Uma Plataforma de Rede Definida por Software para Ambientes Convergentes de Computação Paralela

Alexandre T. Oliveira[1], **Alex B. Vieira**[1], **Antônio Tadeu A. Gomes**[2], **Artur Ziviani**[2]

[1] Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF)

[2]Laboratório Nacional de Computação Científica (LNCC)

alexandre.tavares@ice.ufjf.br,alex.borges@ufjf.edu.br

{atagomes,ziviani}@lncc.br

Abstract. *The data deluge revolutionizes business and science at the same time as it demands ever increasing computational resources. High-performance computing (HPC) platforms tailored to massively parallel numerical simulations offer computational capacity that can be leveraged by Big Data Analytics solutions. Nevertheless, the convergence of Big Data and HPC should be examined in several ways; in particular, the network infrastructure needs to fit rather different applications requirements. The software-defined network model (SDN) may favor this convergence, thanks to its global view of the network and its programmability. In this sense, we present an SDN platform capable of supplying, in a convergent way, Big Data and HPC applications requirements. The platform applies the most appropriate routing mechanisms to each traffic profile, thus allowing a reduction in the execution time of the applications. We demonstrate through simulations the feasibility of our approach, by reducing the execution time of MPI applications in specific scenarios by up to 11% and of Hadoop applications by up to 6%.*

Resumo. *O crescimento no volume dos dados tem revolucionado os negócios e a ciência ao mesmo tempo que demanda capacidade cada vez maior dos recursos computacionais. As plataformas de computação de alto desempenho (HPC), tradicionalmente empregadas em simulações numéricas massivamente paralelas, oferecem capacidade computacional que pode ser aproveitada na análise de Big Data. No entanto, a convergência de Big Data e HPC deve ser examinada sob vários aspectos; em particular, a infraestrutura de rede precisa ajustar-se a demandas de aplicações bem distintas. O modelo de rede definida por software (SDN) pode favorecer essa convergência, graças à sua visão global da rede e sua programabilidade. Nesse contexto, apresentamos uma plataforma SDN capaz de suprir, de forma convergente, os requisitos de aplicações Big Data e HPC. A plataforma aplica mecanismos de roteamento mais adequados a cada perfil de tráfego, permitindo assim a redução no tempo de execução de aplicações. Demonstramos por meio de simulações a viabilidade de nossa plataforma, ao reduzir o tempo de execução de aplicações reais MPI em cenários específicos em até 11% e Hadoop em até 6%.*

1. Introdução

O processamento de dados em larga escala impõe enormes desafios, tanto em termos de gerenciamento de memória quanto de processamento e acesso aos dados. Nesse sentido,

as plataformas de computação de alto desempenho (*High-Performance Computing* – HPC) —tradicionalmente empregadas em simulações numéricas massivamente paralelas, como previsão de tempo e clima, modelagem de reservatórios de hidrocarbonetos e simulação de atracamento molecular— oferecem capacidade de processamento computacional que pode ser aproveitada por uma vasta gama de soluções de computação intensiva de dados. Embora a utilização dessas plataformas de HPC por aplicações de *Big Data* pareça ser natural, o que se vê, de fato, é um descasamento entre essas arquiteturas, de modo que explorar recursos de HPC para atender soluções típicas de *Big Data* demanda o tratamento de diversos problemas [Oliveira et al. 2018].

A convergência entre *frameworks* de *Big Data* e plataformas de HPC precisa ser considerada sob diferentes aspectos, desde a tecnologia de *hardware*, *software* e aplicações/algoritmos até modelos de negócio e educação [Fox et al. 2015]. A rede de dados, em particular, precisa ser eficiente e flexível para se ajustar aos diferentes perfis de tráfego gerados, por exemplo, por aplicações paralelas que utilizam APIs tradicionais de ambientes de HPC, como *Message Passing Interface* (MPI), e por aplicações que empregam *frameworks* típicos de ambientes *Big Data*, como Hadoop e Spark. Note que as redes de dados tradicionais possuem controles complexos, cujo ajuste a requisitos específicos de aplicações demanda um alto custo administrativo, inviabilizando essa convergência na prática. Nesse aspecto, o paradigma de Rede Definida por Software (SDN) pode favorecer a integração dos ambientes de computação paralela. Ao desacoplar os planos de dados e de controle dos ativos de rede (comutadores e roteadores), o modelo SDN proporciona uma visão global e uma maior programabilidade da rede de dados, o que permite aumentar a flexibilidade e a eficiência dessas redes [Oliveira et al. 2018].

Neste artigo, apresentamos uma plataforma de comunicação de dados baseada em SDN capaz de suprir os requisitos de desempenho das aplicações que executam em ambientes convergentes de *Big Data* e HPC. A plataforma procura otimizar a comunicação dos dados em redes multicaminhos mediante o emprego, de forma dinâmica, dos algoritmos de roteamento mais adequados aos perfis de tráfego de cada uma dessas aplicações. Para tal, idealizamos uma API que permite às aplicações paralelas informarem ao controlador SDN, em tempo de execução, características gerais dos perfis de tráfego por elas gerados, tornando a rede ciente da aplicação. Da mesma forma, a API concebida possibilita que o administrador do ambiente configure os algoritmos de roteamento mais apropriados a cada um desses perfis de tráfego. Desse modo, o controlador torna-se apto a identificar o tráfego da aplicação através de configurações reativas e/ou proativas, e a aplicar a estratégia de roteamento previamente definida a esse tráfego.

Avaliamos a proposta por meio de simulações de um ambiente SDN Ethernet, com um cenário específico de topologia multicaminhos. Em uma primeira análise, utilizamos um emulador de aplicações Hadoop para identificar as configurações de rede mais adequadas às aplicações típicas de ambientes *Big Data*. Medimos o tempo de conclusão da emulação em função de quatro algoritmos simples de seleção de rotas. Os resultados dessa simulação mostram diferenças de até 22% nos tempos de execução, dependendo do algoritmo usado. Posteriormente, analisamos o desempenho de aplicações reais *Big Data* e HPC co-localizadas no mesmo ambiente, utilizando os mecanismos de identificação de tráfego baseados na API proposta neste trabalho. Nessa simulação, aplicamos as estratégias de roteamento mais adequadas a cada aplicação, considerando

as avaliações apresentadas neste trabalho para Hadoop e nossas análises anteriores apresentadas em [Oliveira et al. 2018] para aplicações MPI somente. Nessa circunstância, verificamos que a plataforma reduziu os tempos de execução em mais de 11% para as aplicações MPI e em torno de 6% para as aplicações Hadoop, em comparação com um cenário onde é aplicado um algoritmo de roteamento genérico para ambas as aplicações.

