Reduzindo Custos de Implantação e Execução de Clusters Spark em Nuvens Públicas

Alan L. Nunes[1], **Lúcia Maria de Assumpção Drummond**[1], **Cristina Boeres**[1]

[1] Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

alan_lira@id.uff.br, {lucia, boeres}@ic.uff.br

Abstract. *Public cloud providers offer a wide variety of services and computational resources. The use of more specialized and automatically managed services by providers, such as the Platform as a Service (PaaS) model, is one of the causes for the increase in monetary costs charged to users. In this work we present a tool for deploying and running Spark clusters that uses the Infrastructure as a Service (IaaS) model. The results obtained from several use cases indicate that the proposed tool, compared to PaaS, is able to reasonably reduce the costs of running Spark applications in the cloud.*

Resumo. *Provedores de nuvens públicas oferecem uma grande variedade de serviços e recursos computacionais. A utilização de serviços mais especializados e automaticamente gerenciados pelos provedores, tal como o modelo de Plataforma como Serviço (PaaS), é uma das causas para o aumento de custos monetários cobrados aos usuários. Neste trabalho apresentamos uma ferramenta de implantação e execução de clusters Spark que utiliza o modelo de Infraestrutura como Serviço (IaaS). Os resultados obtidos a partir de diversos casos de uso apontam que a ferramenta proposta, comparada ao PaaS, é capaz de reduzir razoavelmente os custos de execução de aplicações Spark na nuvem.*

1. Introdução

Cloud Computing é um modelo de acesso a recursos de computação e armazenamento sob demanda através da internet, que costuma fornecer uma capacidade maior do que a disponível localmente pelos clientes. Acessar a nuvem pode ser mais barato, rápido e conveniente do que adquirir e operar sistemas próprios [Foster and Gannon 2017].

As nuvens públicas compõem o modo de implantação mais difundido deste modelo, dentre as quais Amazon Web Services (AWS), Microsoft Azure e Google Cloud destacam-se como os provedores de serviços mais utilizados. Além das vantagens de provisionamento rápido de recursos e de redução significativa do custo operacional em comparação com infraestruturas dedicadas, os provedores também oferecem descontos notáveis para a locação de instâncias preemptivas de máquinas virtuais (VMs) em recursos computacionais ociosos. Por exemplo, a Amazon EC2 oferta instâncias spot que podem custar até 90% menos do que as instâncias sob demanda. No entanto, os recursos alocados para instâncias preemptivas podem ser reivindicados a qualquer instante pelos provedores, tornando imprescindível aos clientes o uso de técnicas de tolerância a falhas.

Nas últimas décadas, o volume de dados produzido atingiu um nível sem precedentes na academia e na indústria. A necessidade de processamento e consulta de grandes volumes de dados em tempo hábil motivou o desenvolvimento de soluções eficientes

para a análise de dados, conhecidas como frameworks de Big Data, *e.g.*, Apache Hadoop e Apache Spark [Zaharia et al. 2016]. Uma característica marcante dessas soluções é a capacidade de escalar a execução de uma aplicação para uma grande quantidade de nós computacionais interligados. Por essa razão, a nuvem é uma excelente opção para a implantação de um cluster Spark [Yan et al. 2016].

A solução em nuvens públicas que atualmente promove a utilização do Spark é conhecida como *On-demand Managed Big Data Cluster*, um modelo de Plataforma como Serviço (PaaS) que segue a política de precificação pay-as-you-go. As opções mais populares desta modalidade são Azure HDInsight, Amazon Elastic MapReduce (EMR) e Google Dataproc. Apesar deste serviço oferecer facilidade de gerenciamento de clusters para o processamento de grandes volumes de dados, os clientes são obrigados a pagar taxas adicionais de uso além do custo básico dos recursos computacionais alocados.

Neste trabalho apresentamos uma ferramenta open-source de implantação e execução de clusters Spark que explora o modelo de Infraestrutura como Serviço (IaaS) em nuvens públicas. Sua utilização permite eliminar os custos adicionais do modelo PaaS e reduzir os esforços necessários para o gerenciamento manual de recursos computacionais do modelo IaaS. Adotamos a AWS durante a avaliação experimental da nossa ferramenta, pois é a plataforma de nuvem mais abrangente em recursos e serviços. Os resultados indicaram reduções razoáveis do custo monetário de execução de aplicações.

2. Modos Típicos de Implantação de Clusters Spark na AWS

Os serviços de nuvem da AWS mais utilizados para a implantação de clusters Spark são Elastic MapReduce (EMR) e Elastic Compute Cloud (EC2).

