Análise de Desempenho de Rede para Aplicações MPI em Infraestruturas SDNs Convergentes para HPC e Big Data

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. *Extraction of knowledge from large volumes of data imposes challenges to current computing platforms. Although it appears to be natural, the use of HPC platforms by Big Data applications involves the suitability of several of its elements. For instance, the data network infrastructure needs to be efficient and flexible to fit the typical applications of these environments. The SDN paradigm is adequate in this context because of its global vision and its higher level of network programmability. As a consequence, network management is simplified, making it more flexible. In this article, we present an SDN-based communication platform capable of supplying, in a convergent way, HPC and Big Data application requirements. We highlighted the importance of defining the most appropriate network configurations for each traffic profile. We present four routing strategies in a convergent environment and analyze the performance of MPI applications. The results of simulations show that the data throughput in the convergent environment can be up to 39% better, depending on the strategy selected.*

Resumo. *A extração de conhecimento de grandes volumes de dados impõe desafios às plataformas computacionais atuais. Embora pareça ser natural, a utilização de plataformas HPC por aplicações de Big Data envolve a adequação de vários de seus elementos. Por exemplo, a infraestrutura de rede de dados precisa ser eficiente e flexível para se ajustar às aplicações típicas desses ambientes. O paradigma SDN é adequado a esse contexto por sua visão global e seu maior nível de programabilidade da rede. Como consequência, a gerência da rede é simplificada, tornando-a mais flexível. Neste artigo, apresentamos uma plataforma de comunicação baseada em SDN capaz de suprir, de forma convergente, os requisitos de aplicações HPC e Big Data. Evidenciamos a importância da definição das configurações de rede mais adequadas a cada perfil de tráfego. Apresentamos quatro estratégias de roteamento em um ambiente convergente e analisamos o desempenho de aplicações MPI. Os resultados de simulações mostram que a vazão de dados no ambiente convergente pode ser até 39% melhor, dependendo da estratégia selecionada.*

1. Introdução

Big Data vem se consolidando como um dos maiores fenômenos em termos de modelo de negócios e de ciência em quase todos os domínios [Economist 2010]. Empresas, órgãos

públicos e instituições de pesquisa vêm se beneficiando amplamente das abordagens de *Big Data Analytics*, extraindo informações valiosas a partir dos grandes volumes de dados gerados no âmbito dessas organizações. Naturalmente, manipular esses dados impõe enormes desafios, tanto em termos de gerenciamento de memória quanto de processamento e acesso aos dados.

Plataformas de computação de alto desempenho (*High Performance Computing* – HPC) oferecem grande capacidade de processamento que pode ser aproveitada por uma vasta gama de aplicações, incluindo soluções para tratar grandes volumes de dados. Embora a utilização dessas plataformas HPC por aplicações de *Big Data* pareça ser natural, o que se observa, de fato, é um imenso descasamento entre essas arquiteturas. Realmente, explorar recursos de HPC para atender soluções típicas de *Big Data* demanda o tratamento de inúmeros problemas. A integração entre esses mundos envolve o estudo e a adequação de vários elementos desses sistemas. A infraestrutura de comunicação de dados, por exemplo, precisa ser eficiente e flexível para se adequar aos diferentes perfis das aplicações que operam sobre a rede. No entanto, as redes de dados tradicionais possuem controles complexos, o que as tornam "ossificadas". Logo, o ajuste das redes de comunicação aos requisitos de cada uma dessas aplicações demanda um alto custo administrativo, o que inviabiliza, na prática, o uso dessas infraestruturas de forma convergente e eficiente. Nesse aspecto, há novos paradigmas de redes que podem favorecer a integração dos ambientes HPC e *Big Data*, como as Redes Definidas por Software (*Software-Defined Networks* – SDN).

Nesse cenário, a busca por uma plataforma de comunicação convergente pressupõe o atendimento das exigências de rede de aplicações paralelas que utilizam as APIs típicas de ambientes HPC (e.g., *Message Passing Interface* – MPI), e de aplicações típicas de ambientes *Big Data*, executadas sobre arcabouços MapReduce (e.g., Hadoop, Spark). Espera-se também que essa plataforma implemente mecanismos de classificação de tráfego, de modo que a infraestrutura se ajuste aos respectivos padrões de comunicação através de configurações proativas ou reativas. Assim, a rede torna-se ciente das aplicações, adequando sua configuração apropriadamente a cada perfil de tráfego.