Este trabalho possui duas contribuições principais. Primeiro, a plataforma aqui apresentada destaca-se por abordar a otimização de aplicações *Big Data* e HPC no âmbito de um ambiente convergente, com enfoque na rede de comunicação. De fato, as propostas de soluções nessa área geralmente tratam essas classes de aplicações de forma isolada ou conscientes da rede [Takahashi et al. 2018, Bhatia et al. 2017, Alsmadi et al. 2016, Liang and Lau 2016, Narayan et al. 2012, Qin et al. 2015], em oposição à nossa proposta de rede convergente e ciente da aplicação, o que dá ao administrador maior controle para aumentar a eficiência geral da rede. Segundo, esta pesquisa contribui para que se abra uma nova perspectiva no estudo de soluções convergentes, no sentido de se ponderar sobre a implantação de infraestruturas SDN Ethernet, significativamente mais baratas, para atender soluções de HPC. Não obstante, infraestruturas de rede baseadas em tecnologias como InfiniBand ainda podem se beneficiar das capacidades do modelo SDN, tendo em vista que fabricantes de dispositivos de rede que implementam essa tecnologia começam a fornecer produtos com suporte a SDN [Mellanox 2015].

No restante deste artigo, a Seção 2 discute os trabalhos relacionados. Na Seção 3, mostramos alguns conceitos de computação avançada. A Seção 4 apresenta a plataforma de comunicação convergente baseada em SDN. A análise das aplicações Hadoop é apresentada na Seção 5. Por sua vez, os detalhes da avaliação com aplicações reais são descritos na Seção 6. A Seção 7 mostra as conclusões do artigo e os trabalhos futuros.

2. Trabalhos Relacionados

Na literatura, existem trabalhos que empregam SDN para aprimorar o processamento MPI. Por exemplo, [Takahashi et al. 2018] apresentam um mecanismo de coordenação de controle de rede e execução de aplicações definido por *software*. Esse mecanismo é baseado em um *framework* genérico MPI aprimorado por SDN e incorporado em uma biblioteca. No trabalho de [Bhatia et al. 2017], o controlador SDN utiliza estatísticas de fluxo para atualizar um grafo de topologia de rede usado por um algoritmo de roteamento adaptativo na seleção dinâmica dos caminhos. Por sua vez, [Alsmadi et al. 2016] usam informações de rede, obtidas pelo controlador, e empregam um orquestrador de recursos para selecionar os nós mais adequados ao processamento de cada tarefa.

Há ainda na literatura algumas pesquisas que utilizam SDN na melhoria do processamento de dados em larga escala do Hadoop. Por exemplo, os trabalhos de [Liang and Lau 2016] e [Narayan et al. 2012] exploram o uso de SDN para maximizar a utilização da largura de banda da rede durante a fase de "embaralhamento" (*shuffle*) dos *jobs* MapReduce. [Qin et al. 2015] também propõem um agendador de tarefas ciente da largura de banda utilizando o modelo SDN. Nessa abordagem, chamada BASS, o agendador obtém uma avaliação completa da largura de banda da rede através do controlador SDN e a considera como um parâmetro vital para o agendamento da tarefa.

A convergência entre HPC e *Big Data* é um tópico atual de pesquisa. Em parte, essa convergência é tratada por trabalhos como o de [Ponce et al. 2018], que

apresentam uma extensão do modelo de programação paralela e distribuída COMP Superscalar (COMPSs) para o processamento de dados massivos, onde o COMPSs é integrado ao HDFS. Há também trabalhos cujos princípios fornecem diversas percepções em torno dessa convergência em camadas mais baixas no contexto de redes, embora não abordem esse tema de forma explícita. Por exemplo, [Webb et al. 2011] formalizam a possibilidade de uso simultâneo de múltiplos mecanismos de roteamento em um *data center*, permitindo às aplicações defini-los e implantá-los conforme suas necessidades. Como outro exemplo, [Trois et al. 2017] apresentam NetSA, um *framework* que aplica SDN para explorar os padrões de comunicação de aplicações científicas, considerando restrições de latência e largura de banda no balanceamento do tráfego através da rede.

De forma geral, os trabalhos que exploram as capacidades do paradigma SDN nos ambientes HPC e *Big Data* o fazem de maneira isolada, como evidenciam os artigos de [Takahashi et al. 2018], [Bhatia et al. 2017] e [Alsmadi et al. 2016] (MPI), e [Liang and Lau 2016], [Narayan et al. 2012] e [Qin et al. 2015] (Hadoop). Nossa proposta, por sua vez, tira proveito da flexibilidade proporcionada por SDN para prover uma infraestrutura de comunicação verdadeiramente convergente. Nesse aspecto, embora [Ponce et al. 2018] dediquem-se ao tema da convergência, eles focam no nível da aplicação. Além disso, nossa pesquisa baseia-se em um modelo onde a rede é ciente da aplicação, com a rede ajustando-se aos requisitos das aplicações, em contraponto aos estudos de [Alsmadi et al. 2016], [Liang and Lau 2016] e [Qin et al. 2015], que propõem soluções sob um ponto de vista no qual a aplicação se torna consciente da rede, ou seja, adapta-se às condições da rede subjacente sem alterá-la. Por fim, no contexto dos métodos de otimização da comunicação, nossa proposta assemelha-se às de [Webb et al. 2011] e [Trois et al. 2017], ao fazer uso de mecanismos de roteamento distintos, enquanto que [Narayan et al. 2012] recorrem a conceitos de QoS para maximizar o tráfego.

