Uma Abordagem Baseada em Feedback Control Loops para Otimização de Desempenho no Hadoop

Anderson J. C. Pereira e Sandro S. Andrade

Grupo de Pesquisa em Sistemas Distribuídos, Otimização, Redes e Tempo-Real (GSORT)
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBa)
Av. Araújo Pinho, nº 39 - Canela - Salvador - BA - CEP: 40.110-150

{andersoncesar, sandroandrade}@ifba.edu.br

Resumo. *O modelo para computação distribuída MapReduce tornou-se popular no processamento distribuído de grandes conjuntos de dados. Apesar do Hadoop constituir uma plataforma completa para execução de jobs MapReduce, uma configuração mais elaborada dos seus diversos parâmetros pode trazer melhorias de desempenho. Este artigo apresenta uma extensão dos serviços básicos do Hadoop com o objetivo de suportar a execução facilitada de estratégias para feedback control. O objetivo é adicionar recursos de auto-gerenciamento para continuamente otimizar os parâmetros que impactam o desempenho. O artigo apresenta a arquitetura da solução proposta, um exemplo de uso de controladores PID e os resultados de experimentos preliminares.*

1. Introdução

A realização de operações de análise em grandes bases de dados geradas por sistemas atuais é de fundamental importância. O Hadoop [White 2012] é um projeto *open source* que disponibiliza duas tecnologias para este fim: um sistema de arquivos distribuído (HDFS - *Hadoop Distributed File System*) e uma plataforma para computação distribuída (YARN - *Yet Another Resource Negotiator*).

O Hadoop disponibiliza um amplo conjunto de parâmetros de configuração e boa parte deles pode potencialmente impactar o desempenho de execução de *jobs*. Embora o Hadoop defina valores *default* para estes parâmetros – viabilizando uma fácil implantação e execução de *jobs* – uma configuração mais criteriosa pode trazer melhorias de até 50% no desempenho. Pesquisas com sistemas *self-adaptive* [Salehie and Tahvildari 2009] têm obtido bons resultados em situações tais como o controle da quantidade de máquinas necessárias para suportar o *workload* atual e auto-ajuste de *clusters* para análise de *big data* [Lin et al. 2013, Herodotou et al. 2011].

Este artigo apresenta uma extensão aos serviços básicos do Hadoop de modo a suportar, de forma flexível e desacoplada, a utilização de *feedback control loops* [Hellerstein et al. 2004] para regulação automática de parâmetros de *tuning*. O serviço de adaptação proposto permite a utilização de diferentes leis de controle e a definição de diferentes arranjos entre múltiplos *loops* de adaptação. A arquitetura da solução proposta é apresentada e discutida, em conjunto com resultados de experimentos preliminares de utilização do serviço de adaptação.

2. O Hadoop e Feedback Control Loops

O HDFS é formado por dois serviços básicos: o `NameNode` e o `DataNode`. O `NameNode` é responsável pelo gerenciamento global dos blocos e réplicas que repre-

sentam arquivos armazenados no sistema de arquivos distribuído. É o serviço que é consultado quando deseja-se recuperar um arquivo do sistema, através da coleta e integração dos diversos blocos armazenados de forma distribuída no *cluster*. O `DataNode` é um serviço que executa em cada nó de armazenamento do *cluster* e tem como objetivo o gerenciamento dos dados que residem naquela máquina. O `DataNode` envia periodicamente ao `NameNode` informações sobre o *status* dos dados armazenados. Um `NameNode` secundário é utilizado para suportar falhas no `NameNode` primário, evitando a inviabilização de todo o *cluster*. Recursos para federação de `NameNodes`, com o objetivo de melhorar a escalabilidade do serviço, estão também disponíveis no Hadoop2.

O YARN é formado por dois serviços principais: o `ResourceManager` e o `NodeManager`. O `ResourceManager`, por sua vez, possui duas atribuições primárias: coordenar a execução de *jobs* no *cluster* e alocar recursos (memória, CPU e facilidades de comunicação) a *jobs* em execução. Tais atividades são executadas pelos componentes `ApplicationManager` e `Scheduler`, respectivamente. O `NodeManager` é um cliente do `ResourceManager`, presente em cada nó do *cluster* que está habilitado a executar tarefas de *map* ou *reduce*.