Neste artigo, apresentamos uma plataforma baseada em SDN cujo objetivo é fornecer uma infraestrutura de rede de dados flexível capaz de suprir, de forma convergente, os requisitos de desempenho de aplicações *Big Data* e de HPC. A plataforma atende esses requisitos otimizando a comunicação dos dados provenientes dos processos paralelos. Para tanto, o controlador SDN identifica o tráfego da aplicação e utiliza a estratégia de roteamento mais adequada ao perfil desse tráfego. Dessa maneira, o controlador seleciona o melhor caminho para os pacotes conforme essa estratégia e configura as respectivas regras de encaminhamento nas tabelas de fluxos dos dispositivos de rede. Nessa circunstância, nós propomos quatro algoritmos simples de seleção de rotas para ambientes de rede multicaminhos, característicos de topologias HPC.

Nós simulamos um ambiente SDN Ethernet com uma topologia multicaminhos para analisar o desempenho da rede perante o tráfego gerado por um *benchmark* MPI. A partir de diferentes medições utilizando um cenário específico, nós constatamos diferenças de vazão de até 39% na comparação entre as estratégias de roteamento propostas, ao mesmo tempo que a latência dos pacotes permaneceu estável. Os resultados atuais realçam a influência do processo de seleção de rotas na eficiência da rede e a

importância da definição da melhor configuração de rede para cada perfil de tráfego. Embora ainda não tenham sido realizados testes de forma conjunta, ou seja, considerando ambas as aplicações MPI e *Big Data*, as análises mostram a aplicabilidade do paradigma SDN em infraestruturas Ethernet como solução para maximizar a comunicação em redes de alto desempenho, mitigando problemas comuns dessa tecnologia como variação estatística do retardo e baixa capacidade efetiva. Nesse contexto, este trabalho contribui também para que se abra uma nova perspectiva no estudo de soluções convergentes, no sentido de se ponderar sobre a implantação de infraestruturas Ethernet, significativamente mais baratas, para atender soluções de HPC.

No restante deste artigo, a Seção 2 discute os trabalhos relacionados. Na Seção 3, mostramos alguns dos conceitos de computação avançada. A Seção 4 explora os aspectos da infraestrutura convergente. A descrição da análise de desempenho das aplicações MPI é apresentada na Seção 5. A Seção 6 mostra as conclusões do artigo e os trabalhos futuros.

2. Trabalhos Relacionados

Com o amadurecimento do paradigma SDN, o estudo das redes de dados nos ambientes de computação avançada ganhou novos contornos, especialmente na otimização das aplicações típicas de HPC e dos *frameworks Big Data*. Como exemplo, os trabalhos de [Bhatia et al. 2017] e [Alsmadi et al. 2016] empregam SDN para aprimorar o processamento MPI. No primeiro, o controlador utiliza estatísticas de fluxo para atualizar um grafo de topologia de rede usado por um algoritmo de roteamento adaptativo na seleção dinâmica dos caminhos. No segundo, informações de rede obtidas pelo controlador são usadas pelo orquestrador de recursos para selecionar os nós mais adequados ao processamento de cada tarefa.

Há ainda na literatura algumas pesquisas focadas na melhoria dos ambientes típicos de *Big Data*. Por exemplo, [Qin et al. 2015] discutem o processamento de dados em larga escala no Hadoop, uma das mais conhecidas implementações *open source* do modelo de programação MapReduce. No artigo, os autores propõem um agendador de tarefas consciente da largura de banda utilizando o modelo SDN. Essa abordagem, chamada BASS, é capaz não somente de garantir a localidade dos dados a partir de uma visão global, mas também pode eficientemente atribuir tarefas de maneira otimizada.

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. Nesse estudo, o COMPSs é integrado ao HDFS, sistema de arquivos distribuído bastante utilizado nos cenários de *Big Data*. 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. Ainda nesse contexto, [Zhang et al. 2014] propõem um algoritmo ECMP (*Equal-Cost MultiPath*) otimizado por meio de SDN que torna possível ajustar dinamicamente o encaminhamento de fluxos, com o objetivo de melhorar a vazão nas redes de *data center*.

