Análise e Otimização do *framework* Apache Hadoop em uma Arquitetura Paralela com Memória Compartilhada

Eder J. Scheid, Paulo V. Cardoso, Andrea S. Charão
Universidade Federal de Santa Maria

Introdução

Crescimento exponencial dos dados

- Big data;
- Necessidade de um poder computacional capaz de processar esta quantidade de dados também foi crescendo.

Surgem os clusters de computadores.

Apache Hadoop

Desenvolvido pela Yahoo!; Serviu de apoio para outro projeto, chamado Nutch (sistema de busca);

Baseado no paradigma MapReduce;

Arquitetura consiste no sistema de arquivos (HDFS) e na gerência das aplicações (YARN)

Arquitetura NUMA

NUMA significa **non-uniform memory access**, ou seja, o tempo de acesso a memória depende da **distância** na qual ela está do processador.

Cada processador tem sua própria memória.

Acesso a memória dos outros processadores tendem a demorar um pouco mais devido à **latência** do acesso remoto.

Vantagens:

I. Dados em apenas uma máquina
II. Acesso à memória RAM (evita concorrência)
III. Escalabilidade (adição de nós)

SGI UV2000

Configuração da máquina usada:

- 8 nós NUMA interligados;
- Cada nó com 6 cores;
- 62 GB de RAM por nó;
- 1 *terabyte* de armazenamento.

Motivação

Como utilizar, então, uma tecnologia projetada para rodar em clusters, como o framework Apache Hadoop, em apenas **uma** máquina?

Objetivo

Análise do comportamento do Apache Hadoop em um ambiente onde o mesmo não foi projetado:

uma máquina com memória compartilhada.

Desenvolvimento inicial

Algoritmo Usado: PiEstimator

- Foco no teste de processamento;
- Recebe dois argumentos: número de maps e número de pontos;

Desenvolvimento inicial

Configurações do Hadoop

Hadoop em modo *standalone:*

- Apenas 1 processo Java
- HDFS e YARN não usados;

Hadoop em modo *pseudo-distribuído:*

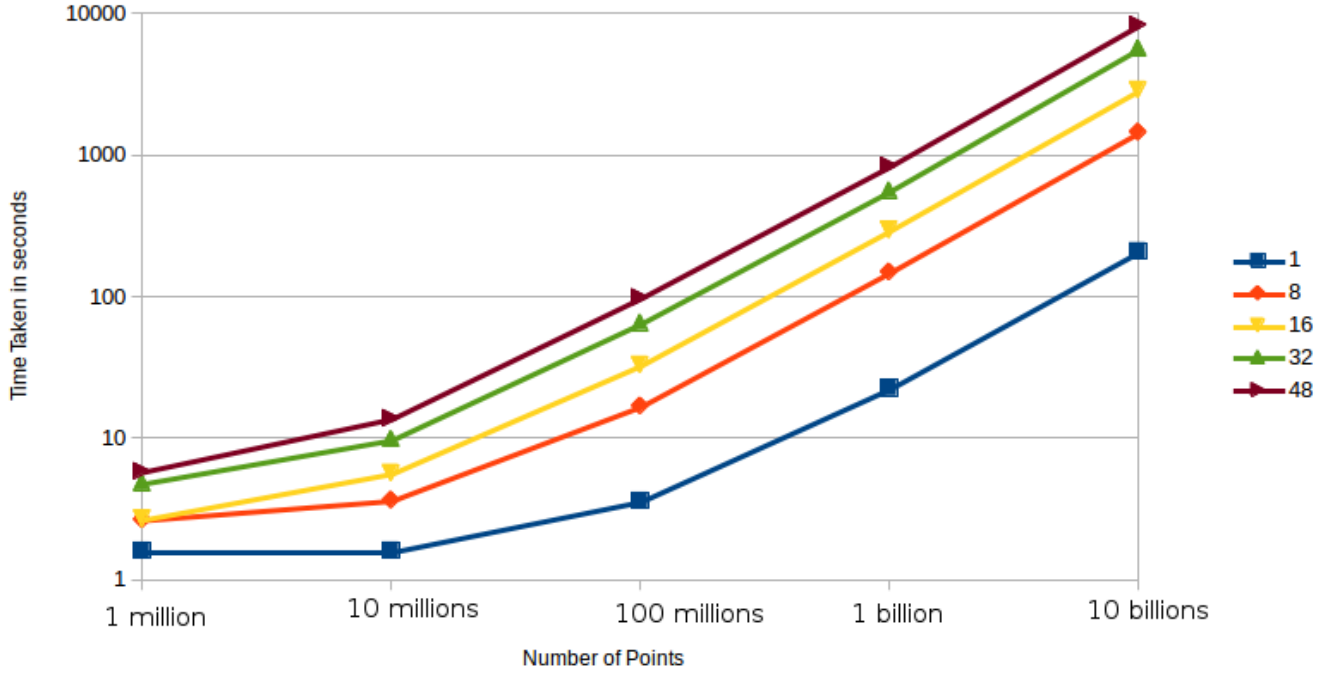
- Simula um comportamento distribuído;
- Mais de 1 processo Java.