Elastic MapReduce (EMR) é uma solução de PaaS que simplifica a criação e a operação de ambientes e aplicativos de Big Data através de provisionamento, dimensionamento e reconfiguração da infraestrutura de cluster[1]. No EMR é possível construir automaticamente clusters Spark utilizando instâncias do EC2, de modo que os clientes selecionam os tipos de instâncias que serão provisionados com base nos requisitos de seu aplicativo. Portanto, o EMR é adequado para usuários iniciantes em nuvem devido ao seu gerenciamento automático e à disponibilização de ambientes pré-configurados. No entanto, há um custo associado ao uso do serviço EMR. Em particular, para cada instância do EC2, os usuários são cobrados adicionalmente por segundo a partir do momento em que o cluster é iniciado até que seja encerrado, com um mínimo de um minuto de cobrança. Por exemplo, ao considerarmos todos os tipos de instância do EC2 com suporte ao EMR na região leste dos Estados Unidos (Virgínia do Norte), o custo adicional médio de utilização desse serviço é de 19,8%[2].

Elastic Compute Cloud (EC2) é uma solução de IaaS que oferece uma ampla plataforma de computação com mais de 500 tipos de instâncias de máquinas virtuais[3]. As instâncias são otimizadas e adequadas para diferentes casos de uso, a fim de que os clientes possam executar aplicativos e cargas de trabalho com computação, memória, armazenamento e equilíbrio de rede ideais. Este modelo oferece uma melhor customização de

[1] `https://aws.amazon.com/emr/features/`
[2] `https://aws.amazon.com/emr/pricing/`
[3] `https://aws.amazon.com/ec2/`

clusters Spark e os usuários são cobrados apenas pelo custo básico dos recursos computacionais alocados. Contudo, o gerenciamento manual desses recursos não é uma atividade trivial aos usuários de nuvens, principalmente se forem inexperientes.

3. Sparking Cloud

Sparking Cloud[4] é uma Command-Line Interface (CLI) open-source desenvolvida em Python que facilita a implantação de clusters Spark em nuvens públicas por meio de um computador local conectado à internet, conforme ilustrado na **Figura 1**. As ações de construção e destruição de clusters são realizadas via Software Development Kits (SDK) disponibilizadas pelos provedores. As ações de configuração do ambiente, definição das aplicações a serem executadas, submissão de jobs e coleta da saída das execuções são efetuadas através de canais seguros de comunicação via Secure Shell (SSH).

A versão atual do Sparking Cloud (v0.2.1) suporta o gerenciamento de múltiplos clusters Spark na AWS, configurações personalizadas para nós Masters e Workers, leitura e escrita de dados via armazenamento local (instance store) ou Simple Storage Service (S3) e utilização de VMs preemptivas (spot) ou não-preemptivas (on-demand).

Uma desvantagem do Sparking Cloud em relação ao EMR, por exemplo, é a ausência de uma interface gráfica interativa de gerenciamento de clusters que auxilie usuários não familiarizados com execuções via linha de comando. Por outro lado, uma potencial vantagem do Sparking Cloud é a criação de clusters em diferentes provedores de nuvens públicas, evitando um eventual aprisionamento tecnológico (vendor lock-in).

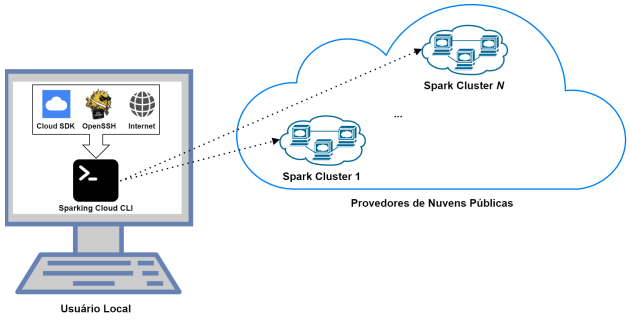

Figura 1. Representação simplificada da Sparking Cloud CLI.

4. Resultados Experimentais

Empregamos o Sparking Cloud em pesquisas recentes para a implantação de variadas configurações de clusters Spark, considerando a nuvem pública da Amazon (AWS). A sua utilização facilitou a execução de centenas de experimentos e permitiu a verificação do contraste de custos monetários das execuções, quando comparados ao serviço EMR.