Ainda, cabe destacar que no nosso trabalho anterior ([Oliveira et al. 2018]), no contexto da plataforma SDN, analisamos as melhores condições de rede para as aplicações MPI somente. Em contraste, no presente artigo, realizamos o mesmo estudo para as aplicações Hadoop, e utilizamos ambos os resultados para avaliar o desempenho dessas aplicações sobre um ambiente efetivamente convergente, com a plataforma garantindo a operação da rede de forma consciente e, acima de tudo, eficiente.

3. Ecossistemas de Computação Avançada

Nesta seção, apresentamos alguns conceitos por trás dos ecossistemas de computação avançada, incluindo arquiteturas paralelas, redes de comunicação de dados e modelos de programação paralela.

3.1. Arquiteturas Paralelas

Aplicações que demandam grande poder computacional utilizam sistemas de computação paralela para acelerarem sua execução. Sistemas paralelos são criados combinando múltiplos elementos de processamento em um único sistema maior. A vasta maioria dos sistemas paralelos modernos encaixam-se na arquitetura MIMD (*Multiple Instruction, Multiple Data*), tipicamente classificada de acordo com a organização de memória: compartilhada ou distribuída [Mattson et al. 2004].

Nosso interesse neste artigo é nos sistemas de memória distribuída, em que cada processo tem seu próprio espaço de endereçamento e a comunicação com os demais

processos ocorre mediante o envio e o recebimento de mensagens via uma rede de comunicação. Em ambientes de HPC, eles são tradicionalmente implementados como aglomerados (*clusters*) de computadores (nós de trabalho) interconectados por tecnologia específica de rede (vide Seção 3.2). Esse é o caso, por exemplo, do supercomputador brasileiro SDumont [LNCC 2018]. Nesse tipo de sistema, os nós de trabalho executam processos que usualmente compartilham um sistema de arquivos distribuídos como o Lustre[1], dotado de nós dedicados ao armazenamento e à gerência dos arquivos.

No ecossistema *Big Data*, os dados a serem processados, a princípio, não cabem na memória principal de um único computador. Assim, *clusters* também são vistos como a plataforma padrão para esses ambientes [Porto 2017]. Contudo, suas aplicações típicas executam com base em um modelo no qual, diferentemente dos ambientes de HPC, os dados são distribuídos pelos nós de trabalho e os processos são alocados a esses nós, buscando assim minimizar a transferência de dados em disco. Para tanto, esses ambientes empregam frequentemente um sistema de arquivos distribuídos como o HDFS (*Hadoop Distributed File System*)[2], cuja arquitetura, diferentemente daquela de sistemas como o Lustre, não considera o compartilhamento de arquivos entre nós de trabalho.

3.2. Redes de Comunicação

As redes de comunicação dos sistemas de *cluster* em ambientes de HPC conectam os nós de trabalho através de interconexões de alta vazão e baixa latência. Essas redes são implantadas com diversas tecnologias de comutação de pacotes e sobre topologias físicas multicaminhos como *fat-tree* e Torus, que oferecem ao mesmo tempo desempenho e redundância. InfiniBand é uma das principais tecnologias utilizadas nesses sistemas HPC, sendo usada em mais de 25% dos supercomputadores que integram atualmente a lista TOP500[3], incluindo o Summit [ORNL 2018], o maior supercomputador da atualidade.

As redes Ethernet são amplamente utilizadas nos ecossistemas de computação avançada. Graças ao seu custo reduzido e à disponibilidade de produtos, fornecedores e especialistas, Ethernet é a tecnologia de comutação de pacotes predominante nos sistemas de *cluster* em ambientes *Big Data*. Mesmo com suas limitações de capacidade efetiva e retardo, essa tecnologia também é usada em ambientes de HPC, estando presente em praticamente 50% das plataformas listadas na última edição da TOP500.

O paradigma SDN também é aplicável nos sistemas de computação paralela, como complemento aos modelos de rede tradicionalmente usados nesses ambientes. O termo SDN refere-se a uma arquitetura de rede fundamentada em quatro pilares: (i) desacoplamento dos planos de dados e de controle; (ii) decisões de encaminhamento baseadas em fluxos; (iii) transferência da lógica de controle para um elemento controlador externo logicamente centralizado; e (iv) programação da rede através de aplicações de *software* executando no topo do modelo [Kreutz et al. 2015]. Nessa arquitetura, a interface entre o controlador SDN e os dispositivos da infraestrutura de rede é fornecida por protocolos seguros como o OpenFlow [McKeown et al. 2008]. O OpenFlow é considerado um padrão *de facto*, utilizado pela maioria das plataformas atuais. Assim, em

[1] `http://lustre.org/`

[2] `https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html`

[3] A lista TOP500 é um *ranking* periódico dos 500 supercomputadores mais poderosos do mundo. Sua edição mais recente está disponível em `https://www.top500.org/lists/2018/11/`

um ambiente SDN típico, os elementos de encaminhamento de dados são habilitados com o OpenFlow, através dos quais podem ser gerenciados por um controlador compatível.

3.3. Programação Paralela

A classificação da arquitetura dos sistemas paralelos e as características das aplicações influenciam a escolha do modelo de programação usado para expressar a concorrência nas aplicações [Mattson et al. 2004]. O modelo de troca de mensagens entre processos é predominante em aplicações de simulação numérica executando sobre sistemas de *cluster* em ambientes de HPC, sendo MPI o padrão de implementação mais usado nessas aplicações. MPI define uma API para gerenciamento de processos e operações de comunicação ponto-a-ponto e coletiva. Atualmente, diversas bibliotecas implementam o padrão MPI, tanto *open source* (OpenMPI, MPICH) como proprietárias.

Nos ambientes *Big Data*, o modelo de programação funcional MapReduce é o mais utilizado. Nele, a computação é definida em termos de funções *map* e *reduce*. A função *map* processa cada item do conjunto de entrada, enquanto a função *reduce* processa um conjunto de elementos da entrada. Com base nesse modelo, a computação no *cluster* pode ser automaticamente paralelizada pelo *framework* de computação intensiva de dados subjacente. Diversos desses *frameworks* estão disponíveis atualmente, entre eles o Hadoop, o Spark e o Flink.

