Um Serviço para Feedback Control Loops em Arquiteturas MapReduce

Anderson J. C. Pereira e Sandro S. Andrade

Grupo de Pesquisa em Sistemas Distribuidos, Otimização, Redes e Tempo-Real (GSORT)
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBa)
Av. Araújo Pinho, nª 39 - Canela - Salvador - BA - CEP: 40.110-150

{andersoncesar, sandroandrade}@ifba.edu.br

Abstract. *The analysis of huge datasets generated by current software systems and the execution of jobs with demands for high performance have been the main factors motivating the adoption of cloud computing technologies. MapReduce is an architectural style and a distributed computation model widely used in industry, which provides inherent support for scalability and fault tolerance. Although Hadoop provides an easy-to-use platform for running MapReduce jobs, substantial performance improvements may be achieved with the use of carefully tailored values for the many parameters comprising the solution. This paper extends the basic Hadoop services, aiming at leveraging the support of feedback control strategies. The goal is to endow Hadoop with self-management capabilities which continuously optimize those parameters that directly impact job's performance. We present the architecture of the proposed mechanism, as well as examples of use of PID controllers and some preliminary empirical results.*

Resumo. *A análise de grandes bases de dados geradas por sistemas atuais e a execução de jobs com demandas por alto desempenho têm sido os principais fatores motivadores do uso de tecnologias para cloud computing. O MapReduce é um estilo arquitetural e um modelo para computação distribuída amplamente utilizado atualmente e com excelente suporte à escalabilidade e tolerância a falhas de nós do cluster. Embora o Hadoop disponibilize uma plataforma de fácil uso para execução de jobs MapReduce, melhorias substanciais de desempenho são observadas ao realizar uma configuração mais especializada dos diversos parâmetros que compõem a solução. 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 a otimização contínua dos 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

Nos últimos anos, a realização de operações de análise em grandes bases de dados tem sido fator determinante para o sucesso de muitas empresas. Sistemas corporativos, serviços *web* e redes sociais produzem juntos um volume impressionante de dados, alcançando a dimensão de *petabytes* diários [Russom et al. 2011, Zikopoulos et al. 2011]. A realização de análises sofisticadas em bases de dados contendo informações geradas por usuários

permite a investigação de tendências, derivação de perfis de uso de serviços e direcionamento de campanhas comerciais mais efetivas [Zhang et al. 2010].

O *MapReduce* [Dean and Ghemawat 2008] é um estilo arquitetural e um modelo computacional para processamento distribuído, amplamente utilizado na análise de *big data*. O *MapReduce* assume que os dados a serem processados estão armazenados em um sistema de arquivos distribuído e disponibiliza um *framework* que facilita o processamento paralelo e a posterior consolidação (redução) dos dados de saída gerados. A arquitetura é inerentemente escalável e com bom suporte a falhas em nós do *cluster*.

O Hadoop [Apache Foundation 2014, White 2012] é um projeto *open source* que disponibiliza duas tecnologias principais: um sistema de arquivos distribuído (HDFS - *Hadoop Distributed File System*) e uma plataforma para computação distribuída (YARN - *Yet Another Resource Negotiator*). O HDFS provê suporte escalável e tolerante a falhas para armazenamento de grandes bases de dados. O YARN suporta uma diversidade de modelos para computação paralela, dentre eles o *MapReduce*.

O desempenho de *jobs* paralelos executados sobre o Hadoop é influenciado pela natureza e organização dos dados de entrada e também por valores atribuídos a mais de 190 parâmetros, referentes ao HDFS, YARN e *MapReduce*. Embora o Hadoop defina valores *default* para estes parâmetros de modo a viabilizar uma fácil implantação e execução destes *jobs*, estudos [Jiang et al. 2010, Babu 2010, Kambatla et al. 2009] demonstram que ganhos de desempenho de até 50% podem ser obtidos com valores mais especializados para as características do *cluster* e do *job* em questão. Entretanto, conhecer os pontos de configuração e os valores ótimos para um determinado *job* requer amplo conhecimento da infrastrutura e modo de operação do HDFS, do YARN e do *MapReduce*.