Desenvolvimento inicial

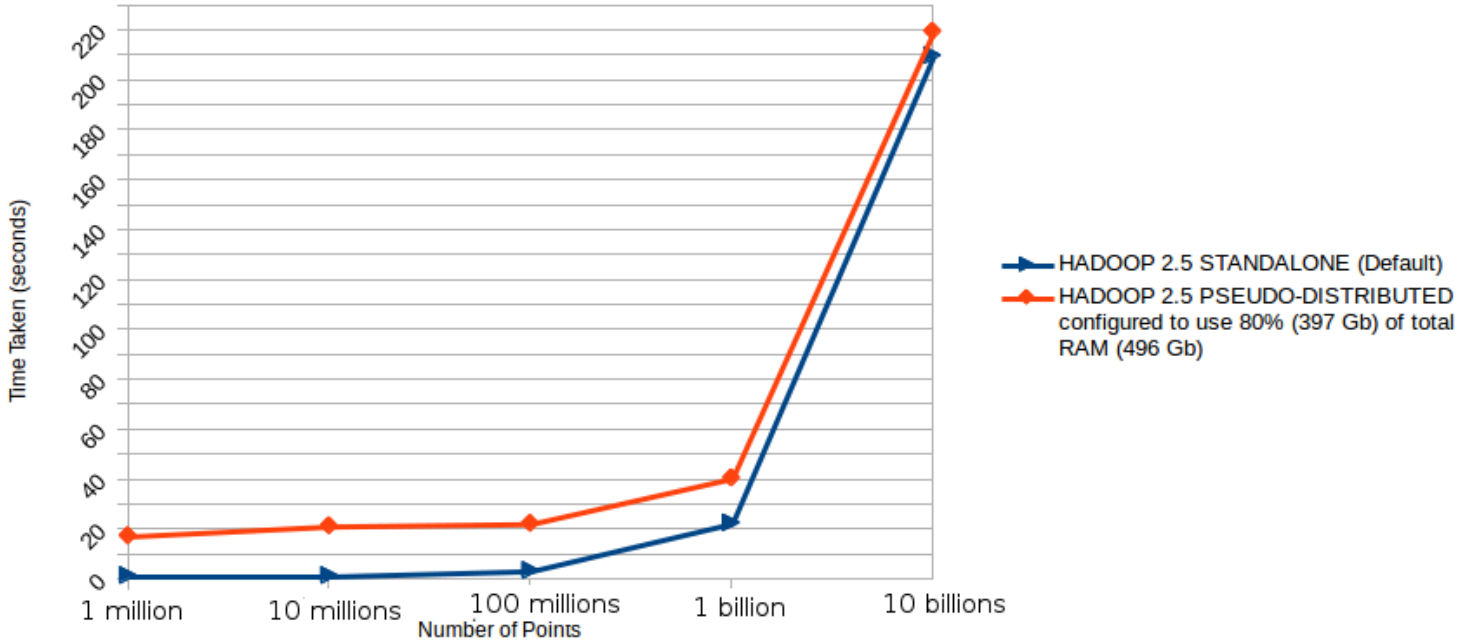
Configuração para 3 *containers* com 132 GB de RAM