4. Plataforma de Comunicação Convergente Baseada em SDN

Nossa proposta de plataforma de comunicação convergente é idealizada sobre uma arquitetura baseada em SDN. Ela é capaz de suprir, de forma flexível e dinâmica, os requisitos de desempenho das aplicações típicas de *Big Data* e de HPC. Essas aplicações são executadas em paralelo pelos nós da arquitetura, os quais trocam dados e/ou mensagens através da infraestrutura de rede. A plataforma atende esses requisitos mediante a otimização da comunicação. Nesse processo, a vazão da rede é maximizada e o tempo de execução das aplicações tende a diminuir.

Considerando uma plataforma convergente sobre uma arquitetura de *cluster* com topologia multicaminhos qualquer, o paradigma SDN é usado no aprimoramento da comunicação dos dados, aproveitando os múltiplos caminhos disponíveis na infraestrutura de rede. Para tal, o controlador SDN identifica o tráfego das aplicações e aplica, de forma dinâmica, automatizada e eficiente, políticas de roteamento mais adequadas aos diferentes fluxos de pacotes de dados que trafegam sobre a rede. Os algoritmos de roteamento podem implementar diversas técnicas de engenharia de tráfego, encaminhando os fluxos por caminhos menos congestionados, por caminhos isolados ou simplesmente balanceando o tráfego. Na verdade, alternativas de roteamento são facilmente incorporadas à plataforma. Diferentes aplicações MapReduce e MPI ajustam-se melhor a diferentes estratégias de roteamento em diferentes cenários. Portanto, a definição dos métodos de seleção de rotas mais adequados a cada padrão de comunicação influencia diretamente no desempenho da plataforma. Assim, na nossa abordagem, é essencial que o administrador da rede estabeleça as melhores alternativas para cada caso, levando em conta todo o ambiente.

A identificação do tráfego pode ser reativa ou proativa. No modo reativo, as aplicações informam suas características ao controlador SDN em tempo de execução. Embora exija modificações nas aplicações, essa abordagem fornece grande flexibilidade

aos usuários do ambiente convergente. No modo proativo, o controlador SDN reconhece o tráfego de forma transparente, através de padrões comuns dessas aplicações, fornecidos pelo administrador da rede. Em ambos os modos, a rede se torna ciente da aplicação.

Seguindo esses princípios, idealizamos uma API básica. Ela é fundamentada na simples associação entre o tipo da aplicação (i.e., Hadoop, MPI) e os números de portas dos protocolos da camada de transporte usados por cada uma delas. A API também permite que o administrador do ambiente defina o mapeamento entre o tipo da aplicação e o algoritmo de roteamento correspondente. Eventuais sobreposições de regras podem ser tratadas por controles próprios. Para a associação entre os tipos de aplicações e suas portas, a API propõe ao menos três funções, que podem ser invocadas diretamente pelas aplicações (modo reativo) ou pelo administrador do ambiente (modo proativo). São elas:

– **addAppData(app_type, port_ini, port_fin)**: Atribui um tipo de aplicação a um intervalo de números de portas de protocolos da camada de transporte usado para a comunicação dos dados. *app_type* é uma referência a uma aplicação *Big Data* ou HPC tal como '$hadoop$', 'mpi' ou qualquer outra que seja reconhecida pela plataforma. *port_ini* é a primeira porta do intervalo. *port_fin* é a última porta do intervalo.

– **removeAppData(port_ini, port_fin)**: Remove a associação de um tipo de aplicação a seu intervalo de números de portas de protocolos da camada de transporte. *port_ini* é a primeira porta do intervalo. *port_fin* é a última porta do intervalo.

– **clearAppData()**: Exclui todas as associações entre os tipos de aplicações e seus intervalos de números de portas de protocolos da camada de transporte.

Para o mapeamento entre os tipos de aplicações e seus algoritmos de roteamento, a API concebida propõe ao menos duas funções. Idealmente, essas funções são usadas pelo administrador do ambiente. São elas:

– **setAlgAppMapping(app_type, alg)**: Mapeia um tipo de aplicação à sua estratégia de roteamento mais adequada no ambiente de execução. *app_type* é uma referência a uma aplicação *Big Data* ou HPC tal como '$hadoop$', 'mpi' ou qualquer outra que seja reconhecida pela plataforma. *alg* é uma referência a um algoritmo de roteamento tal como 'stp', '$ecmp$' ou qualquer outro que seja implementado na plataforma.

– **setAlgDefault(alg)**: Configura o algoritmo de roteamento padrão a ser aplicado aos fluxos de pacotes de dados para os quais não existe mapeamento previamente definido. *alg* é uma referência a um algoritmo de roteamento tal como 'stp', '$ecmp$' ou qualquer outro que seja implementado na plataforma.

A Figura 1 exibe uma representação, em alto nível, da arquitetura da plataforma SDN convergente com uma topologia multicaminhos. Sua implantação é baseada em módulos de *software* de rede que implementam os mecanismos essenciais da proposta. Os principais componentes são aqueles relacionados à descoberta dos múltiplos caminhos, à identificação do tráfego e à seleção das rotas de acordo com a estratégia de roteamento previamente definida. Essa estratégia é configurada pelo administrador com a ajuda da API descrita. Módulos de encaminhamento de pacotes, de coleta de estatísticas de rede, de processamento da API, etc, implementam as demais funcionalidades. Na Figura 1 também é possível notar o *cluster* de nós de trabalho onde são executadas as aplicações *Big Data* e HPC em paralelo. Essas aplicações podem chamar as funções da API diretamente para informar ao controlador SDN seus respectivos tipos e portas de rede.