Tal cenário – caracterizado pela complexidade das operações de *tuning* e inexistência de uma configuração ótima para todas as situações – tem sido o foco de pesquisas relacionadas a sistemas *self-adaptive* [Salehie and Tahvildari 2009]. O objetivo é dotar aplicações com a capacidade de continuamente monitorar o seu próprio funcionamento e do ambiente de execução, realizando adaptações sempre que o nível esperado de qualidade de serviço não for atendido. Como exemplos, pode-se citar os mecanismos para otimização de consumo de energia em *data centers* e os sistema de auto-ajuste de *clusters* para análise de *big data* [Heller et al. 2010, Herodotou et al. 2011].

Este artigo apresenta como os serviços básicos do Hadoop podem ser estendidos 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.

O restante deste artigo está organizado como segue. A seção 2 apresenta a arquitetura básica do HDFS e do YARN. A seção 3 discute os fundamentos sobre *feedback control*, enquanto a seção 4 apresenta o projeto do serviço de adaptação proposto. A seção 5 apresenta o cenário de avaliação utilizado e a discussão dos resultados. Por fim, a seção 6 apresenta os trabalhos correlatos e a seção 7 conclui o artigo, apresentando possibilidades de trabalhos futuros.

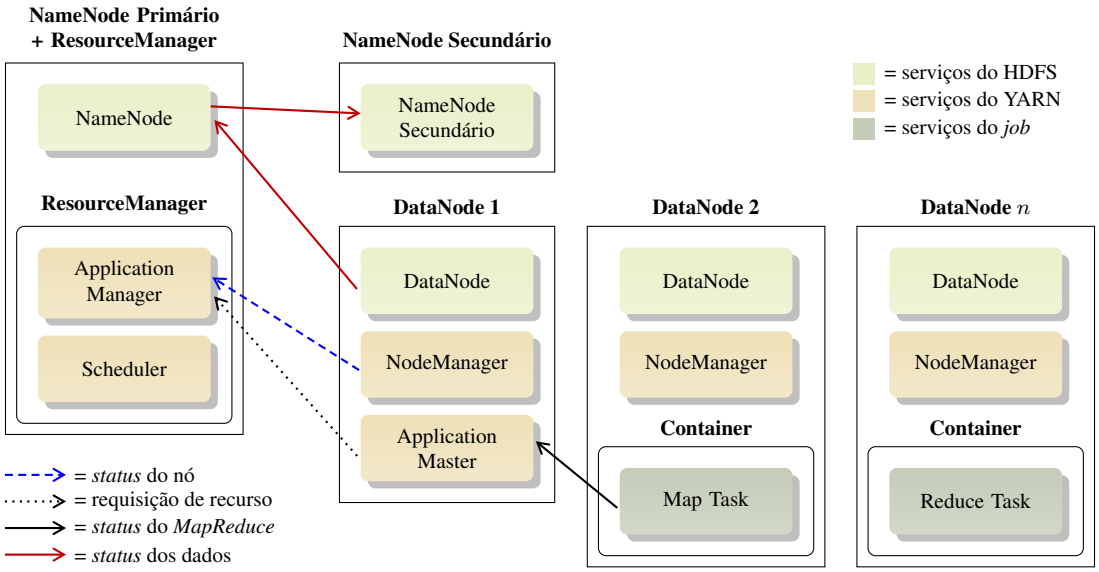

Figura 1. Componentes fundamentais do HDFS e do YARN (Hadoop 2.3.0).

2. Hadoop e Aplicações MapReduce

O Hadoop2 traz uma série de decisões arquiteturais que resolvem parte dos problemas de escalabilidade identificados na sua versão anterior. A Figura 1 apresenta os principais serviços disponibilizados pelo HDFS e YARN, bem como aqueles presentes durante a execução de um *job MapReduce*.

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 representam 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 a execução do `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`

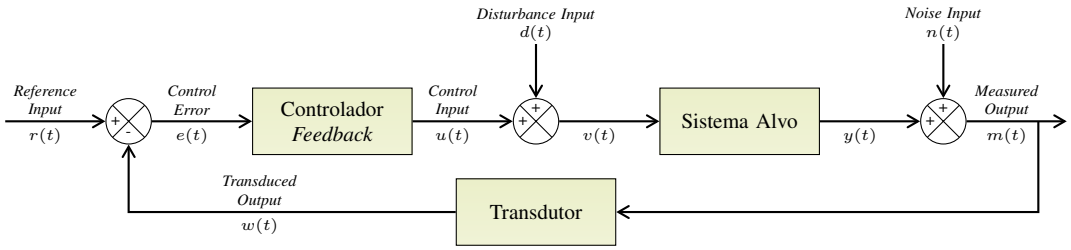

Figura 2. Componentes de uma arquitetura genérica para controle em malha fechada.