Ao receber uma solicitação de execução de *job*, o YARN seleciona uma máquina do *cluster* para hospedar o `ApplicationMaster` - serviço de coordenação daquele *job* em particular. O `ApplicationMaster` é responsável pela solicitação, ao `ResourceManager`, dos recursos necessários à execução do *job*. O `Scheduler` solicita a alocação de *containers* de execução de tarefas de *map*, preferencialmente naquelas máquinas que contêm blocos do arquivo de entrada a ser processado (minimizando o tráfego de dados na rede). Dependendo do número de tarefas de *reduce* configurado para o *job*, uma ou mais máquinas do *cluster* são selecionadas para execução destas tarefas.

Um sistema de controle [Ogata2009] tem como objetivo fazer com que um determinado parâmetro de desempenho de um sistema (*measured output*) seja mantido em um valor de referência pré-definido (*reference input*), através da manipulação de um valor de entrada (*control input*) que interfere no desempenho em questão. Dentre as principais leis de controle utilizadas em sistemas com única entrada e única saída (SISO - *Single-Input Single-Output*), o PID (*Proportional-Integral-Derivative*) e suas variações têm sido amplamente utilizados na indústria. O PID decide a atuação a ser realizada no sistema a partir de três ações distintas: a primeira é proporcional ao erro atual (*reference input* - *measured output*), a segunda é proporcional à integral do erro e a terceira, proporcional à taxa de mudança (derivada) do erro. Um controlador PID é descrito,no domínio do tempo, por:

$$u[k] = K_P \cdot e[k] + K_I \cdot \sum_{i=1}^{k} e[i] + K_D \cdot (e[k] - e[k-1]) \quad (1)$$

Diferentes valores dos parâmetros K_P, K_I e K_D produzem controladores com diferentes características em relação ao tempo de estabilização e *overshoot* (ultrapassagem demasiada do *reference input*) apresentados na resposta de controle.

3. A Abordagem Proposta

O serviço para *feedback control loops* apresentado neste artigo realiza extensões no `NodeManager` e `ResourceManager` de modo a suportar a configuração facili-

tada de estratégias específicas de controle. O serviço realiza a execução de dois *loops* integrados de controle, implementados por dois novos serviços aqui propostos: o `NodeControlManager` e o `ApplicationControlManager`. O *loop* executado pelo `NodeControlManager` tem como objetivo a realização de adaptações locais nos `NodeManagers`, controlando o desempenho de uma tarefa de *map* em particular.

O serviço `ApplicationControlManager` é responsável pela execução de adaptações *cluster-wide*, tais como a adição/remoção de novos nós de processamento. Ambos os serviços são configurados a partir de um novo arquivo XML definido neste trabalho – `feedback-site.xml`. Neste arquivo, são informadas as estratégias de controle a serem utilizadas em cada um dos *loops*, bem como os valores de parâmetros requeridos pelo controlador em uso. O *loop* executado pelo serviço `ApplicationControlManager` deve operar a uma frequência menor que aquele executado pelo `NodeControlManager`, pois realiza adaptações mais drásticas, necessárias quanto atuações nos `DataNodes` não conseguem entregar o serviço desejado.

O variável a ser controlada deve ser um dos parâmetros de configuração do Hadoop e é informada através das propriedades `nodectrl.controlvar.file` e `nodectrl.controlvar.name` – para o controlador do `Nodemanager` – e `appctrl.controlvar.file` e `appctrl.controlvar.name` – para o controlador do `ApplicationManager`. Atualmente, apenas o controlador PID e derivados estão disponíveis para uso e os serviços realizam toda a configuração dos *loops* a partir dos parâmetros descritos no arquivo `feedback-site.xml`. Um outro parâmetro, `appctrl.handler.count`, define o número de *threads* a serem criadas no `ApplicationControlManager`, com o objetivo de efetivar as comunicações com os `NodeControlManagers` do *cluster*. Sensores básicos para monitoração de utilização de CPU (no sistema operacional Linux) e tempo médio de resposta de *jobs MapReduce* são também disponibilizados na solução proposta.