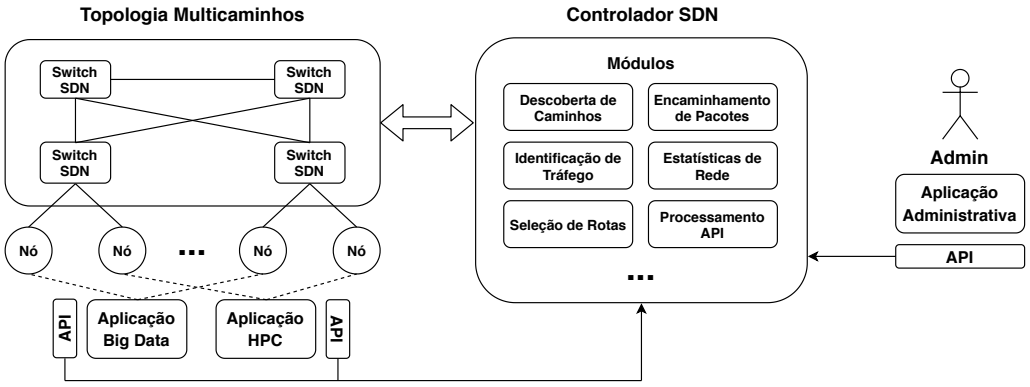


Figura 1. Arquitetura da plataforma SDN para ambientes convergentes *Big Data* e HPC.

5. Avaliação de Configurações de Rede para Aplicações Hadoop

Em um contexto de plataforma convergente e eficiente para *Big Data* e HPC, a infraestrutura de rede assume papel crítico. Assim, buscamos determinar a melhor configuração de rede para cada tipo de aplicação nesse ambiente integrado. Nesta seção, apresentamos uma análise de aplicações MapReduce Hadoop. O mesmo tipo de avaliação para aplicações MPI foi conduzido anteriormente em [Oliveira et al. 2018]. Na Seção 6, apresentamos brevemente essa avaliação.

5.1. Cenário Considerado

A Figura 2 apresenta o cenário de avaliação considerado neste trabalho. Ele representa uma arquitetura SDN formada por um *cluster* de pequena escala com uma topologia física multicaminhos *fat-tree 4-port, 3-tree*, isto é, com *switches* de quatro portas e uma árvore de três níveis (acesso, distribuição e núcleo). Arquiteturas *fat-tree* são utilizadas em inúmeras plataformas reais, incluindo os supercomputadores SDumont e Summit. A rede L2 possui 20 *switches* ($s1..s20$) gerenciados por um controlador SDN (não exibido na Figura 2). O cenário ainda é composto por 16 *hosts* ($h1..h16$) e um servidor ($srv1$), o qual exerce o papel de servidor de arquivos que pode ser compartilhado pelos nós. Nessa figura também é possível observar a representação dos quatro PODs (*Point of Delivery*).

Esse ambiente reduzido captura todos os aspectos relevantes à nossa avaliação. Topologias similares são usadas em trabalhos relacionados (e.g., [Bhatia et al. 2017]). A arquitetura atende os requisitos de um ambiente típico HPC, visto que os *hosts* formam um sistema de memória distribuída, compartilhando dados através de $srv1$. O cenário ainda encaixa-se em uma arquitetura típica de ambientes *Big Data* (i.e., sem compartilhamento), pois os *hosts* possuem capacidade própria de armazenamento. A comunicação de dados dispõe, via de regra, de múltiplos caminhos entre os pares de nós. O número de caminhos mínimos entre esses pares varia em relação às suas posições na rede: nós conectados ao mesmo *switch* de acesso não possuem caminhos redundantes; nós conectados a *switches* de acesso diferentes dentro do mesmo POD dispõem de dois caminhos mínimos entre eles; nós conectados a PODs distintos possuem quatro caminhos mínimos diferentes.

5.2. Metodologia de Avaliação

Implementamos o ambiente de rede descrito na Subseção 5.1 no emulador Mininet versão 2.3.0d1, com os comutadores executando o Open vSwitch 2.0.2 e enlaces limitados a

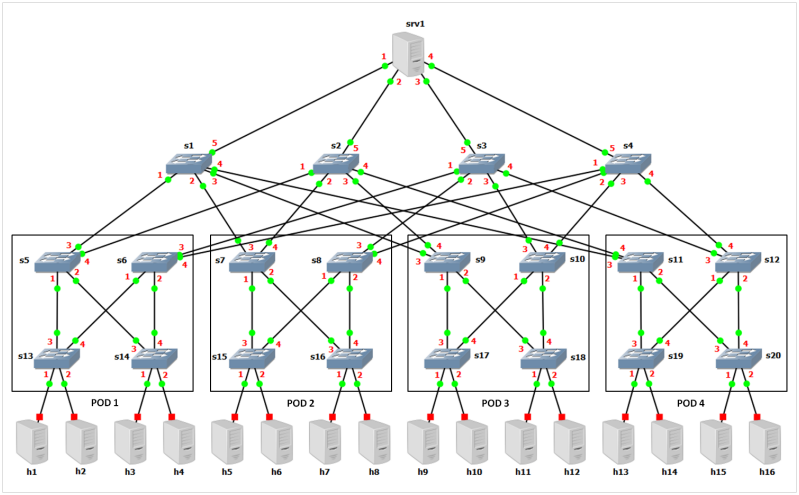


Figura 2. Cenário de avaliação considerado.

1 Gbps. Como controlador SDN, empregamos o POX *carp* utilizando o OpenFlow 1.0. Montamos o ambiente em um servidor 64 bits com duas CPUs Intel Xeon E5520 2,27 GHz, 16 núcleos, memória RAM de 48 GB, HD de 1 TB e SO Ubuntu 14.10.

Preparamos o modelo de testes visando avaliar um ambiente de execução de aplicações MapReduce. Para estabelecer a melhor configuração de rede para essas aplicações, analisamos o cenário a partir de quatro métodos simples de seleção de rotas. Propostos originalmente no nosso trabalho anterior ([Oliveira et al. 2018]), os mecanismos de roteamento são baseados em algoritmos comumente utilizados em redes de comutação de pacotes com múltiplas rotas. São eles:

• **"stp"**: Seleciona sempre um mesmo caminho (e.g, o primeiro) dentre o conjunto de caminhos mínimos disponíveis. Ele é inspirado no protocolo *spanning tree* ao ignorar as demais rotas, considerando-as como redundantes.

• **"traffic"**: Seleciona o caminho mínimo menos congestionado entre uma origem e um destino. Para tanto, dentre o conjunto de caminhos mínimos, é calculada a menor taxa de tráfego total instantânea dos seus enlaces.