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.

O Hadoop2 disponibiliza uma arquitetura flexível, onde novos escalonadores ou novos modelos de computação distribuída podem ser definidos e integrados ao *framework*. Este trabalho adiciona um novo serviço para configuração dinâmica de parâmetros do *cluster* e novos arquivos de configuração, que indicam as estratégias e parâmetros de controle a serem utilizados.

3. Feedback Control em Sistemas Computacionais

Um sistema de controle [Ogata 2009] tem como objetivo fazer com que um determinado parâmetro de desempenho de um sistema (*measured output*) seja mantido em um valor de referência predefinido (*reference input*), através da manipulação de um valor de entrada (*control input*) que interfere no desempenho em questão. Após o desenvolvimento dos fundamentos matemáticos para estabilidade de sistemas dinâmicos no final do século XIX e das técnicas de análise e projeto de controladores no século XX, os sistemas de controle passaram a estar presentes em aplicações de diversas áreas. Atualmente, são utilizados para navegação de mísseis, aeronaves e navios; controle de níveis, temperaturas e concentrações em processos químicos; automação de plantas industriais e robôs; posicionamento de antenas; leitura de CDs; dentre inúmeras outras aplicações.

A Teoria de Controle disponibiliza todo o fundamento matemático e sistematização dos processos de projeto e análise de sistemas de controle. O objetivo é garantir que certas propriedades sejam mantidas no sistema controlado. Tais propriedades implicam, por exemplo, no conforto e segurança dos passageiros de uma aeronave, na qualidade apresentada pelo produto final de um processo químico, ou na garantia da qualidade de serviço em um sistema computacional.

A Figura 2 apresenta os principais componentes de um sistema de *feedback control*. O objetivo é monitorar uma variável de interesse (*measured output*) – como por exemplo tempo médio de resposta ou uso de CPU – e verificar quão distante esta variável está do valor para ela desejado (*reference input*). Esta diferença (*control error*) é utilizada como parâmetro de entrada para o controlador, que decide como atuar no sistema-alvo (via *Control Input*) de modo a fazer com que o *measured output* se iguale ao *reference input*.

Dentre as principais leis de controle utilizadas em sistemas com única en-

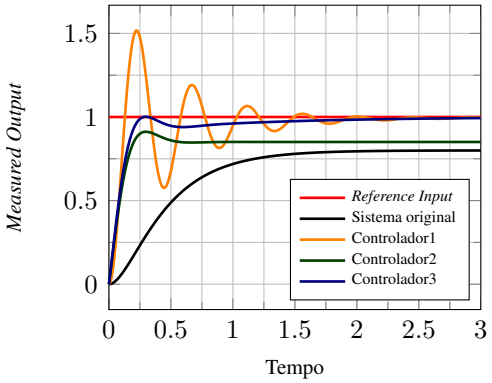

Figura 3. Ações de controladores com diferentes valores para os parâmetros K_P, K_I e K_D.

trada 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 (*control error = 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:

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

Os componentes proporcional, integral ou derivativo podem ser anulados ajustando os parâmetros K_P, K_I ou K_D para zero, respectivamente, o que viabiliza a utilização de variações tais como controle P, I, PI e PD. Controladores P representam uma forma simples de compensação de perturbações, mas são conhecidos pela sua impossibilidade de redução do erro em regime estacionário a zero (baixa precisão). Se o erro $e[k]$ for zero a atuação $u[k]$ também será zero, levando à existência de erro em regime estacionário sempre que o *reference input* ou perturbação forem não-nulos.

Controladores I, por sua vez, atuam de forma proporcional à integral (histórico acumulado) do erro, ou seja, continuam ampliando a atuação mesmo quando o erro se mantém relativamente constante. Em função deste comportamento, são capazes de reduzir o erro em regime estacionário a zero. Por outro lado, este benefício é geralmente acompanhado de tempos de estabilização mais longos. Controladores PI buscam combinar a precisão dos controladores I com os tempos de estabilização mais baixos dos controladores P. Esta combinação é a mais prevalente na indústria atualmente.