De forma geral, os trabalhos que exploram as capacidades de SDN nos ambientes

HPC e *Big Data* o fazem de maneira independente, como evidenciam os artigos de [Bhatia et al. 2017], [Alsmadi et al. 2016] e [Qin et al. 2015]. Nossa proposta, por sua vez, tira proveito da flexibilidade proporcionada pelo paradigma SDN para prover uma infraestrutura de comunicação verdadeiramente convergente e, acima de tudo, eficiente. Além disso, nossa proposta 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] e [Qin et al. 2015], que propõem soluções sob um ponto de vista no qual a aplicação se torna consciente da rede, ou seja, a aplicação se adapta às condições da rede. Por fim, no contexto da definição dos métodos de roteamento mais apropriados a cada padrão de comunicação, fundamental na proposta de convergência delineada neste artigo, nossa análise de desempenho considera o perfil de tráfego das aplicações MPI, com foco na plataforma convergente, ao contrário dos artigos de [Webb et al. 2011] e [Zhang et al. 2014], que avaliam padrões de tráfego genéricos.

3. Ecossistemas de Computação Avançada

Esta seção apresenta alguns dos conceitos por trás dos ecossistemas de computação avançada, incluindo suas arquiteturas e modelos de programação paralela.

3.1. Arquiteturas Paralelas

Sistemas que demandam grande poder computacional, como aplicações científicas e até aplicações/ferramentas de negócios, utilizam computação paralela para processarem seus dados no menor tempo possível. Sistemas paralelos são criados combinando múltiplos elementos de processamento em um único sistema maior. Desde a metade dos anos 1990, tecnologias como *multithreading* e *multicore* tornaram esses sistemas amplamente disponíveis. A maioria dos sistemas paralelos modernos encaixam-se na arquitetura MIMD (*Multiple Instruction, Multiple Data*), tipicamente classificada de acordo com a organização de memória: compartilhada e distribuída [Mattson et al. 2004].

Em sistemas de memória compartilhada, processos compartilham um único espaço de memória. Esses sistemas também são classificados em SMP (*symmetric multiprocessors*), NUMA (*nonuniform memory access*) e sua variação ccNUMA (*cache-coherent* NUMA). Sistemas de memória distribuída são aqueles onde cada processo tem seu próprio espaço de endereçamento e a comunicação com os demais processos ocorre mediante o envio e o recebimento de mensagens (*message passing*). Eles são tradicionalmente divididos em MPP (*massively parallel processors*) e *clusters*, que são sistemas mais baratos e, portanto, cada vez mais comuns. Arquiteturas paralelas também podem se apresentar como sistemas híbridos e *grids* [Mattson et al. 2004].

As atuais arquiteturas de computação de alto desempenho (HPC) possuem a tendência de migrar dos tradicionais sistemas SMP e MPP para sistemas de *cluster* de memória distribuída, com nós de computação conectados através de interconexões de alta largura de banda e baixa latência [Date et al. 2016]. As redes desses *clusters* são implantadas com diversas tecnologias e sobre topologias físicas multicaminhos como *fat-tree* e Torus, que oferecem ao mesmo tempo desempenho e redundância. Nesses ambientes, os nós compartilham o acesso aos dados através de um sistema de arquivos distribuído como, por exemplo, o sistema de arquivos paralelos *open source* Lustre.

No ecossistema *Big Data*, os dados a serem processados, a princípio, não cabem na memória principal de um único computador. Assim, o sistema de *cluster* de servidores

é visto como a plataforma padrão para esses ambientes [Porto 2017]. Suas aplicações típicas executam sob um modelo onde os arquivos são distribuídos pelos nós e os processos alocados a esses nós, através do qual se busca minimizar a transferência de dados. Para tanto, esses ambientes utilizam tradicionalmente o sistema de arquivos distribuídos HDFS (*Hadoop Distributed File System*). Considera-se que esses sistemas se encaixam em uma arquitetura sem compartilhamento, onde a tecnologia de rede predominante é a Ethernet.

3.2. Programação Paralela

Em programação paralela, cada arquitetura tem um modelo de programação apropriado. Os modelos mais comuns são baseados nas arquiteturas de memória compartilhada, de memória distribuída com passagem de mensagem, ou em uma combinação híbrida das duas [Mattson et al. 2004]. O OpenMP é um dos principais ambientes de programação para as arquiteturas de memória compartilhada, sendo frequentemente usado para adicionar paralelismo ao código sequencial, sobretudo em arquiteturas SMP.

O MPI é o modelo de programação paralela padrão para arquiteturas de memória distribuída, especialmente para os ambientes HPC. Por definição, MPI é uma especificação de interface de bibliotecas de passagem de mensagem. Nesse modelo, um processo empacota informação em uma mensagem e a envia a um outro processo. MPI possui um conjunto de APIs de alto nível que fornece rotinas para gerenciamento de processos e operações de comunicação coletiva. Criado nos anos 1990, sua especificação mais recente é a versão 4.0. Atualmente, MPI possui implementações *open source* como OpenMPI e MPICH, e implementações proprietárias como Intel MPI e Bull MPI.