No primeiro caso de uso [Nunes et al. 2021], definimos 19 configurações de clusters, conforme indicado na **Tabela 1**. Vale notar que, as instâncias utilizadas no primeiro cluster não são disponibilizadas pela Amazon para o serviço EMR, o que impossibilitou a comparação de custos neste cenário. Além disso, os últimos seis clusters contavam inicialmente com oito Workers spot, mas que ao longo das execuções sofreram, em

[4] https://github.com/alan-lira/sparking-cloud

instantes t pré-determinados, $t \in \{30, 60, 120\}$ minutos, revogações simuladas de dois Workers, cada. Os eventos de revogação permitiram-nos verificar a tolerância a falhas do framework Spark e investigar, em cada cenário, o impacto da utilização de instâncias preemptivas em uma aplicação Spark, em termos de tempo e custo monetário de execução.

Tabela 1. Configurações de clusters Spark utilizados em [Nunes et al. 2021].

Configuração do Cluster	Master Nodes					Worker Nodes				
	Tipo de Instância	Mercado	Custo (USD por hora)		Quant.	Tipo de Instância	Mercado	Custo (USD por hora)		Quant.
			EC2	EMR				EC2	EMR	
Cluster #1	t2.medium	on-demand	0,0464	N/A	1	t2.medium	spot	0,0139	N/A	8
Cluster #2	r5.xlarge	on-demand	0,252	0,063	1	r5.xlarge	on-demand	0,252	0,063	8
Cluster #3	r5.xlarge	on-demand	0,252	0,063	1	r5.xlarge	spot	0,1374	0,063	8
Cluster #4	r5dn.xlarge	on-demand	0,334	0,084	1	r5dn.xlarge	on-demand	0,334	0,084	8
Cluster #5	r5dn.xlarge	on-demand	0,334	0,084	1	r5dn.xlarge	spot	0,1232	0,084	8
Cluster #6	z1d.xlarge	on-demand	0,372	0,093	1	z1d.xlarge	on-demand	0,372	0,093	8
Cluster #7	z1d.xlarge	on-demand	0,372	0,093	1	z1d.xlarge	spot	0,1116	0,093	8
Cluster #8	d3.xlarge	on-demand	0,499	0,1247	1	d3.xlarge	on-demand	0,499	0,1247	8
Cluster #9	d3.xlarge	on-demand	0,499	0,1247	1	d3.xlarge	spot	0,1497	0,1247	8
Cluster #10	h1.2xlarge	on-demand	0,468	0,117	1	h1.2xlarge	on-demand	0,468	0,117	8
Cluster #11	h1.2xlarge	on-demand	0,468	0,117	1	h1.2xlarge	spot	0,1404	0,117	8
Cluster #12	i3en.xlarge	on-demand	0,452	0,113	1	i3en.xlarge	on-demand	0,452	0,113	8
Cluster #13	i3en.xlarge	on-demand	0,452	0,113	1	i3en.xlarge	spot	0,1356	0,113	8
Cluster #14	z1d.xlarge	on-demand	0,372	0,093	1	z1d.xlarge	spot	0,1116	0,093	8*
Cluster #15	i3en.xlarge	on-demand	0,452	0,113	1	i3en.xlarge	spot	0,1356	0,113	8*
Cluster #16	z1d.xlarge	on-demand	0,372	0,093	1	z1d.xlarge	spot	0,1116	0,093	8*
Cluster #17	i3en.xlarge	on-demand	0,452	0,113	1	i3en.xlarge	spot	0,1356	0,113	8*
Cluster #18	z1d.xlarge	on-demand	0,372	0,093	1	z1d.xlarge	spot	0,1116	0,093	8*
Cluster #19	i3en.xlarge	on-demand	0,452	0,113	1	i3en.xlarge	spot	0,1356	0,113	8*

A **Tabela 2** sumariza o tempo total de execução de aplicações Spark em cada cluster formado em [Nunes et al. 2021]. A partir dessas métricas, e considerando os custos dos serviços EC2 e EMR e as dimensões de cada cluster (indicados na Tabela 1), calculamos os respectivos custos monetários totais de execução em dólares americanos (USD). Observe que a utilização da nossa ferramenta durante o primeiro caso de uso, quando comparada ao Amazon EMR, permitiu uma redução de custos monetários mínima de 19,98%, máxima de 39,83% e média de 32,23% com desvio-padrão de 8,94.

Tabela 2. Custos monetários de execução em [Nunes et al. 2021].