O componente derivativo indica que a atuação deve ser mais forte quanto mais drástica for a mudança do erro. Embora isso ajude a projetar controladores com reação mais rápida, este comportamento mais "agressivo e precipitado" pode aumentar o *overshoot* (ultrapassagem demasiada do *reference input*) se mal utilizado ou quando o *measured output* possui fator estocástico forte. O componente derivativo nunca é utilizado sozinho, pois um erro constante implica em derivada nula e, consequentemente, atuação nula. Controladores PD podem ser utilizados para reduzir o *overshoot* em sistemas com oscilações quando controlados via controlador P. Finalmente, controladores PID podem ser utilizados quando deseja-se combinar as influências dos três fatores.

Diferentes valores dos parâmetros K_P, K_I e K_D produzem controladores com diferentes características em relação ao tempo de estabilização e *overshoot* apresentados na resposta de controle. Conforme ilustrado na Figura 3, um sistema original – apresentando tempo de estabilização e erro de convergência altos – pode ter tais características melhoradas através do uso de um controlador. Embora o `controlador1` apresente erro de convergência nulo (o *measured output* passa a ser exatamente igual ao *reference input*), um alto *overshoot* é verificado. Tal situação pode implicar no uso demasiado de recursos computacionais e de rede. Já o `controlador2` apresenta *overshoot* nulo e tempo de estabilização baixo, porém a convergência é ruim (*measured output* converge para um valor menor que o *reference input*). Note como o `controlador3` apresenta um comportamento caracterizado por *overshoot* e erro de convergência nulos, associados a um baixo tempo de estabilização.

4. O Serviço Proposto

O serviço para *feedback control loops* proposto neste artigo realiza extensões no `NodeManager` e `ResourceManager` do Hadoop de modo a suportar a configuração facilitada de estratégias específicas de controle. Conforme apresentado na Figura 4, o serviço proposto prevê a execução de dois *loops* integrados de controle. O *loop* executado pelo serviço `NodeControlManager` tem como objetivo a realização de adaptações locais nos `NodeManagers`, controlando o desempenho de uma tarefa de *map* em particular.

O serviço `ApplicationControlManager`, por sua vez, é responsável pela execução de adaptações *cluster-wide*, tais como a adição ou remoção de novos nós de processamento. Ambos os serviços são configurados a partir de um novo arquivo XML definido no Hadoop – `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 sendo utilizado. O *loop* executado pelo serviço `ApplicationControlManager` deve operar a uma frequência menor que a adotada no `NodeControlManager`, pois realiza adaptações mais drásticas, necessárias quanto atuações locais nos `DataNodes` não conseguem entregar o nível de 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 de tempo médio de resposta de *jobs MapReduce* são também disponibilizados na solução proposta.

5. Experimentos de Avaliação

Uma implementação preliminar da proposta apresentada neste trabalho foi avaliada através da utilização do serviço aqui proposto no controle de execuções do