Nos ambientes *Big Data*, i.e. em arquiteturas sem compartilhamento, é utilizado o modelo de programação funcional MapReduce. Nele, usuários especificam a computação 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 nessa programação, o sistema de tempo de execução subjacente automaticamente paraleliza a computação através do *cluster*. A partir do modelo MapReduce, surgiram os principais *frameworks* de computação intensiva de dados, entre eles o Hadoop, o Spark, e mais recentemente o Flink, todos atualmente mantidos pela fundação Apache.[1]

4. Infraestrutura de Comunicação Convergente

A proposta de plataforma de comunicação convergente é concebida sobre uma arquitetura HPC baseada em SDN capaz de atender, além das próprias aplicações HPC, aplicações típicas de *Big Data*. Essas aplicações são executadas em paralelo pelos nós da arquitetura. Esses nós trocam dados/mensagens através da rede. O controlador SDN, por sua vez, procura otimizar a comunicação na rede identificando o tráfego das aplicações e aplicando as estratégias de roteamento mais adequadas a cada perfil de tráfego. Por fim, a rede torna-se consciente da aplicação.

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

[1] https://www.apache.org

elemento controlador externo logicamente centralizado; e (iv) programação da rede através de aplicações de *software* executando no topo do modelo [Kreutz et al. 2015]. Conforme mostra a Figura 1, a interface entre o controlador SDN e os dispositivos da infraestrutura de rede (*southbound* API) é fornecida por protocolos seguros como o OpenFlow [McKeown et al. 2008]. O OpenFlow é considerado um padrão *de facto*, sendo utilizado pela maioria das plataformas atuais. A *northbound* API não é padronizada ainda. Assim, em 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. Uma implementação do protocolo bastante comum é o Open vSwitch.

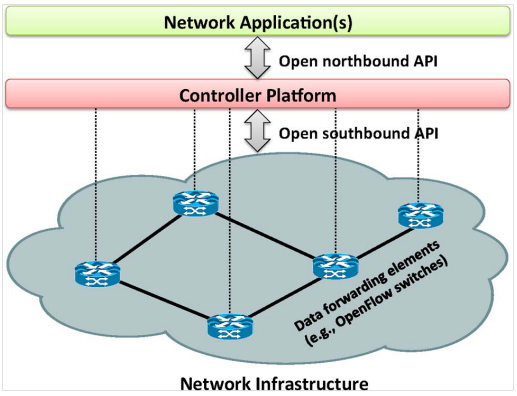


Figura 1. Visão simplificada de uma arquitetura típica SDN [Kreutz et al. 2015].

Na nossa concepção, a identificação do tráfego pode seguir tanto uma abordagem proativa quanto reativa. No modo proativo, as aplicações informam suas características ao controlador da rede através de APIs de programação disponibilizadas pela plataforma, o que exige modificações nas aplicações. No modo reativo, a rede toma ciência dos requisitos das aplicações de forma transparente. Nesse caso, o controlador efetua o reconhecimento de assinaturas, ou seja, de padrões explícitos dessas aplicações como, por exemplo, os números de portas dos protocolos da camada de transporte dos pacotes. O controlador pode também realizar a classificação do tráfego mediante metodologias baseadas em inspeção profunda de pacotes ou aprendizado de máquina, embora esses métodos sejam mais custosos computacionalmente [Qazi et al. 2013].

A escolha da estratégia de roteamento tem papel fundamental na rede. Definir os caminhos mais adequados a cada padrão de tráfego pode implicar em aumento de vazão e diminuição da latência dos pacotes. Para cada tipo de classificação, decidir corretamente a rota pode fazer com que se eleve o desempenho da plataforma convergente. Por exemplo, aplicações que utilizam o modelo MapReduce possuem um padrão de comunicação bem definido, com destaque para a movimentação dos dados entre os nós que computam as funções *map* e os que computam as funções *reduce* (fase de "embaralhamento"). No modelo MPI, o padrão depende da aplicação, com operações de comunicação coletiva geralmente prevalecendo sobre operações ponto-a-ponto. Apesar da dependência da aplicação, o sistema de passagem de mensagem apresenta um perfil peculiar.

