Hadoop e MongoDB

Fernando Felix do Nascimento Junior

Sumário

- Hadoop
- MongoDB
- Integração
- Como
- Exemplo
- Resultados
- Conclusão
- Referências

Hadoop

Hadoop

Framework de código aberto, implementado em Java e utilizado para o processamento e armazenamento em larga escala, para alta demanda de dados, utilizando máquinas comuns.

Vantagens

- Código aberto: todo projeto aberto de sucesso tem uma comunidade ativa
- Economia: processamento de dados utilizando máquinas e rede convencionais
- Robustez: recuperação automática de falhas
- Escalabilidade: Relativamente simples, mudanças no ambiente implicam em pequenas modificações na configuração
- Simplicidade: retira do desenvolvedor a responsabilidade de gerenciar questões relativas à computação paralela

Desvantagens

- Único nó mestre
- Dificuldade depurar a execução da aplicação e de analisar os logs
- Problemas não paralelizáveis ou com grande dependência entre os dados
- Processamento de arquivos pequenos (overhead)
- Problemas com muito processamento em poucos dados

Aplicações

- Data Warehouse
- Business Intelligence
- Aplicações analíticas
- Mídias Sociais
- Sugestão de Compras
- Analise preditiva
- Compras Coletivas
- Recomendações

Componentes

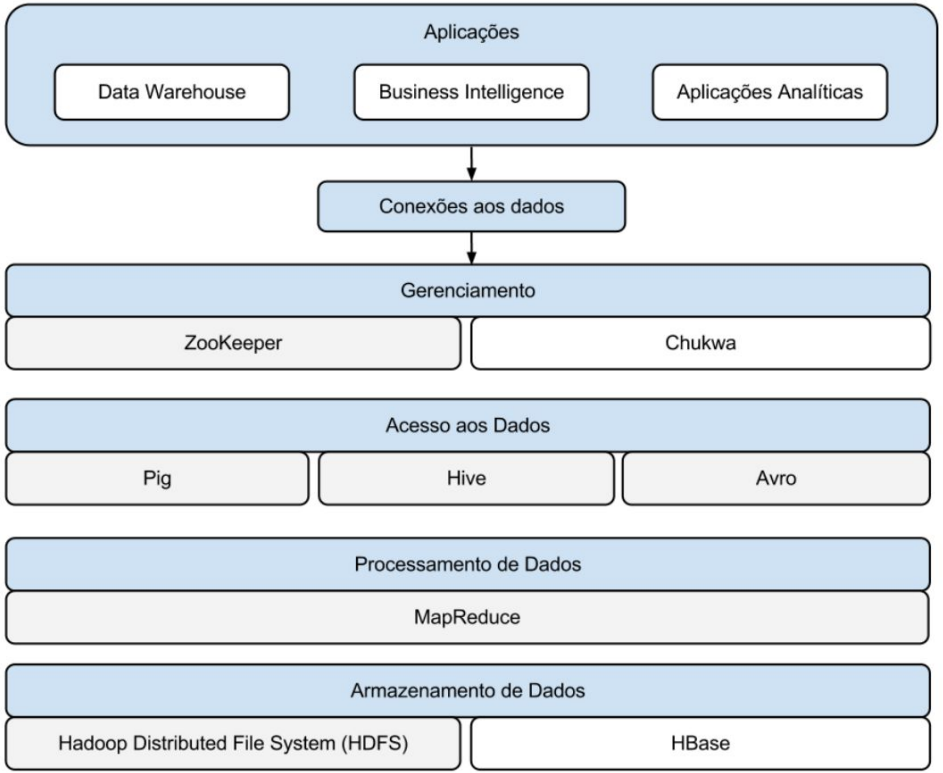
Principais:

- Hadoop Distributed File System
- MapReduce: estrutura de programação

Outros:

- Avro, Chukwa, Hbase, Hive, Pig, ZooKeeper

Componentes

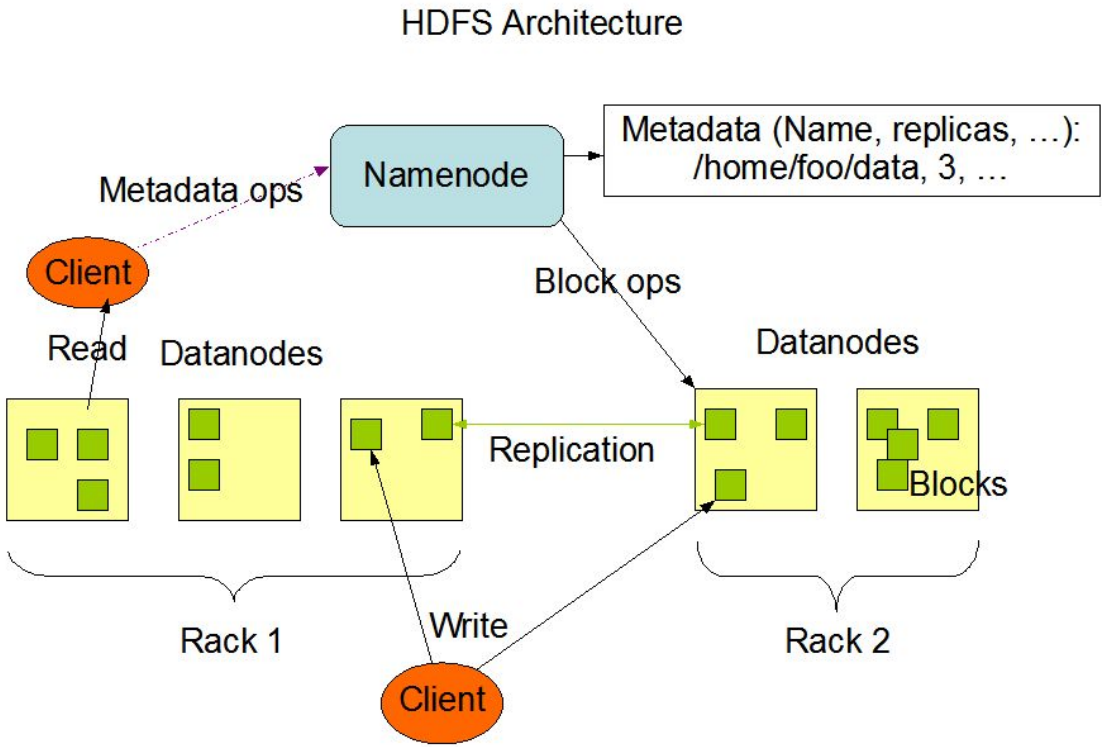


HDFS

O Hadoop Distributed File System (HDFS)

- Sistema de arquivos distribuído integrado ao Hadoop
- Suporte ao armazenamento e ao processamento de grandes volumes de dados em um agrupamento de computadores heterogêneos
- Quanto mais máquinas, maior a chance de acontecer algum erro, por isso promove tolerância, detecção e recuperação automática de falhas
    - Um processamento em uma máquina pode ser reiniciado, ou no pior caso, repassado para uma outra máquina disponível
    - Transparente ao usuário