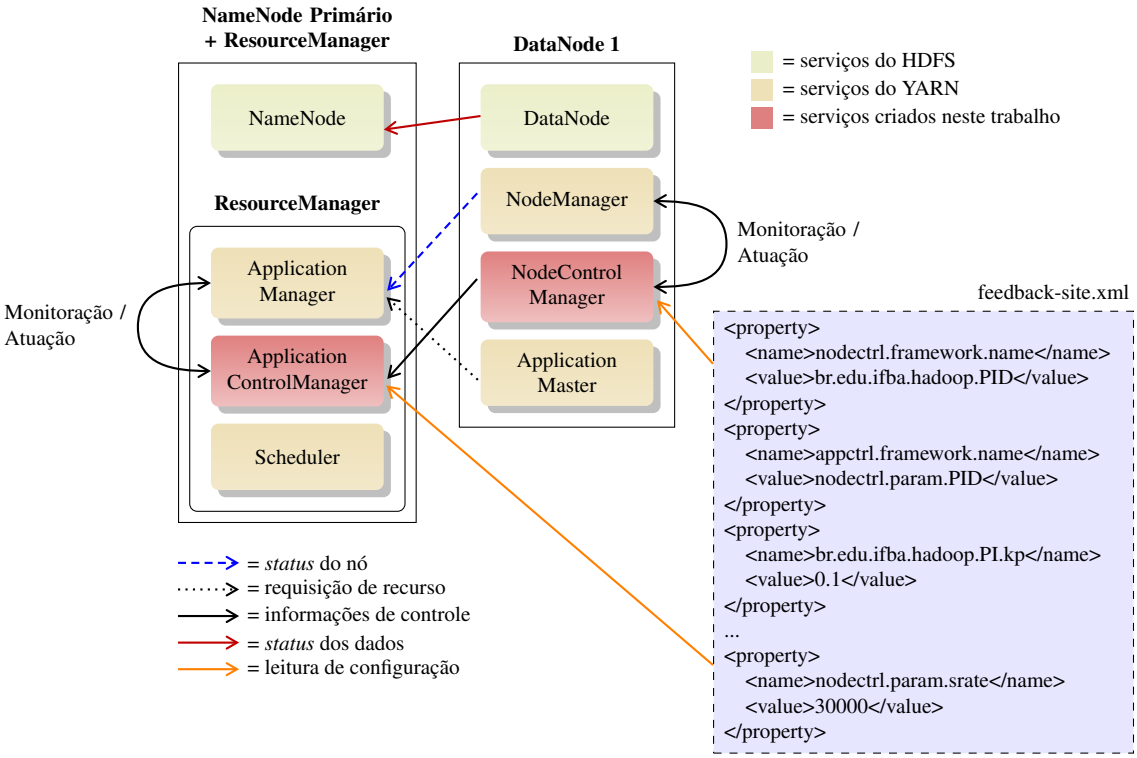

Figura 4. Extensões realizadas neste trabalho para o suporte a *feedback control loops* no Hadoop.

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 da 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* (MIAC) [Hellerstein et al. 2004], 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 `NodeControlManagers` para o `ApplicationControlManager`. O uso do *loop* mais externo, para adição e remoção de nós, não foi ainda considerado neste experimento.

6. Trabalhos Correlatos

Uma série de esforços para implementação de comportamento *self-adaptive* em *clusters* através do uso de *feedback control* podem ser encontrados na literatura. [Herodotou et al. 2011] apresentam o Starfish, um serviço de auto-configuração do Hadoop em três níveis: configuração de *job*, configuração de *workflow* e configuração de *workload*. A abordagem utiliza técnicas de instrumentação dinâmica e coleta de estatísticas para guiar o processo de *tuning* automático. Um novo *workflow-aware scheduler* e uma *what-if engine* são introduzidas para viabilizar a configuração de *workflows*.

[Li et al. 2009] propõem uma abordagem que utiliza *feedback control* no gerenciamento adaptativo de máquinas virtuais. O objetivo é ajustar a utilização de recursos virtualizados de modo a alcançar os SLAs (*Service-Level Agreements*) especificados. Os autores utilizam *dynamic state space feedback control*, uma técnica para implementação de controladores MIMO (*Multiple-Inputs Multiple-Outputs*). [Wang and Chen 2008] apresentam uma técnica para controle de energia em *clusters* através do uso de *model-predictive controllers*. O artigo apresenta a análise de estabilidade do controlador proposto e comparações empíricas em relação a dois outros controladores.

[Lin et al. 2013] apresentam uma abordagem para controlar o número de máquinas que devem permanecer ligadas em um *data center*, de modo a suportar o *workload* atualmente experimentado. Os autores apresentam um algoritmo chamado *Lazy Capacity Provisioning*, que utiliza uma janela de predição e um modelo ótimo de referência para decidir o número de máquinas necessárias. [Berekmeri et al. 2014] apresentam uma solução híbrida que utiliza controladores PI e controladores *feedforward* para controlar o tempo médio de resposta de *jobs* na presença de perturbações, através da manipulação do número de nós presentes no *cluster*.