Propriedade	Valor
yarn.scheduler.minimum-allocation-mb	135168
yarn.scheduler.maximum-allocation-mb	405504
yarn.nodemanager.resource.memory-mb	405504
mapreduce.map.memory.mb	135168
mapreduce.map.java.opts	-Xmx108134m
mapreduce.reduce.memory.mb	135168
mapreduce.reduce.java.opts	-Xmx108134m
yarn.app.mapreduce.am.resource.mb	135168
yarn.app.mapreduce.am.command-opts	-Xmx108134m
mapreduce.task.io.sort.mb	54067

Hadoop em modo standalone (PiEstimator)



Comparação entre os modos standalone e pseudo-distribuído



Resultados iniciais

- Standalone vs Pseudo-distribuído:
  - Modo standalone mais rápido, mas não paralelo;
  - Modo pseudo-distribuído gerencia mais componentes do Hadoop.
- Possíveis problemas:
  - Parâmetros gerados pelo script;
  - Configuração não usando todos os cores;
  - Revisão do cálculo do número de containers;

Otimização do Hadoop

Experimentos com os escalonadores: **CapacityScheduler** e **FairScheduler**

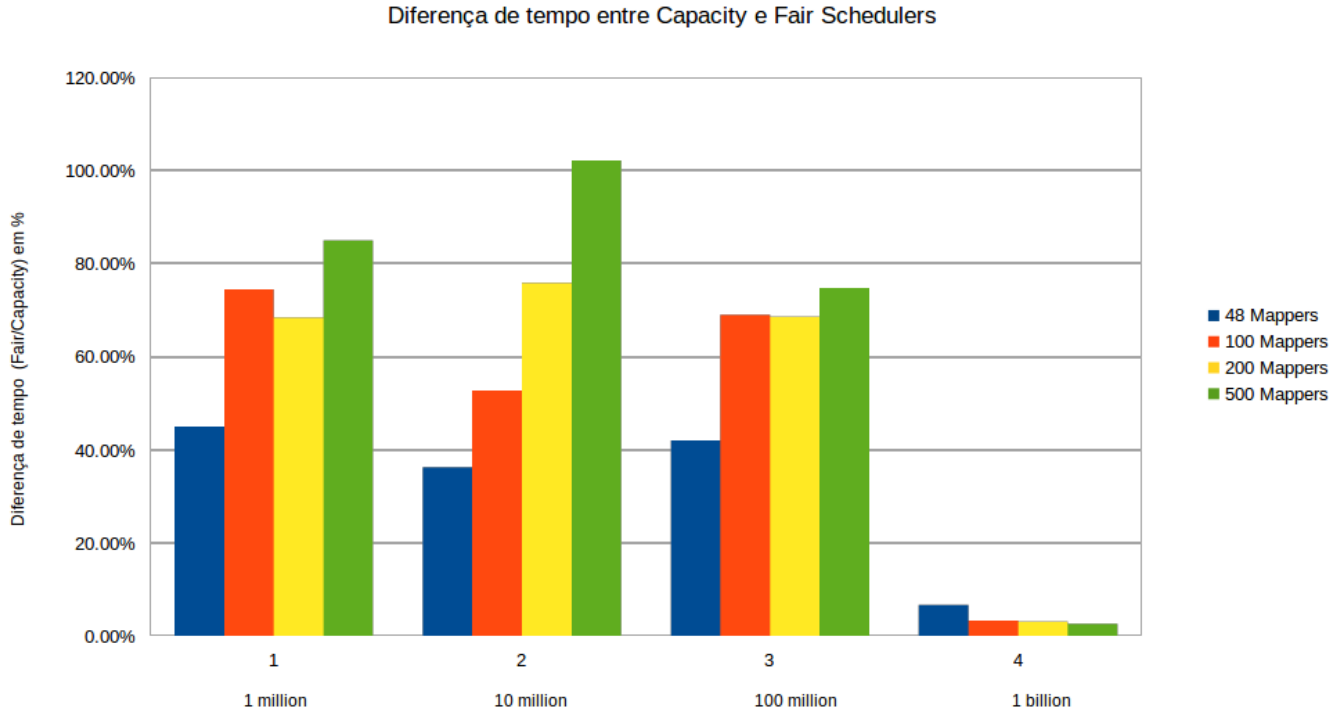
Análise do uso de cores da máquina

- Número de containers e memória alocada.