Considerando, portanto, uma plataforma convergente SDN sobre um *cluster* com topologia multicaminhos, nós propomos quatro mecanismos de roteamento a serem utilizados na infraestrutura de rede desses ambientes. Eles são idealizados com a

finalidade de maximizar a comunicação dos dados ao mesmo tempo que tentam minimizar o custo computacional. Em termos gerais, as políticas de roteamento são baseadas em algoritmos comumente utilizados em redes de comutação de pacotes com múltiplas rotas.

O **Algoritmo 1**, aqui referenciado como **"stp"**, seleciona sempre o mesmo caminho (por exemplo, o primeiro) dentre o conjunto de caminhos mínimos disponíveis entre uma origem e um destino. Ele é inspirado no protocolo *spanning tree* no sentido de que ignora as demais rotas, considerando-as como redundantes. Seu custo computacional é baixo, porém desperdiça os múltiplos caminhos da topologia. Um exemplo de pseudocódigo para o algoritmo "stp" é apresentado a seguir:

Algoritmo 1: "stp"

```
   Entrada: Conjunto de caminhos mínimos
1  início
2  |  caminho mínimo ← primeiro caminho
3  fim
4  retorna caminho mínimo
```

O **Algoritmo 2**, aqui referenciado como **"traffic"**, seleciona o caminho mínimo menos congestionado entre uma origem e um destino. Para tal, dentre o conjunto de caminhos mínimos, é calculada a menor taxa de tráfego total instantânea dos seus enlaces. Seu custo computacional é alto, entretanto usufrui das rotas com maior largura de banda disponível. A seguir, é mostrado um pseudocódigo para esse algoritmo:

Algoritmo 2: "traffic"

```
   Entrada: Conjunto de caminhos mínimos
1  início
2  |  caminho mínimo ← primeiro caminho
3  |  taxa total menor ← ∞
4  |  para cada caminho ∈ caminhos mínimos faça
5  |  |  taxa do caminho ← 0
6  |  |  para cada comutador ∈ ao caminho faça
7  |  |  |  taxa do comutador ← bytes enviados/recebidos na porta de saída
8  |  |  |  taxa do caminho ← taxa do caminho + taxa do comutador
9  |  |  fim
10 |  |  se taxa do caminho ≤ taxa total menor então
11 |  |  |  taxa total menor ← taxa do caminho
12 |  |  |  caminho mínimo ← caminho
13 |  |  fim
14 |  fim
15 fim
16 retorna caminho mínimo
```

O **Algoritmo 3**, aqui referenciado como **"ecmp"**, seleciona o caminho mínimo através de uma função de *hash*. A chave da função pode ser formada por uma n-tupla composta por campos de cabeçalho do fluxo de pacotes — por exemplo, (IP de origem, IP de destino, ID do protocolo) — ou pelo próprio valor do campo identificador do fluxo que o SDN fornece. Para mapear a chave a um índice associado ao caminho, pode ser utilizada uma função simples como a de módulo. Seu custo computacional é baixo e os

multicaminhos são bem aproveitados, porém o uso de funções de *hash* não perfeitas pode gerar colisões e, consequentemente, caminhos ociosos. Esse algoritmo é inspirado no algoritmo ECMP [Thaler and Hopps 2000]. Um pseudocódigo é exibido adiante:

Algoritmo 3: "ecmp"

```
Entrada: Conjunto de caminhos mínimos
1  início
2  |  índice do caminho ← ID do fluxo % nº de caminhos mínimos
3  |  caminho mínimo ← caminhos mínimos [índice do caminho]
4  fim
5  retorna caminho mínimo
```

Por fim, o **Algoritmo 4**, aqui referenciado como **"isolated"**, é uma variação do Algoritmo 3. Ele seleciona o caminho mínimo por meio de uma função de *hash* e o "isola" temporariamente, de forma que a rota permaneça exclusiva para o fluxo, ficando indisponível para seleção pelos demais fluxos. Seu custo computacional também é baixo. Um pseudocódigo para esse algoritmo é apresentado a seguir:

Algoritmo 4: "isolated"

```
Entrada: Conjunto de caminhos mínimos
1  início
2  |  enquanto ∃ caminhos mínimos faça
3  |  |  índice do caminho ← ID do fluxo % nº de caminhos mínimos
4  |  |  caminho mínimo ← caminhos mínimos [índice do caminho]
5  |  |  se caminho mínimo ∉ caminhos isolados então
6  |  |  |  caminho mínimo ∪ caminhos isolados
7  |  |  |  vai para 14
8  |  |  fim
9  |  |  senão
10 |  |  |  caminhos mínimos ← caminhos mínimos - caminho mínimo
11 |  |  fim
12 |  fim
13 fim
14 retorna caminho mínimo
```

A proposta desses algoritmos baseou-se, essencialmente, em trabalhos publicados no domínio das redes de *data center*, como os artigos de [Webb et al. 2011] e [Zhang et al. 2014], descritos na Seção 2. Inclusive, [Webb et al. 2011] apontam que aplicações MapReduce adequam-se melhor a estratégias de seleção de caminhos que privilegiam a largura de banda disponível.