Configuração do Cluster	Tempo Total de Execução (Horas)	Custo Monetário de Execução (USD)		Variação Percentual (%)
		Amazon EMR	Sparking Cloud	
Cluster #1	20,93	N/A	3,30	N/A
Cluster #2	8,73	24,75	19,80	−20,0%
Cluster #3	8,73	16,75	11,80	−29,55%
Cluster #4	8,72	32,80	26,21	−20,09%
Cluster #5	8,72	18,09	11,50	−36,43%
Cluster #6	6,76	28,28	22,62	−20,01%
Cluster #7	6,76	14,21	8,55	−39,83%
Cluster #8	8,26	46,35	37,08	−20,0%
Cluster #9	8,26	23,28	14,01	−39,82%
Cluster #10	10,08	53,09	42,48	−19,98%
Cluster #11	10,08	26,66	16,05	−39,80%
Cluster #12	8,45	42,98	34,39	−19,99%
Cluster #13	8,45	21,58	12,99	−39,81%
Cluster #14	7,04	12,53	7,67	−38,79%
Cluster #15	9,19	16,16	9,90	−38,74%
Cluster #16	6,96	13,02	7,92	−39,17%
Cluster #17	8,87	16,25	9,91	−39,02%
Cluster #18	6,65	13,70	8,26	−39,71%
Cluster #19	8,80	17,34	10,50	−39,45%

No segundo caso de uso [Campbell et al. 2022], definimos 17 configurações de clusters Spark apenas com VMs do tipo r5.large no mercado spot. No entanto, esse tipo de

instância também não é disponibilizado pela Amazon para o serviço EMR, o que impediu de realizarmos uma análise comparativa de custos monetários de execução.

Finalmente, no terceiro caso de uso [Nunes et al. 2023], definimos 235 configurações de clusters Spark com VMs de diversos tipos nos mercados on-demand e spot. Deste total, 30 configurações não foram utilizadas durante a análise comparativa, pois empregavam tipos de VMs não suportadas pelo EMR. Contudo, dos cenários confrontáveis, percebemos uma redução de custos monetários mínima de 5,99%, máxima de 23,06% e média de 15,56% com desvio-padrão de 4,22.

5. Conclusão e Trabalhos Futuros

Este trabalho apresentou uma ferramenta de implantação de clusters Spark em nuvens públicas que utiliza o modelo IaaS. Obtivemos, com o emprego da ferramenta proposta, reduções de até 39,83% dos custos monetários de execução de aplicações Spark na Amazon EC2, quando comparados com a plataforma Amazon EMR.

Para trabalhos futuros, planejamos expandir o seu uso para a criação de clusters em múltiplos provedores. Quanto à leitura e escrita de dados, planejamos dar suporte ao sistema de arquivos HDFS (em andamento). Por fim, planejamos incluir um monitoramento de clusters com redimensionamentos dinâmicos baseados na taxa de utilização de recursos e na ocorrência de revogações de recursos preemptivos (em andamento).

Referências

Campbell, R., Nunes, A. L., Boeres, C., and Drummond, L. M. A. (2022). MapReduce na AWS: Uma Análise de Custos Computacionais Utilizando os Serviços FaaS e IaaS. In *Anais do XXIII Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 2022)*, pages 145–156, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/wscad.2022.226308.

Foster, I. and Gannon, D. B. (2017). *Cloud Computing for Science and Engineering*. MIT Press, Cambridge, MA, USA.

Nunes, A. L., Melo, A., Boeres, C., de Oliveira, D., and Drummond, L. M. A. (2021). Towards Analyzing Computational Costs of Spark for SARS-CoV-2 Sequences Comparisons on a Commercial Cloud. In *Anais do XXII Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 2021)*, pages 192–203, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/wscad.2021.18523.

Nunes, A. L., Melo, A., Tadonki, C., Boeres, C., de Oliveira, D., and Drummond, L. M. A. (2023). Optimizing computational costs of Spark for SARS-CoV-2 sequences comparisons on a commercial cloud. *Concurrency and Computation: Practice and Experience*, page e7678. DOI: 10.1002/cpe.7678.

Yan, Y., Gao, Y., Chen, Y., Guo, Z., Chen, B., and Moscibroda, T. (2016). TR-Spark: Transient Computing for Big Data Analytics. *17th ACM SoCC*, pages 484–496. DOI: 10.1145/2987550.2987576.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., et al. (2016). Apache Spark: A Unified Engine for Big Data Processing. *Communications of the ACM*, 59(11):56–65. DOI: 10.1145/2934664.