Escalonadores

- CapacityScheduler
  - Cada organização recebe uma capacidade mínima, mas não máxima;
  - Recursos de outras organizações podem ser realocados;
- FairScheduler
  - Todos os *jobs* recebem em média, uma parte igualitária dos recursos;

Desenvolvimento com configurações otimizadas



Desenvolvimento com configurações otimizadas

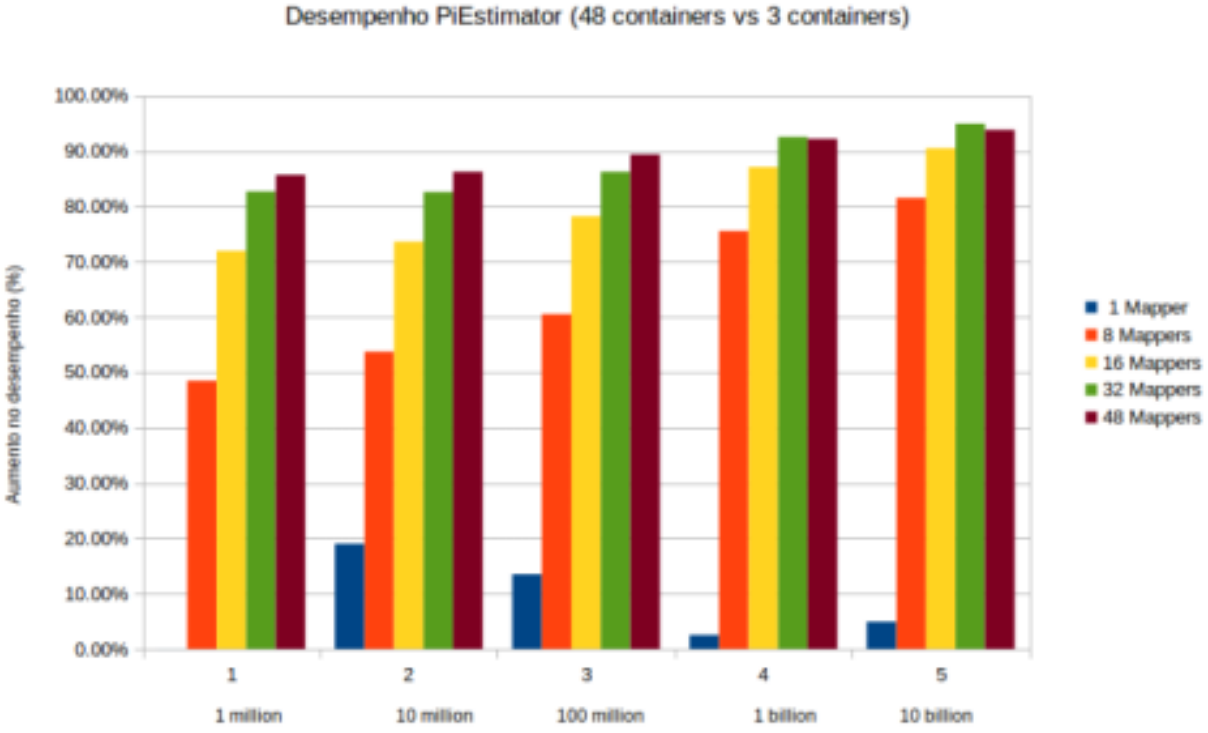
Número de containers e memória alocada:

- ➔ Máximo de 48 containers (número de *cores);*
- ➔ Containers de 10 GB;
- ➔ 480 GB total, 16 GB para o S.O;

Desenvolvimento com configurações otimizadas

Algoritmo PiEstimator: testes realizados nas mesmas condições dos experimentos iniciais

Resultados Finais



Resultados Finais

Redução do tempo de execução em mais de 90% em alguns casos:

- 4932.161 segundos (1h 30min) para 249.814 segundos (4 minutos).

Impacto maior em quantidades maiores de mappers.

Conclusão

Nova configuração melhorou o desempenho do Apache Hadoop no caso proposto;

Diversos parâmetros devem ser levados em consideração;

Desempenho depende da aplicação a ser executada;

Análise e Otimização do *framework* Apache Hadoop em uma Arquitetura Paralela com Memória Compartilhada

Eder J. Scheid, Paulo V. Cardoso, Andrea S. Charão
Universidade Federal de Santa Maria