5. Análise de Desempenho de Aplicações MPI

No contexto da avaliação da plataforma de comunicação convergente entre HPC e *Big Data*, nós descrevemos nesta seção os detalhes de uma análise de desempenho de aplicações MPI, considerando os algoritmos descritos na Seção 4.

5.1. Cenário Considerado

A Figura 2 apresenta o cenário de avaliação considerado neste trabalho. Ele representa uma arquitetura HPC SDN formada por um *cluster* de servidores de memória distribuída

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). 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 compartilhado pelos nós. Nessa figura também é possível observar a representação dos quatro PODs (*Point of Delivery*).

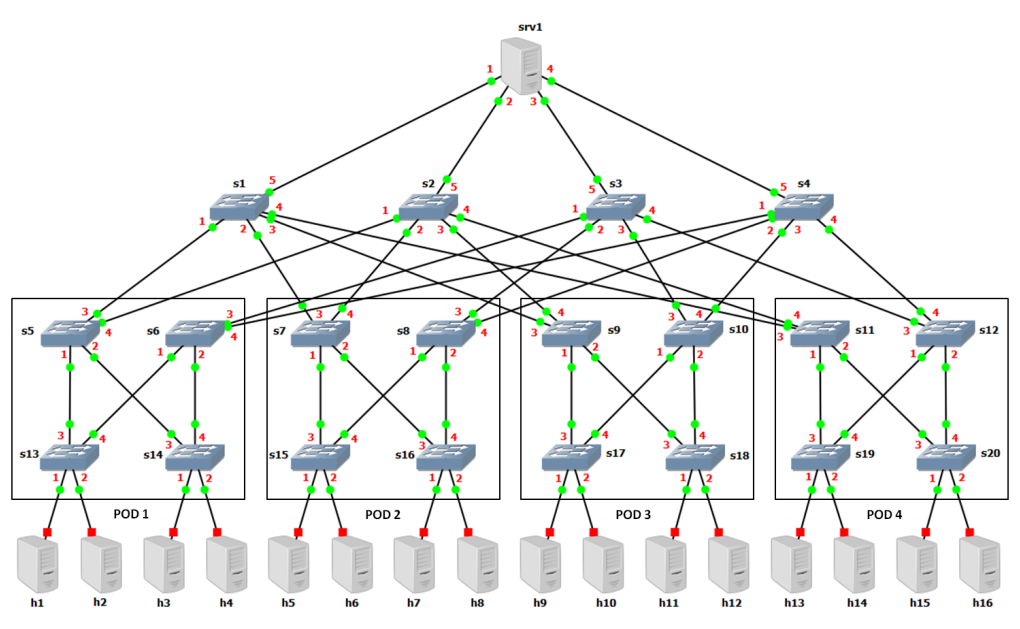


Figura 2. Cenário de avaliação considerado.

Esse cenário retrata um ambiente HPC reduzido mas, ainda assim, capaz de capturar todos os aspectos relevantes à nossa avaliação. Nele, as aplicações MPI são executadas em paralelo pelos nós ($h1..h16$). A comunicação entre os processos dispõe, quase sempre, 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

O ambiente de rede descrito na Seção 5.1 foi implementado no emulador Mininet versão 2.3.0d1 com os comutadores executando o Open vSwitch 2.0.2 e enlaces limitados a 1 Gbps. Como controlador SDN foi empregado o POX *carp* utilizando o OpenFlow 1.0. Todo o ambiente foi montado 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.

O modelo de testes foi preparado visando avaliar os quatro algoritmos previamente propostos. Um *benchmark* MPI foi executado em paralelo a partir do *host* $h1$ utilizando nós de diversos PODs. A ferramenta utilizada foi o HPCC (HPC *Challenge Benchmark*)[2] versão 1.5.0. A *suite* HPCC consiste de basicamente sete testes que incluem diferentes

[2]http://icl.cs.utk.edu/hpcc

métricas de desempenho. Entre eles, nós focamos em um conjunto de testes baseado no b_eff (*effective bandwidth benchmark*), o qual mede a vazão e a latência de vários padrões de comunicação simultâneos.

