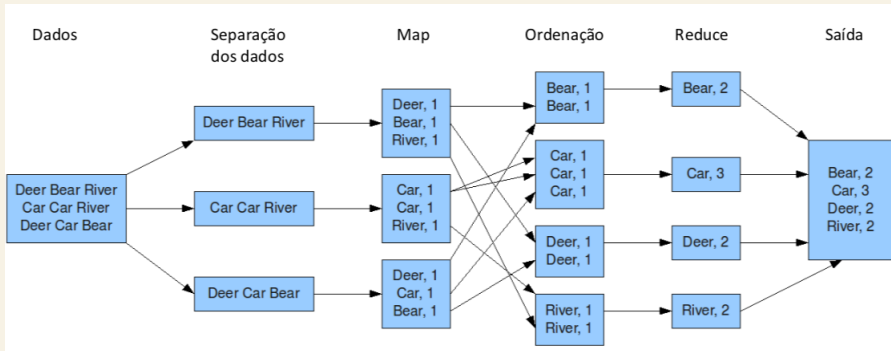
Estratégia

- ❑ A função Map recebe um conjunto de dados e retorna um conjunto chave-valor
- ❑ A função reduce recebe esse conjunto chave-valor, aplica uma operação e retorna outro conjunto chave-valor
- ❑ A ideia é que tanto o map quanto o reduce sejam operações simples que possam ser paralelizadas

Um exemplo clássico de uso do Hadoop é a aplicação para contar palavras únicas em um arquivo de texto

- ❑ A função Map lê cada linha e gera uma tupla palavra,1
- ❑ A função reduce lê todas as tuplas e cria novas tuplas com o somatório de cada palavra encontrada
- ❑ O map a medida que lê o texto já gera as tuplas
- ❑ O reduce a medida que lê as tuplas, mesmo que diferentes nós, consegue fazer a soma
 - No caso do reduce, a medida que as somas são realizadas para cada nó, é possível acumular hierarquicamente

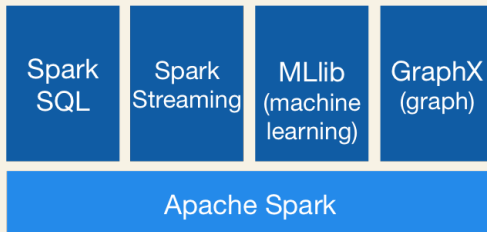
A estratégia é simples, porém muito complexa de implementar para outros casos
Outros modelos surgiram com o objetivo de facilitar o uso desse modelo



Apache Spark ³ é uma ferramenta de Big Data

Tem como objetivo processar grandes volumes de dados em clusters de computadores

Baseado em Map Reduce



³<https://spark.apache.org/>

O recurso básico do Spark é o RDD (Resilient Distributed Datasets)

- ☐ Unidade fundamental de dados em Spark
- ☐ Resiliente: se dados na memória são perdidos, podem ser recriados
- ☐ Distribuído: armazenados na memória por todo o cluster
- ☐ Datasets: dados iniciais podem vir de um arquivo ou ser criado programaticamente
- ☐ Muitos programas Spark se baseiam na manipulação de RDDs

MLlib

- ☐ Módulo que implementa algoritmos de machine learning e recursos para manipular dataframe
- ☐ Similar ao Scikit-learn, oferece várias implementações e permite testar vários algoritmos com o mesmo dataframe
- ☐ Otimizado para desempenho e paralelismo em cluster