• **"ecmp"**: Seleciona o caminho mínimo através de uma função de *hash*, como por exemplo a função de módulo, com o objetivo de balancear o tráfego. A chave da função pode ser formada por uma n-tupla composta por campos de cabeçalho do fluxo de pacotes ou pelo próprio valor individual do campo identificador do fluxo que o SDN fornece. Esse algoritmo é inspirado no algoritmo ECMP.

• **"isolated"**: Seleciona o caminho mínimo por meio de uma função de *hash* e o "isola" logica e temporariamente, de forma que a rota permaneça exclusiva para o fluxo, ficando indisponível para seleção pelos demais fluxos. O isolamento é condicionado à existência de outras rotas para os fluxos seguintes. Esse algoritmo é uma variação do "ecmp".

Para a experimentação, utilizamos a ferramenta MRemu [Neves et al. 2015], um *framework* baseado em emulação para pesquisa em rede usando cargas de trabalho MapReduce. MRemu usa traços de rede, criados a partir de execuções de aplicações Hadoop em *cluster* reais, como base para produzir cargas de tráfego emuladas no Mininet. Internamente, essas cargas são geradas pela ferramenta iPerf. MRemu abstrai

o processamento nos nós de trabalho do *cluster*, focando no tráfego de rede. Assim, o ambiente emulado não necessita de grande capacidade de processamento.

Para o projeto das simulações, selecionamos três traços de rede, os quais variam em função do número de tarefas *map*/*reduce*, produzindo quantidades distintas de transferências na rede[4]: (a) 128x4 (128 *maps*/4 *reduces* – 512 transferências); (b) 192x16 (192 *maps*/16 *reduces* – 3072 transferências); e (c) 256x16 (256 *maps*/16 *reduces* – 4096 transferências). Os traços usados geram tráfego entre os 16 *hosts* do cenário. A partir desse planejamento, executamos três sessões de testes, onde para cada sessão somente um traço de rede foi usado. Cada configuração foi executada utilizando os algoritmos descritos anteriormente: (1) "stp"; (2) "traffic"; (3) "ecmp"; e (4) "isolated". Em cada rodada de testes efetuamos 30 execuções do MRemu a partir do *host* $h1$, com intervalos de 30 segundos entre elas. Definimos o tempo de execução do emulador como variável de resposta. Os resultados são médias obtidas com intervalos de confiança de 95%.

5.3. Resultados

A Figura 3 apresenta, para cada traço de rede, o tempo de execução do emulador de aplicações Hadoop e os respectivos intervalos de confiança em função do algoritmo escolhido[5]. Nota-se uma influência do processo de seleção de caminhos no desempenho da comunicação de dados, para todas as medições. Na sessão de testes com o traço 128x4 (Figura 3(a)) essa influência é relativa, com os algoritmos "traffic", "ecmp" e "isolated" apresentando tempos de execução bem próximos entre si (157,52 s, 156,10 s e 157,02 s, respectivamente). Em relação ao algoritmo "stp", com tempo de execução de 188,69 s, esses algoritmos alcançaram resultados significativamente melhores. De fato, o ganho do melhor algoritmo ("ecmp") em relação ao pior ("stp") foi de mais de 17%.

As Figuras 3(b) e 3(c) mostram claramente a relevância da definição de um método de roteamento apropriado no desempenho da rede. Como indica a Figura 3(b), na sessão de testes com o traço 192x16, o ganho do "ecmp" (tempo de 137,71 s) em comparação com o "stp" (tempo de 177,49 s) foi superior a 22%. Nessa sessão, o algoritmo "ecmp" obteve tempos de execução 10% menores em relação ao algoritmo "traffic" (tempo de 153,02 s) e quase 3% menores em comparação com o algoritmo "isolated" (tempo de 141,27 s). Por sua vez, a sessão de testes com o traço 256x16 (Figura 3(c)) também revelou diferenças relevantes de tempos de execução entre os algoritmos "ecmp" (294,33 s) e "stp" (335,30 s), o que representa uma melhoria acima de 12%. Nesses mesmos testes, os algoritmos "traffic" e "isolated" também proporcionaram melhores desempenhos, com tempos de execução de 304,79 s e 297,28 s, respectivamente.

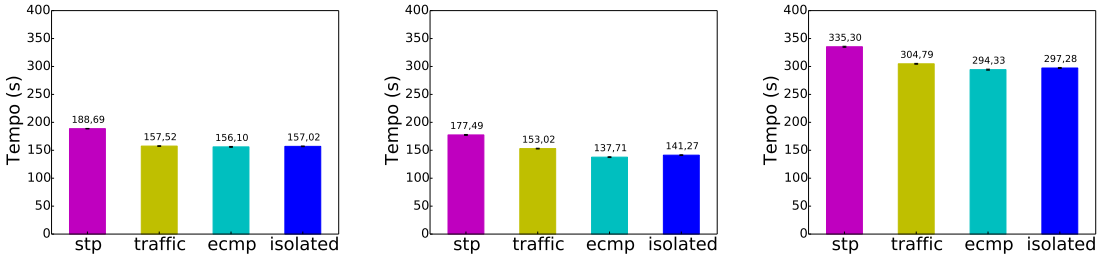
Em resumo, para o cenário de avaliação descrito nesta Seção 5, o algoritmo de roteamento "ecmp", que utiliza a estratégia de seleção de caminhos por meio de funções de *hash*, apresentou o melhor desempenho em todas as sessões de testes.

6. Avaliação da Plataforma Proposta Utilizando Aplicações Reais

Nesta seção, mostramos a viabilidade de uso da plataforma de rede convergente proposta, através da análise de desempenho de aplicações reais de computação paralela. Para tanto, avaliamos o uso concorrente de aplicações MapReduce Hadoop e MPI.

[4]Disponíveis em https://github.com/mvneves/mremu

[5]Os dados resultantes das simulações e os códigos-fontes dos protótipos dos componentes de *software* desenvolvidos para as avaliações estão disponíveis em https://github.com/netlabufjf/sdn-convergente



(a) Traço 128x4 (b) Traço 192x16 (c) Traço 256x16

Figura 3. Médias dos tempos de execução do emulador de aplicações Hadoop.