[Krebs and Mehta 2013] apresentam uma abordagem que combina escalonadores baseados em prioridades com técnicas de *feedback control* para realizar isolamento de desempenho em aplicações *multi-tenant* em *clouds*. A abordagem baseia-se no escalonamento *Weighted Round Robin* e no uso de controladores PI. A avaliação foi realizada utilizando a plataforma *SAP Hana Cloud* em conjunto com o *benchmark* MT TPC-W. [Wu et al. 2014] apresentam uma abordagem para redução de duplicação de dados em sistemas de arquivos distribuídos através do uso de controladores. Um *Index Name Server* atua como controlador, gerenciando e otimizando os nós de armazenamento de dados de acordo com condições de transmissão dos clientes. Experimentos indicam que a abordagem diminui de 20%-50% o *overhead* de transmissão de dados devido à replicação.

A abordagem apresentada neste trabalho diferente dos estudos acima em uma série de aspectos. Primeiro, dotar o Hadoop com serviços genéricos e flexíveis para implementação de *feedback control loops* abre caminho para a avaliação facilitada de abordagens alternativas e diminui a complexidade de implementação de tais mecanismos. Segundo, o esquema de múltiplos *loops* definido neste trabalho vem sendo amplamente utilizado para controle multi-escala (em diferentes granularidades de adaptação) de serviços de *cloud computing*. Por fim, a flexibilidade de configuração das variáveis controladas e dos *inputs* de controle permite a investigação rápida dos parâmetros mais influentes e uma especificação mais produtiva dos *loops* de controle envolvidos.

7. Conclusões e Trabalhos Futuros

O uso de técnicas para manutenção de qualidade de serviço em plataformas de *cloud computing* e outras infraestruturas *multi-tenant* é de fundamental importância para a diminuição de custos e boa utilização dos recursos computacionais disponíveis. O Hadoop – embora amplamente utilizado em projetos reais na indústria – carece ainda de mecanismos flexíveis e expressivos para *auto-tuning* e auto-gerenciamento em geral.

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

Referências

[Apache Foundation 2014] Apache Foundation (2014). Hadoop. `http://hadoop.apache.org`. Acesso: 29/04/2014.

[Babu 2010] Babu, S. (2010). Towards automatic optimization of mapreduce programs. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 137–142, New York, NY, USA. ACM.

[Berekmeri et al. 2014] Berekmeri, M., Serrano, D., Bouchenak, S., Marchand, N., and Robu, B. (2014). A control approach for performance of big data systems. In *IFAC World Congress*. hal-00940282.

[Dean and Ghemawat 2008] Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

[Heller et al. 2010] Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., and McKeown, N. (2010). ElasticTree: Saving energy in data center networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 17–17, Berkeley, CA, USA. USENIX Association.

[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 2011, Fifth Biennial Conference on Innovative Data Systems Research*, pages 261–272.

[Jiang et al. 2010] Jiang, D., Ooi, B. C., Shi, L., and Wu, S. (2010). The performance of MapReduce: An in-depth study. *Proceedings of the VLDB Endowment*, 3(1-2):472–483.

[Kambatla et al. 2009] Kambatla, K., Pathak, A., and Pucha, H. (2009). Towards optimizing Hadoop provisioning in the cloud. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, Berkeley, CA, USA. USENIX Association.

[Krebs and Mehta 2013] Krebs, R. and Mehta, A. (2013). A feedback controlled scheduler for performance isolation in multi-tenant applications. In *Cloud and Green Computing (CGC), 2013 Third International Conference on*, pages 195–196. IEEE.

[Li et al. 2009] Li, Q., Hao, Q., Xiao, L., and Li, Z. (2009). Adaptive management of virtualized resources in cloud computing using feedback control. In *Information Science and Engineering (ICISE), 2009 1st International Conference on*, pages 99–102. IEEE.

[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.

[Ogata 2009] Ogata, K. (2009). *Modern Control Engineering (5th Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Russom et al. 2011] Russom, P. et al. (2011). Big data analytics. *TDWI Best Practices Report, Fourth Quarter*.

[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.

[Wang and Chen 2008] Wang, X. and Chen, M. (2008). Cluster-level feedback power control for performance optimization. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 101–110. IEEE.

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

[Wu et al. 2014] Wu, T.-Y., Pan, J.-S., and Lin, C.-F. (2014). Improving accessing efficiency of cloud storage using de-duplication and feedback schemes. *IEEE Systems Journal*, 8(1):208–218.

[Zhang et al. 2010] Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.

[Zikopoulos et al. 2011] Zikopoulos, P., Eaton, C., et al. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media.