4. Avaliação

Uma implementação preliminar da proposta apresentada neste trabalho foi avaliada através da utilização do serviço aqui proposto no controle de execuções do *job* `TeraSort`, uma das demonstrações mais conhecidas do Hadoop. O objetivo foi controlar o tempo médio de resposta dos *jobs*, manipulando o número máximo de tarefas de *map* executadas concorrentemente em cada nó do *cluster* (parâmetro `mapreduce.tasktracker.map.tasks.maximum` do arquivo de configuração `mapred-site.xml`). O Hadoop apresenta um valor *default* de 2 para este parâmetro. Em *clusters* formado por máquinas mais potentes, entretanto, um valor maior para este parâmetro potencializaria o desempenho, principalmente de *jobs data-intensive*.

Para isso, o Hadoop 2.3.0 e as extensões propostas neste trabalho foram instalados em um *cluster* formado por 10 *datanodes* e mais uma máquina executando o `NameNode` e o `ResourceManager`. As máquinas utilizadas foram computadores QuadCore com 4Gb de memória RAM e 500Gb de espaço em disco, todas executando o sistema operacional ArchLinux (kernel 3.13.7). Todos os parâmetros do Hadoop foram mantidos em seus valores *default* e o número de tarefas concorrentes de *map* foi controlado à medida em que o número de clientes solicitando execução de *jobs* aumentava. O objetivo foi garantir o tempo de resposta especificado, mesmo na presença de variações abruptas nas

demandas por serviço.

A modelagem do sistema foi realizada via *bump test*, uma técnica utilizada para obtenção de parâmetros que descrevem a dinâmica da aplicação a ser controlada. Estes parâmetros foram utilizados em um modelo FOPDT (*First-Order Plus Dead Time*) e empregou-se então o método empírico de *Ziegler-Nichols* para a obtenção dos parâmetros K_P, K_I e K_D. Análises de estabilidade foram realizadas para verificar se o sistema continua estável após a aplicação do controlador.

Resultados preliminares indicam que o serviço é capaz de realizar o controle do tempo de resposta para um número de clientes numa faixa entre 1 e 15. Para valores acima desta faixa, o controlador deixa de apresentar desempenho satisfatório, provavelmente como consequência de não-linearidades características do sistema sendo controlado. Técnicas para controle adaptativo, tais como escalonamento de ganho ou *Model-Identification Adaptive Control*, podem ser utilizadas para contornar esta situação. Observou-se ainda que o *overhead* de execução do controlador PID não é significativo, bem como o *overhead* de uso da rede para envio de informações dos `NodeControlManager`s para o `ApplicationControlManager`. O uso do *loop* mais externo, para adição e remoção de nós, não foi ainda considerado neste experimento.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou o projeto e implementação de um serviço para utilização de *feedback control loops* em arquiteturas *MapReduce* executadas sobre o Hadoop. Foram apresentados os novos serviços que viabilizam a infraestrutura de controle, como eles interagem com os componentes originais do Hadoop, bem como aspectos de configuração que seguem as estratégias já utilizadas no Hadoop. Experimentos preliminares com controladores PID demonstram que um grau de satisfatório de controle pode ser obtido quando valores adequados para os parâmetros do controlador são utilizados.

Como trabalhos futuros, destacam-se: a implementação de novas estratégias de controle (ex: *state-space control*, escalonamento de ganho), disponibilização prévia de implementações de filtros, experimentos com *loops* em cascata e investigação de mecanismos MIMO (*Multiple-Inputs Multiple-Outputs*) de controle.

Referências

[Hellerstein et al. 2004] Hellerstein, J. L., Diao, Y., Parekh, S., and Tilbury, D. M. (2004). *Feedback Control of Computing Systems*. John Wiley & Sons.

[Herodotou et al. 2011] Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., and Babu, S. (2011). Starfish: A self-tuning system for big data analytics. In *CIDR*, pages 261–272.

[Lin et al. 2013] Lin, M., Wierman, A., Andrew, L. L., and Thereska, E. (2013). Dynamic right-sizing for power-proportional data centers. *Networking, IEEE/ACM Transactions on*, 21(5):1378–1391.

[Salehie and Tahvildari 2009] Salehie, M. and Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):14:1–14:42.

[White 2012] White, T. (2012). *Hadoop: The Definitive Guide*. O'Reilly Media, 3rd edition.