Para o projeto das simulações foram definidas duas variáveis de resposta, ambas medidas sob o ponto de vista do *host* $h1$: (i) vazão utilizada na comunicação; e (ii) latência dos pacotes. Tais métricas são importantes no âmbito das redes de ambientes HPC, as quais exigem interconexões de alta largura de banda (grande vazão) e baixa latência. Para as simulações paralelas, foram utilizados três tamanhos de grupos de *hosts*: (a) 4 nós ($h1$, $h3$, $h5$ e $h7$ – PODs 1 e 2); (b) 8 nós (*hosts* ímpares – todos os PODs); e (c) 16 nós (todos os *hosts*). A partir desse planejamento foram executadas três sessões de testes, onde para cada sessão somente um grupo de nós foi usado. Cada configuração foi executada utilizando os 4 algoritmos propostos: (1) "stp"; (2) "traffic"; (3) "ecmp"; e (4) "isolated". Foram utilizadas as configurações padrão do HPCC: medições de vazão com mensagens de 2000000 *bytes*; medições de latência com mensagens de 8 *bytes*; e comunicação lógica por anel realizada em ambas as direções. Em cada rodada de testes foram efetuadas 30 execuções do HPCC, com intervalos de 30 segundos entre elas. Os resultados são médias obtidas com intervalos de confiança de 95%.

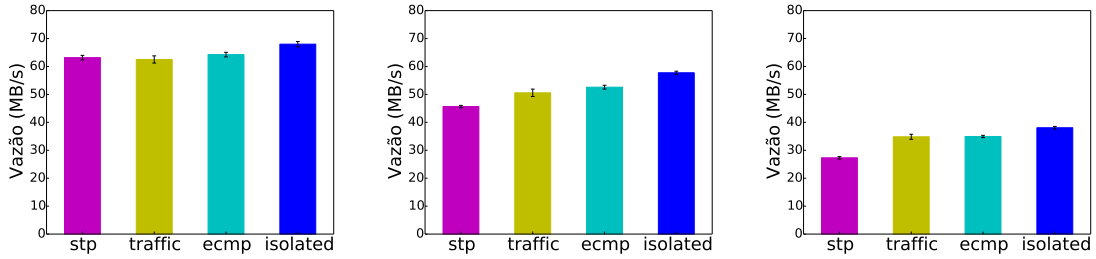
5.3. Resultados

A Figura 3 apresenta, para cada grupo de nós, a vazão média e os respectivos intervalos de confiança em função do algoritmo escolhido[3]. Podemos notar que há uma clara influência do processo de seleção de caminhos no desempenho da comunicação MPI, para todos os cenários avaliados. Essa influência é menos acentuada na sessão de testes com 4 nós (Figura 3(a)), com os algoritmos "stp" e "traffic" apresentando, inclusive, taxas equivalentes de vazão (63,17 MB/s e 62,51 MB/s, respectivamente). Os algoritmos "ecmp", com 64,25 MB/s, e "isolated", com 67,99 MB/s, alcançaram melhores resultados nessa sessão. O ganho do melhor algoritmo em relação ao pior foi de cerca de 9%, o que sugere uma correlação direta entre o número de mensagens na rede e os possíveis benefícios do uso de estratégias de roteamento distintas.

De fato, os gráficos das sessões de testes com 8 e 16 nós mostram mais claramente a relevância da definição de um método de roteamento apropriado na melhoria do desempenho da rede. Como indica a Figura 3(b), na sessão de testes com 8 nós, o ganho relativo do "isolated" (vazão de 57,75 MB/s) em comparação com o "stp" (vazão de 45,65 MB/s) foi superior a 26%. Ainda, os algoritmos "traffic", com 50,57 MB/s, e "ecmp", com 52,59 MB/s, também apresentaram melhores taxas de vazão em relação ao "stp". A sessão de testes com 16 nós (Figura 3(c)), embora apresente menores médias, revelou diferenças expressivas de taxas de vazão entre os algoritmos "stp" (27,31 MB/s) e "isolated" (38,07 MB/s), o que representa uma melhoria de aproximadamente 39%. Nesse mesmo cenário, os algoritmos "traffic" e "ecmp" também proporcionaram melhores desempenhos, com taxas de vazão de 34,83 MB/s e 34,94 MB/s, respectivamente.