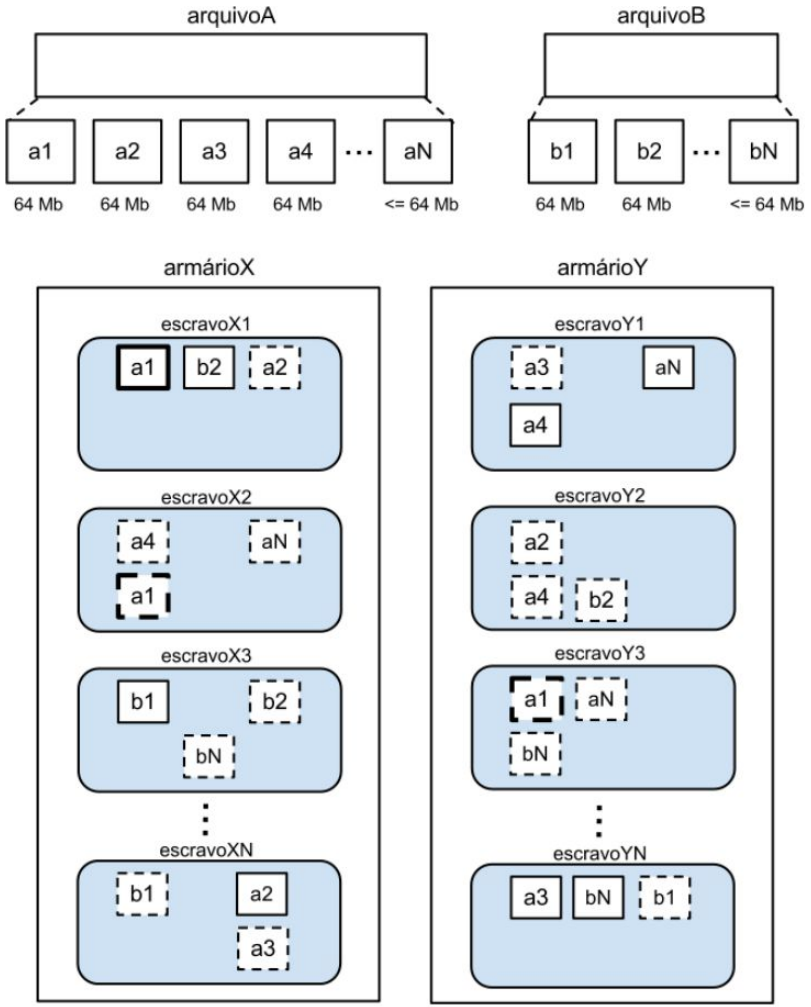
HDFS


HDFS Architecture
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Namenode
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

HDFS

Além de dividir os arquivos em blocos, o HDFS ainda replica esses blocos na tentativa de aumentar a segurança

Benefício: tolerância a falhas, confiabilidade dos dados, não existe necessidade de transferência de dados nem interrupção da execução da aplicação.



MapReduce

O paradigma de programação MapReduce implementado pelo Hadoop se inspira em duas funções simples (Map e Reduce) presentes em diversas linguagens de programação funcionais.

- Processa grandes volumes de dados em paralelo
- Divide o trabalho em um conjunto de tarefas independentes
- Esconde detalhes de implementação do desenvolvedor
    - Divide os dados, executa vários mappers sobre as divisões, embaralha os dados para os redutores, executa vários redutores, guarda os resultados finais

MapReduce

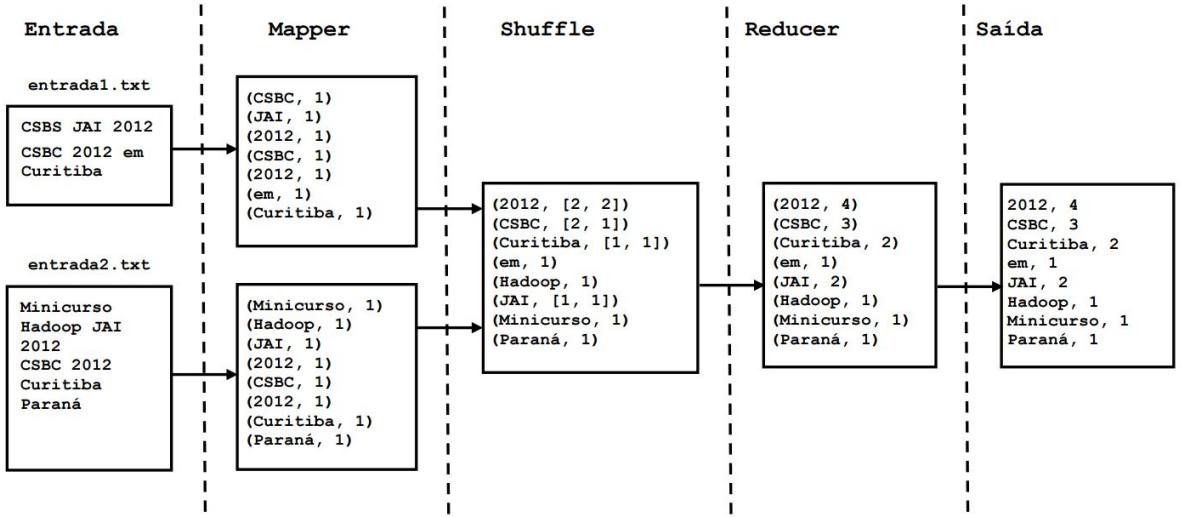
Escalonador de tarefas:

- Escalonador padrão
  - Fila simples com prioridade sem preenpção
- Escalonador justo
  - Parcela igual dos recursos ao longo do tempo. Se há apenas um trabalho em execução no aglomerado, todos os recursos disponíveis podem ser alocados para o mesmo
- Escalonador de capacidade
  - Permite que cada empresa com acesso ao cluster tenha uma garantia mínima de capacidade de execução (tempo de processamento). Proporciona também elasticidade
- Escalonador de demanda
  - Usa um gerenciador de recursos para fazer a alocação de nós e neles inicializa o HDFS e executa as aplicações MapReduce quando solicitado

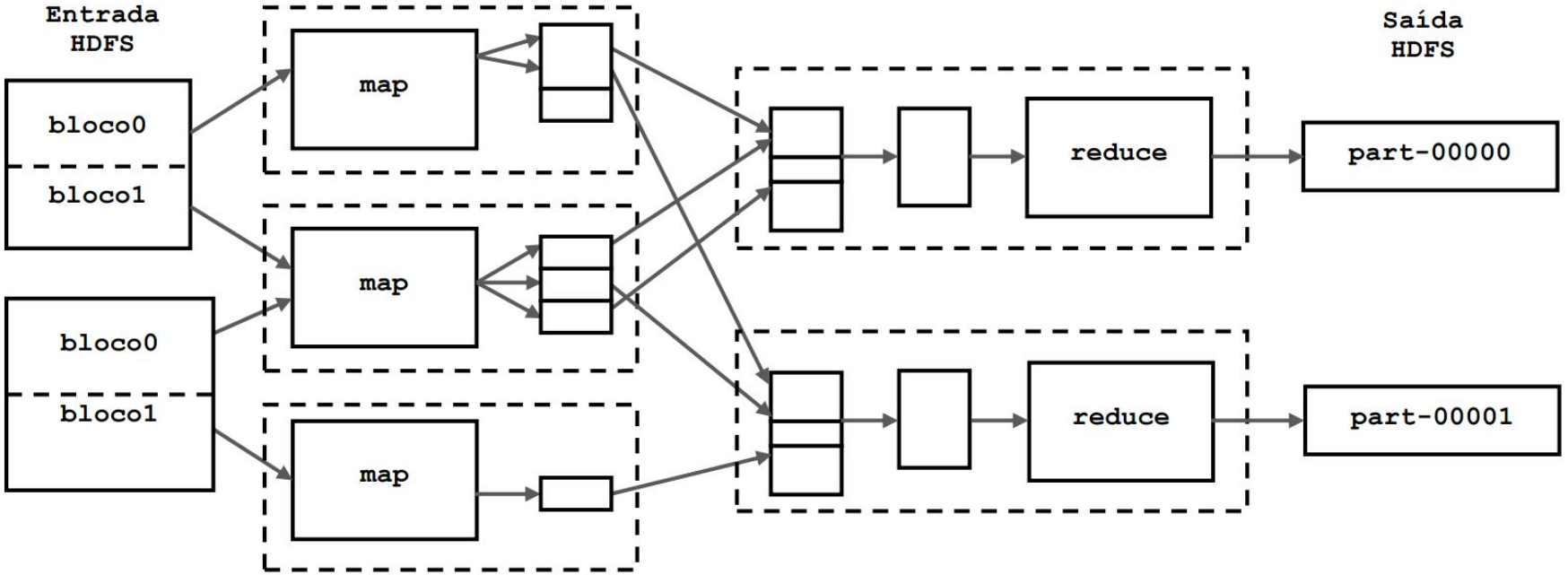
MapReduce

```
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1");

reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result));
```



MapReduce


Entrada
HDFS
bloco0
bloco1
bloco0
bloco1
map
map
map
reduce
reduce
Saída
HDFS
part-00000
part-00001