6.1. Configurações de Rede

Nesta análise, configuramos mecanismos de identificação de tráfego proativos baseados na API proposta. Associamos o tipo de aplicação '*hadoop*' às portas 13562, 9000, 50010 e 50020, e aos intervalos de portas 8030-8040 e 8480-8490, respeitando padrões de um ambiente Hadoop. Associamos também o tipo de aplicação '*mpi*' à porta 22 e ao intervalo de portas 1024-1044 (padrão do parâmetro *btl_tcp_port_min_v4* do Open MPI). De acordo com análises prévias durante nossos testes preliminares, essas configurações permitem ao controlador SDN reconhecer mais de 86% dos fluxos gerados por uma aplicação Hadoop e quase 100% dos fluxos gerados por uma aplicação Open MPI. O restante refere-se a fluxos de pacotes que usam portas dinâmicas aleatórias, impossibilitando a sua associação.

Aplicamos à plataforma de rede os algoritmos de roteamento mais adequados a esse cenário de testes, tomando como base nossas avaliações anteriores. Assim, mapeamos a aplicação Hadoop ao algoritmo "ecmp", conforme análise da Seção 5. Por sua vez, para a aplicação MPI, aplicamos o algoritmo "isolated", tendo em vista resultados apresentados em [Oliveira et al. 2018], onde avaliamos o desempenho dessas aplicações medindo, entre outros, a vazão de dados na rede. Nesse trabalho, executamos o *benchmark* MPI HPCC em paralelo usando três grupos de *hosts*, contendo 4, 8 e 16 nós. A Tabela 1 resume os valores médios das taxas de vazão obtidas naquela experimentação.

Tabela 1. Médias das taxas de vazão (em MB/s) da análise de aplicações MPI (Fonte: [Oliveira et al. 2018]).

	stp	traffic	ecmp	isolated
4 Nós	63,17	62,51	64,25	**67,99**
8 Nós	45,65	50,57	52,59	**57,75**
16 Nós	27,31	34,83	34,94	**38,07**

Para os fluxos de pacotes de dados não associados a nenhuma das aplicações paralelas, configuramos um algoritmo de roteamento comum e que minimiza o custo computacional da plataforma. Nesse caso, aplicamos o "stp", que comporta-se da mesma forma que o *Spanning Tree Protocol* e suas variações, os quais são habilitados por padrão na imensa maioria dos comutadores de rede corporativos/profissionais.

6.2. Metodologia de Avaliação

Aqui, consideramos a mesma arquitetura SDN utilizada nas demais avaliações. No entanto, para a simulação do ambiente, utilizamos o Containernet [Peuster et al. 2016],

um *fork* do Mininet que permite usar contêineres Docker como *hosts* em redes emuladas. Containernet supera uma restrição do Hadoop diante da qual ele exige que cada nó de trabalho do *cluster* tenha um *hostname* separado, o que o impossibilita de ser executado nos *hosts* virtuais do Mininet [Qiao et al. 2016]. Devido às maiores exigências computacionais desse ambiente, executamos a experimentação em um servidor mais robusto, com duas CPUs Intel Xeon E5-2620 2,40 GHz 64 bits, 24 núcleos, memória RAM de 64 GB, HD de 4 TB e SO Suse Linux Enterprise Server (SLES) 12.

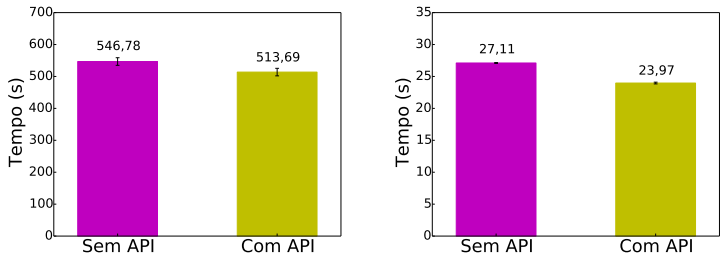
Na avaliação MapReduce, utilizamos o Hadoop 2.7.2 e sua aplicação paralela de ordenação de dados Terasort. Para a geração desses dados, usamos a aplicação Teragen. Com Terasort/Teragen é possível avaliar o desempenho geral do Hadoop, pois ele combina as capacidades de distribuição (HDFS) e de processamento dos dados (*map*/*reduce*) no *cluster*. O *host* $h1$ atuou como NameNode/ResourceManager e os demais hosts ($h2..h16$) exerceram o papel de DataNode/NodeManager. Na avaliação MPI, utilizamos o Open MPI 1.6.5 para a execução paralela de um código de resolução numérica de equação de calor em três dimensões (Heat 3D). Essa aplicação caracteriza-se por um paralelismo de granularidade fina, que incrementa a comunicação de dados na rede.[6]

Para o projeto das simulações, estabelecemos as análises das duas aplicações considerando as seguintes condições: (a) Hadoop Terasort com conjunto de dados de 10 GB; e (b) MPI Heat 3D com matriz de 32x64x16. Nesses termos, ambas as aplicações geram um intenso tráfego de dados na rede, o que permite avaliar a plataforma em um contexto adequado. Para fins comparativos, executamos duas sessões de testes, onde em cada sessão a plataforma opera de duas formas diferentes: (1) Sem API – controlador SDN ignora as informações configuradas e seleciona os caminhos aplicando um algoritmo de roteamento genérico; e (2) Com API – controlador SDN identifica a aplicação através da API e aplica o roteamento mais adequado. Na sessão sem API, o algoritmo utilizado foi o "stp". Em cada rodada de testes efetuamos 15 execuções de cada aplicação a partir do *host* $h1$, com intervalos de 30 segundos entre elas. Definimos o tempo de execução como variável de resposta. Os resultados médios possuem níveis de confiança de 95%.