Por fim, avaliamos a latência fim-a-fim dos pacotes em todas as sessões de testes. A Figura 4 mostra, para cada conjunto de nós, as respectivas funções de distribuição de probabilidade acumulada em relação aos algoritmos utilizados. Os gráficos demonstram

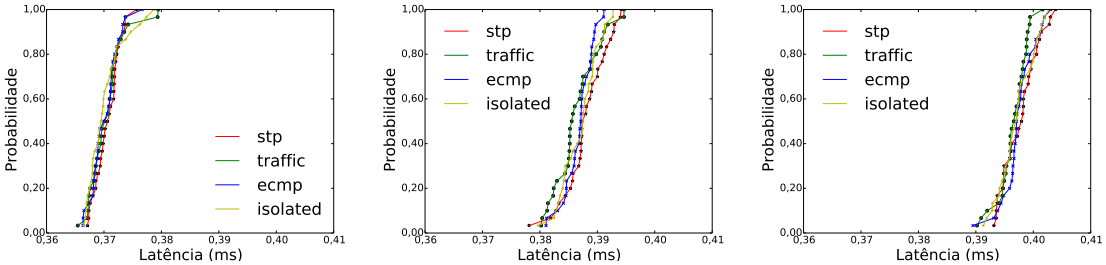
[3]Os dados resultantes das simulações e os códigos-fontes dos protótipos dos componentes de *software* desenvolvidos para a avaliação estão disponíveis em https://github.com/netlabufjf/hpc-sdn



(a) 4 Nós (b) 8 Nós (c) 16 Nós

Figura 3. Médias das taxas de vazão na comunicação MPI.

que o desempenho da rede permaneceu estável no que se refere a esse parâmetro, mesmo diante do uso de métodos de roteamento distintos. Realmente, os valores médios apresentaram-se muito próximos entre si: 0,37 ms (4 nós); 0,39 ms (8 nós); e 0,40 ms (16 nós). Os resultados também evidenciam que não houve contenção de rede no cenário.



(a) 4 Nós (b) 8 Nós (c) 16 Nós

Figura 4. Distribuição de latência dos pacotes.

Em suma, o algoritmo de roteamento "isolated", que utiliza a estratégia de isolamento de caminhos, apresentou o melhor desempenho em todos os cenários.

6. Conclusões e Trabalhos Futuros

Neste artigo, apresentamos a concepção de uma plataforma de comunicação baseada em SDN cujo propósito é suprir, de forma integrada, os requisitos de aplicações típicas de ambientes HPC e *Big Data*. Evidenciamos a importância da definição das configurações de rede mais adequadas aos perfis de tráfego e, por isso, apresentamos quatro estratégias de roteamento. Nós simulamos a arquitetura proposta e a avaliamos a partir de um *benchmark* bem conhecido. Os resultados mostram que a estratégia de isolamento de caminhos de rede apresenta taxas de vazão superiores em até 39%, dependendo da configuração de rede selecionada. Nós concluímos, portanto, que a exploração das capacidades de SDN pode maximizar a comunicação em uma infraestrutura de rede convergente, tornando-a mais eficiente.

Como trabalhos futuros, nós pretendemos replicar a análise deste estudo no contexto das aplicações *Big Data*, inclusive ratificando os resultados aqui encontrados em cenários de avaliação maiores. Mais adiante, nós esperamos desenvolver os mecanismos de classificação de tráfego de rede para ambas as aplicações. Planeja-se, ainda,

avaliar a utilização de ferramentas de "fatiamento" de rede como solução alternativa à implementação da plataforma convergente.

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 International Symposium on High Performance Parallel and Distributed Computing (HPDC)*.

Date, S., Abe, H., Khureltulga, D., Takahashi, K., Kido, Y., Watashiba, Y., U-chupala, P., Ichikawa, K., Yamanaka, H., Kawai, E., and Shimojo, S. (2016). SDN-accelerated HPC infrastructure for scientific research. *International Journal of Information Technology*, 22(1).

Economist (2010). The data deluge. *Special Supplement*.

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

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.

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*.

Qazi, Z., Lee, J., Jin, T., Bellala, G., Arndt, M., and Noubir, G. (2013). Application-awareness in SDN. *ACM SIGCOMM Computer Communication Review*, 43(4):487–488.

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*.

Thaler, D. and Hopps, C. (2000). Multipath issues in unicast and multicast next-hop selection. RFC 2991, RFC Editor. `http://www.rfc-editor.org/rfc/rfc2991.txt`. Accessed: July, 2018.

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

Zhang, H., Guo, X., Yan, J., Liu, B., and Shuai, Q. (2014). SDN-based ECMP algorithm for data center networks. In *Proc. of the IEEE ComComAp*.