MongoDB

MongoDB

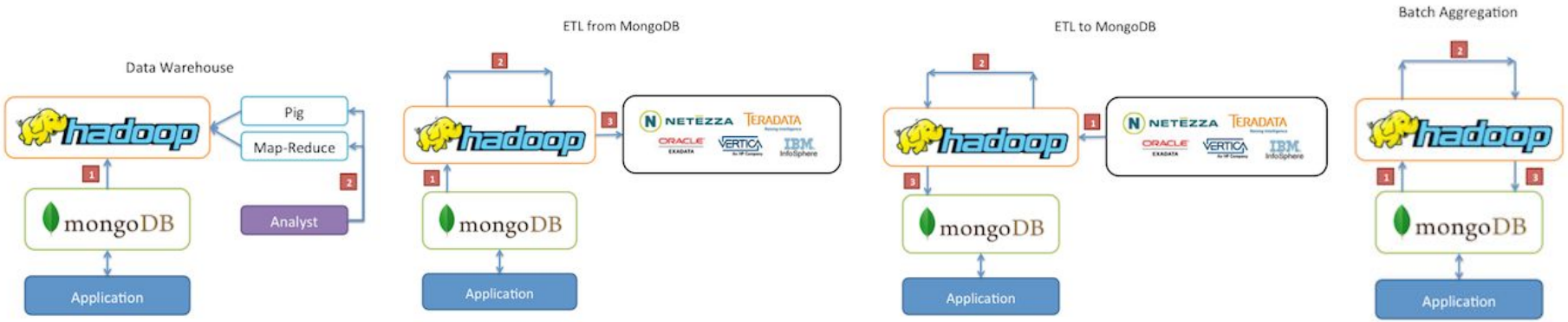
- Banco de dados orientado a documentos
- **Alto desempenho (escrita), alta disponibilidade (replicação) e escalabilidade automática (divisão)**
- Integração de dados em certos tipos de aplicações mais fácil e rápido

RDBMS	**MongoDB**
Database	Database
Table	Collection
Row	Document (binary + JSON)
Index	Index
Join	Embedded Document
Foreign Key	Reference

Integração

Integração

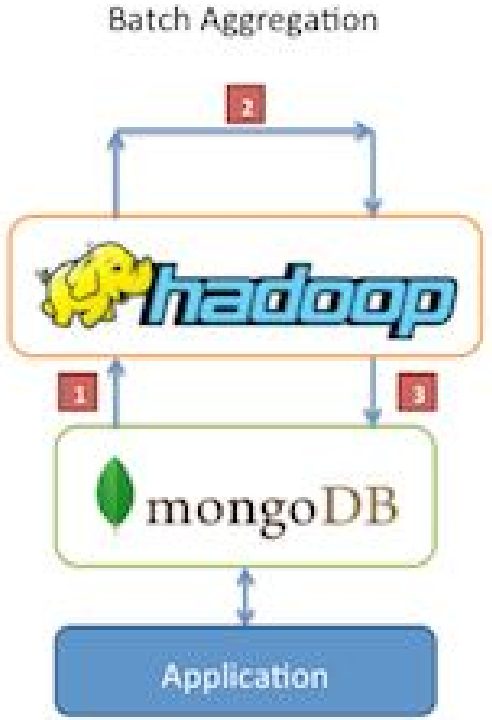
O Hadoop e o MongoDB podem se integrar de várias maneiras para fornecer soluções big data típicas.



Integração

Agregação:

- Em vários cenários, a funcionalidade de agregação fornecida pelo próprio MongoDB é suficiente para analisar dados
- Agregações de dados significativamente mais complexas podem ser necessárias
- O Hadoop pode fornecer uma poderosa estrutura para análises complexas



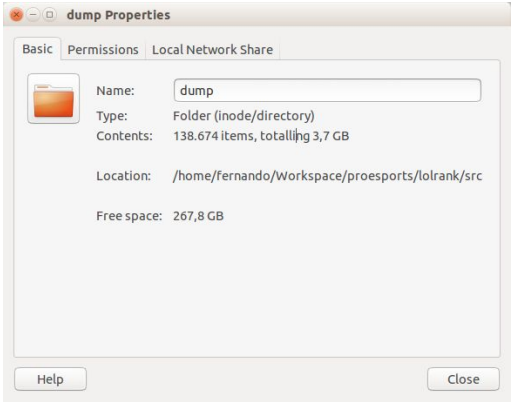
Como

Como

- A tutorial to use MongoDB with Hadoop
  - https://github.com/fernandojunior/mongo-hadoop-tutorial
- Instalação do hadoop (2.6)
  - https://gist.github.com/fernandojunior/35d3b17e4fbd4665f4be
- Instalação do MongoDB (3.0.7)
  - https://gist.github.com/fernandojunior/8c7bfc720e51f69645da
- Integração
  - https://gist.github.com/fernandojunior/1231cba3325cdca6239d