6.3. Resultados

A Figura 4 exibe os resultados obtidos nos testes das aplicações Hadoop Terasort e MPI Heat 3D, considerando as duas formas de operação da plataforma (sem API e com API). Apesar da sobrecarga adicionada pelos mecanismos de identificação de tráfego e seleção de rotas, constatamos que o uso da API assegurou uma diminuição nos tempos médios de execução dessas aplicações, sem gerar efeitos colaterais na rede. De fato, com a API, a topologia lógica pode ser alterada para cada fluxo, de modo que a comunicação de dados é maximizada através do melhor uso dos múltiplos caminhos de acordo com os perfis de tráfego de cada aplicação. Como aponta a Figura 4(a), na sessão de testes com Hadoop, a plataforma operando com API completou a execução da aplicação em um tempo médio de 513,69 s, o que representa uma redução em torno de 6% em comparação com o cenário sem API, que a executou em um tempo médio de 546,78 s. Na sessão de testes com a aplicação MPI (Figura 4(b)), a melhoria de desempenho foi mais significativa. Nessa sessão, a execução sobre a plataforma com API (23,97 s) foi acima de 11% mais rápida em relação à plataforma sem API (27,11 s).

[6]Para fins de reprodutibilidade, os arquivos com os parâmetros de configuração de ambas as aplicações também estão disponíveis em https://github.com/netlabufjf/sdn-convergente



(a) Hadoop Terasort (b) MPI Heat 3D

Figura 4. Médias dos tempos de execução das aplicações Hadoop e MPI.

7. Conclusões e Trabalhos Futuros

Este artigo apresentou uma plataforma SDN cujo propósito é suprir, de forma integrada, os requisitos de desempenho das aplicações típicas de ambientes *Big Data* e HPC. A plataforma atende esses requisitos maximizando a comunicação dos dados na rede. Para tal, é oferecida uma API que permite a identificação do tráfego e o emprego das melhores estratégias de roteamento para cada aplicação paralela. Diante disso, evidenciamos a importância da definição das configurações de rede mais adequadas a cada perfil de tráfego em determinado ambiente. Simulamos a arquitetura proposta em um cenário específico e examinamos aplicações Hadoop por meio de um emulador. A partir dessa avaliação e de uma análise anterior de aplicações MPI, simulamos aplicações paralelas reais e avaliamos a viabilidade e a utilidade da plataforma convergente. Os resultados mostram que a API permite o ajuste dinâmico da infraestrutura de rede, tornando-a mais eficiente e contribuindo para a diminuição dos tempos de execução dessas aplicações.

Como trabalhos futuros, pretendemos desenvolver mecanismos inteligentes de identificação e classificação de tráfego de rede para ambas as aplicações. Planeja-se, ainda, avaliar a utilização de ferramentas de "fatiamento" de rede como solução complementar à implementação da plataforma, além de executar experimentos reais.

Agradecimentos

Os autores agradecem o apoio de CAPES, CNPq, FAPEMIG, FAPERJ e FAPESP.

Referências

Alsmadi, I., Khamaiseh, S., and Xu, D. (2016). Network parallelization in HPC clusters. In *Proc. of the IEEE CSCI*.

Bhatia, S., Sinha, Y., Chalapathi, G., and Kumar, R. (2017). MPI aware routing using SDN. *Poster Presented at the 26th HPDC*.

Fox, G., Qiu, J., Jha, S., Ekanayake, S., and Kamburugamuve, S. (2015). Big Data, simulations and HPC convergence. In *Big Data Benchmarking*, pages 3–17. Springer.

Kreutz, D., Ramos, F., Veríssimo, P., Rothenberg, C., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proc. of the IEEE*, 103(1):14–76.

Liang, F. and Lau, F. (2016). BAShuffler: Maximizing network bandwidth utilization in the shuffle of YARN. In *Proc. of the ACM HPDC*, pages 281–284.

LNCC (2018). Configuração do SDumont. https://sdumont.lncc.br/machine.php?pg=machine#. Accessed: December, 2018.

Mattson, T., Sanders, B., and Massingill, B. (2004). *Patterns for Parallel Programming*. Pearson Education.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Mellanox (2015). SX6536 Product Brief. http://www.mellanox.com/related-docs/prod_ib_switch_systems/PB_SX6536.pdf. Accessed: December, 2018.

Narayan, S., Bailey, S., and Daga, A. (2012). Hadoop acceleration in an OpenFlow-based cluster. In *Proc. of the IEEE SCC*.

Neves, M., De Rose, C., and Katrinis, K. (2015). MRemu: An emulation-based framework for datacenter network experimentation using realistic MapReduce traffic. In *IEEE MASCOTS*, pages 174–177.

Oliveira, A., Vieira, A., Gomes, A., and Ziviani, A. (2018). Análise de desempenho de rede para aplicações MPI em infraestruturas SDNs convergentes para HPC e Big Data. In *Proc. of the WSCAD*, pages 397–408. SBC.

ORNL (2018). SUMMIT – Oak Ridge National Laboratory's next High Performance Supercomputer. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/. Accessed: December, 2018.

Peuster, M., Karl, H., and Van Rossem, S. (2016). MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments. *IEEE NFV-SDN*.

Ponce, L., Santos, W., Meira-Jr, W., and Guedes, D. (2018). Extensão de um ambiente de computação de alto desempenho para o processamento de dados massivos. In *Proc. of the SBRC*.

Porto, F. (2017). Algoritmos e modelos de programação em Big Data XXXVI Jornada de Atualização em Informática (JAI). In *Proc. of the SBC Congress*.

Qiao, Y., Wang, X., Fang, G., and Lee, B. (2016). Doopnet: An emulator for network performance analysis of Hadoop clusters using Docker and Mininet. In *IEEE ISCC*, pages 784–790.

Qin, P., Dai, B., Huang, B., and Xu, G. (2015). Bandwidth-aware scheduling with SDN in Hadoop: A new trend for Big Data. *IEEE Systems Journal*.

Takahashi, K., Date, S., Khureltulga, D., Kido, Y., Yamanaka, H., Kawai, E., and Shimojo, S. (2018). Unisonflow: A software-defined coordination mechanism for message-passing communication and computation. *IEEE Access*, 6:23372–23382.

Trois, C., Bona, L., Del Fabro, M., Martinello, M., Bidkar, S., Nejabati, R., and Simeonidou, D. (2017). Softening up the network for scientific applications. In *2017 25th Euromicro International Conference on PDP*, pages 108–115. IEEE.

Webb, K., Snoeren, A., and Yocum, K. (2011). Topology switching for data center networks. *Hot-ICE*, 11.