# Uma Estratégia de Alocação Dinâmica e Preditiva de Recursos de I/O para Reduzir o Tempo de Execução em Aplicações *MapReduce*

**Kassiano Matteussi**[1], **César De Rose** [1]

[1] Programa de Pós Graduação em Ciência da Computação - PPGCC
Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS
Caixa Postal 1429 – 90619-900 – Porto Alegre – RS – Brazil
kassiano.matteussi@acad.pucrs.br, cesar.derose@pucrs.br

***Resumo.*** *Aplicações Big Data utilizam MapReduce (MR) para analisar dados em larga escala na nuvem. Em aplicações MR, fases de* map *influenciam significativamente no tempo total de execução de uma aplicação. Pois, terminam em tempos alternados devido a contenção de recursos de I/O. Neste contexto, este trabalho propõe uma estratégia para alocação dinâmica e preditiva de recursos de I/O para diminuir o tempo de execução de aplicações MR.*

## 1. Introdução

*MapReduce* (MR) é um modelo de programação distribuído utilizado em larga escala por grandes corporações, como *Facebook*, *Yahoo* e *Twitter*. MR tem como foco a análise e processamento massivo de dados. Cada *task* de uma aplicação MR é executada essencialmente por duas fases, *map* e *reduce*. Ambas sendo interligadas pela fase de comunicação e sincronização, denominada *shuffle*. Para que a fase de *reduce* possa ser iniciada, os dados intermediários produzidos por todas as fases de *map* devem estar disponíveis para uso, ou seja, consolidados em disco. Um dos principais problemas de desempenho encontrado em modelos MR são as chamadas fases *map* de longa duração, muitas vezes um pequeno número dessas fases pode influenciar de forma significativa no tempo de conclusão de todas as outras, pois tendem a terminar em tempos alternados, bloqueando o progresso de uma *task* MR e prejudicando o tempo total de execução de uma aplicação MR. Além disso, outra implicação é a subutilização de recursos do *cluster*, uma vez que o mesmo pode ficar em grande parte ocioso durante o processamento de fases *map* de longa duração. Por meio do YARN é possível efetuar o agendamento de trabalhos, alocação e gestão de recursos de forma dinâmica e isolada utilizando *containers*. Isso permite o compartilhamento dinâmico de recursos em um *cluster* dentre uma variedade de estruturas de processamento, incluindo MR, *Impala*, e *Spark*.

Atualmente, a gestão de recursos do YARN limita-se a alocação de memória e CPU, não efetuando nenhum controle sobre as operações de disco I/O [Apache 2014, Ryza 2014]. Esse fator contribui de forma significativa para que a fase de *map* leve um maior tempo de execução e que termine em um tempo diferente, sendo que para o encerramento da mesma, é necessário consolidar os dados intermediários em disco. Esse problema também ocorre de forma semelhante na fase de *reduce*. Dessa maneira, o presente trabalho tem como objetivo propor uma estratégia para alocação dinâmica e preditiva de recursos de I/O para diminuir o tempo de execução de aplicações MR. A proposta a ser mencionada aqui, leva em conta somente a fase de *map* de uma *task* MR.

## 2. Proposta e Perspectivas

Como o YARN não oferece suporte para restrição de recursos de I/O e fases *map* de longa duração utilizam intensivamente esse tipo recurso, o presente trabalho propõe uma es-

tratégia para alocação dinâmica e preditiva de recursos de I/O para diminuir o tempo de execução de aplicações MR. Para proporcionar a alocação e gestão de recursos o *Kernel* do *Linux* fornece uma funcionalidade chamada cgroups (grupos de controle). Com cgroups pode-se alocar um *container* YARN em um grupo com recursos controlados. Dessa maneira, se os recursos de um nó ou de um *container* estiverem ociosos, é possível que outros *containers* possam utilizar esses recursos. Frente a essas possibilidades, a presente proposta utiliza uma função preditiva[1] de I/O [Neves et al. 2014][2] que faz uso de *logs* gerados no Hadoop para prever a quantidade de dados a serem gravadas no próximo instante de tempo e, por fim, efetua-se a melhor alocação de recursos de I/O com base no cenário atual, ou seja, atribuir ou remover recursos de I/O para cada *container* mantendo as prioridades sobre as aplicações e diferentes *workloads*.

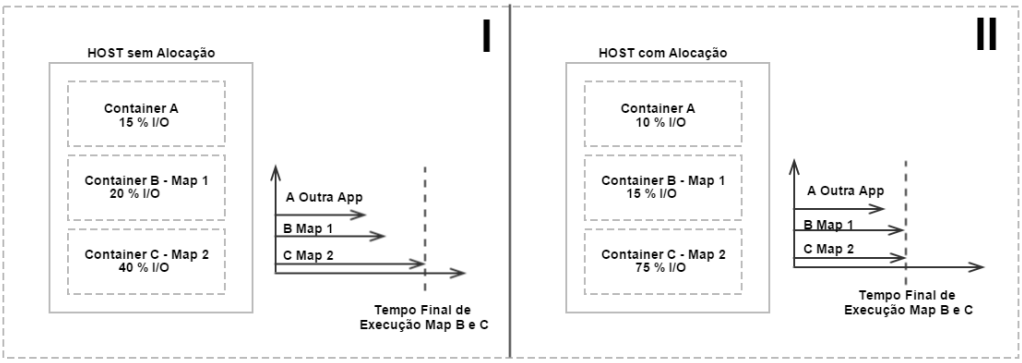


**Figura 1. Tempo de Execução Pré e Pós Alocação Preditiva de Recursos de I/O**

A Figura 1 ilustra o cenário de um cluster MR contendo 3 *containers*. Inicialmente, como pode ser visto em I, o *container* A possui 15% dos recursos do *cluster* e roda uma aplicação qualquer, enquanto os *containers* B e C rodam tarefas *map* e possuem 20 % e 40 % dos recursos do *cluster* MR. A linha tracejada indica o momento em que a fase de *map* de longa duração conclui a consolidação de dados em disco, nesse momento a fase de *reduce* é iniciada. Já com os ajustes efetuados utilizando a função preditiva, como pode ser visto em II, acredita-se ser possível restringir e realocar os recursos de I/O entre os *containers* de forma que fases *map* de longa duração recebam a maior fatia dos recursos durante a consolidação de seus dados e, que o tempo total de execução diminua de forma significativa devido a melhor alocação e gestão dos recursos.

## Referências

Apache (2014). Add support for disk io isolation/scheduling for containers.

Neves, M. V., De Rose, C. A., Katrinis, K., and Franke, H. (2014). Pythia: Faster big data in motion through predictive software-defined network optimization at runtime. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 82–90. IEEE.

Ryza, S. (2014). Managing multiple resources in hadoop 2 with yarn.

[1]Obs. Função preditiva já implementada e testada em ambientes reais

[2]Neves propõe uma abordagem preditiva para ambientes de rede