Exemplo

Exemplo



- League of Legends
  - Jogo competitivo 5x5
  - API para acessar dados das partidas (JSON)
- Dados
  - Dados de 1 dia, das 8h às 22h
    - +138 mil partidas, +130 milhões participantes
    - 3,7GB (JSONs)
  - Filtro partidas e atributos: +23 mil partidas, +23 milhões participantes
    - 26MB (CSV)
- Problema
  - Qual a quantidade de participantes por hora? Qual a quantidade de abates por hora? Qual a a média por hora?
- Experimento
  - 4 nós: 1 mestre e 4 escravos
  - Máquina virtual: 32 bit, 1 core, 1GB ram, Ubuntu Server

Resultados

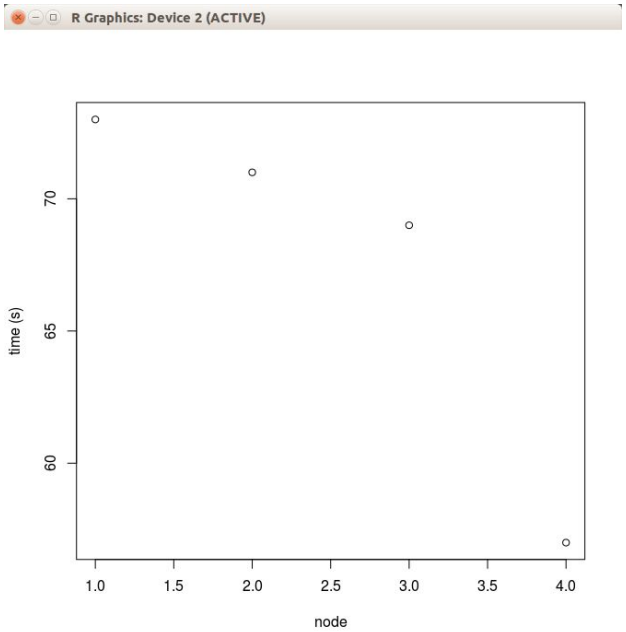
Resultados

Experimento

```
nodes, t1, t2, t3, t4, avg
1, 1m12s, 1m27s, 58s, 1m15s, 73s
2, 1m24s, 1m24s, 1m1s, 58s, 71s
3, 1m29s, 1m5s, 57s, 1m5s, 69s
4, 56s, 1m1s, 57s, 56s, 57s
```

Saída do MapReduce


```
{"_id":8,  "count":2840,  "avg":6.224, "sum":17678}
{"_id":9,  "count":9700,  "avg":6.320, "sum":61312}
{"_id":10, "count":21180, "avg":6.204, "sum":131416}
{"_id":11, "count":25740, "avg":6.306, "sum":162321}
{"_id":12, "count":31500, "avg":6.309, "sum":198756}
{"_id":13, "count":38440, "avg":6.339, "sum":243692}
{"_id":14, "count":26560, "avg":5.943, "sum":157866}
{"_id":15, "count":36710, "avg":5.981, "sum":219580}
{"_id":16, "count":13390, "avg":6.223, "sum":83339}
{"_id":19, "count":10000, "avg":6.348, "sum":63993}
{"_id":20, "count":20290, "avg":6.348, "sum":128803}
```

Conclusão

Conclusões

- Melhoria do desempenho conforme aumento de nós
- Transparente (replicação, recuperação, etc.)
- Facilmente escalável
- Funciona em computadores heterogêneos e convencionais
- Integração:
  - Permite trabalhar com dados JSON
  - Melhoria de performance do mongoDB

Referências

- Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. http://www.ime.usp.br/~ipolato/JAI2012-Hadoop.pdf
- Introduction to MongoDB. https://docs.mongodb.org
- MongoDB. Disponível em https://en.wikipedia.org/wiki/MongoDB
- Hadoop and MongoDB Use Cases. https://docs.mongodb.org/ecosystem/use-cases/hadoop/

Dúvidas?

fernandofelix@copin.ufcg.edu.br

https://br.linkedin.com/in/fernandofnjr