Análise de Precificação de Recursos Utilizados em Computação em Nuvem

Gustavo J. Portella[1], Genaína N. Rodrigues[1], Alba C. M. de Melo[1]

[1]Departamento de Ciência da Computação (CIC)
Universidade de Brasília (UnB) – Brasília, DF – Brazil

gustavo.portella@aluno.unb.br, {genaina,albamm}@cic.unb.br

Abstract. *Cloud providers usually offer a wide variety of resources to their users. In this scenario, selecting an inappropriate resource can lead to financial losses and/or long response times. Therefore, the use of an effective strategy for selecting cloud computing resources can bring important benefits to users. In this paper, we present experiments in order to determine the influence of the characteristics of processor and memory in the composition of the cost of different types of resources available on the Amazon EC2 and Google GCE providers. Such analysis can be incorporated in strategies which aim to decrease the financial cost and to attain a good performance for the cloud applications.*

Resumo. *Os provedores de nuvem geralmente oferecem uma grande variedade de recursos a seus usuários. Nesse cenário, a seleção de um recurso inapropriado pode levar a perdas financeiras e/ou tempos de resposta longos. Sendo assim, a utilização de uma estratégia eficiente de seleção de recursos em nuvem pode trazer importantes benefícios para os seus usuários. Neste artigo, são apresentados experimentos com o objetivo de se determinar a influência de características do processador e da memória na composição do custo de diferentes recursos disponíveis nos provedores Amazon EC2 e Google GCE. Tal análise pode ser incorporada em estratégias que visam ao mesmo tempo a diminuição do custo financeiro e a manutenção de bom desempenho para as aplicações em nuvem.*

1. Introdução

O surgimento da computação em nuvem proporcionou uma verdadeira mudança quanto ao uso de infraestrutura de recursos computacionais. Nesse novo modelo, um usuário tem a possibilidade de escolher os recursos computacionais de acordo com os requisitos de execução de sua aplicação e pagar pelo tempo de uso [Foster et al. 2008]. Essa possibilidade faz com que a escolha correta dos recursos mereça uma atenção especial, ainda mais pelo fato de que os custos decorrentes da utilização são aferidos não somente pelo tipo, mas também em função do tempo de utilização dos recursos.

Nesse contexto, os provedores de computação em nuvem oferecem serviços com modelos de precificação de recursos flexíveis e de acordo com as necessidades dos usuários. Um exemplo disso é o caso da *Amazon*, que oferece três diferentes modelos de precificação para comercialização de seu serviço *EC2 – Elastic Compute Cloud* [Amazon 2016], a depender do tipo de instância ou máquina virtual utilizada. A empresa Google também oferece um modelo flexível por meio do serviço *GCE* –

Google Compute Engine [Google 2016], dependendo dos recursos de processamento, memória, armazenamento e largura de banda escolhidos.

Este trabalho apresenta uma análise quantitativa de modelos de precificação utilizados por provedores de computação em nuvem no modelo *IaaS – Infrastructure as a Service* [Mell and Grance 2011], com o objetivo de estabelecer qual a influência dos parâmetros de infraestrutura, principalmente características de processamento e memória, na composição do custo financeiro. Na próxima seção será feita uma contextualização sobre nuvens no modelo *IaaS*. Na seção seguinte serão detalhados o escopo da pesquisa e os experimentos realizados. Em seguida, serão indicados trabalhos relacionados e, finalmente, na conclusão do artigo serão apresentadas as principais observações e conclusões realizadas, assim como os trabalhos futuros.

2. Computação em Nuvem e o Modelo *IaaS*

No modelo *IaaS*, o provedor de nuvem oferece aos usuários um conjunto de recursos virtualizados. Os usuários são responsáveis por solicitar recursos virtuais (e.g máquinas virtuais) e gerenciá-los enquanto o provedor é responsável por (a) selecionar um recurso físico e instanciar a máquina virtual nele; (b) monitorar o recurso físico de maneira a garantir a disponibilidade; e (c) cobrar pelo uso.

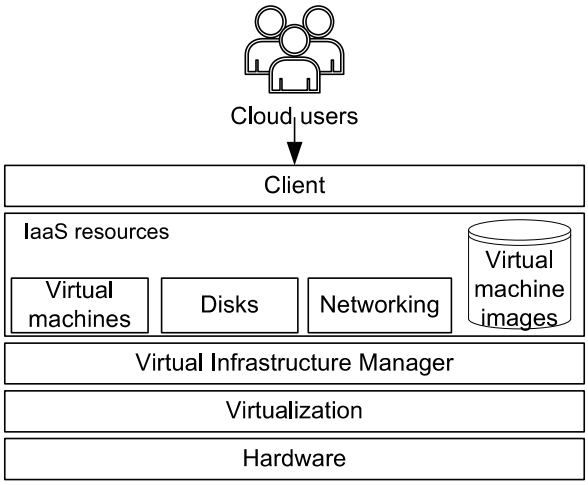


Figura 1. Arquitetura *IaaS* [Leite 2014]

A Figura 1 apresenta uma arquitetura *IaaS* genérica. Como pode ser visto, acima no nível de hardware e virtualização, existe um gerente de infraestrutura virtual que cria recursos virtuais. Os serviços são organizados no *service layer* e incluem máquinas virtuais, *storage,* rede e imagens de máquinas virtuais.

Os serviços de computação são oferecidos como instâncias de máquinas virtuais. Uma máquina virtual pertence a um tipo de instância que determina características como número de *cores* (i.e. *vCPUs*), quantidade de memória RAM, capacidade de disco e capacidade de rede.

As instâncias são divididas em famílias de acordo com o seu propósito. No momento da escrita do presente artigo, o provedor Amazon EC2 oferece 40 tipos de instâncias dispostas em 5 famílias: *general purpose, compute optimized, memory optimized, GPU e storage optimized* [Amazon 2016] e o Google GCE oferece 18 tipos

de instâncias dispostas em quatro famílias: *small*, *standard*, *high CPU* e *high memory* [Google 2016].

Para utilizar o serviço de nuvem *IaaS*, o usuário deve [Sotomayor 2009]: (1) escolher uma região; (2) selecionar um tipo de instância; (3) selecionar uma imagem de máquina virtual; (4) selecionar um tipo de disco; (5) solicitar que o provedor crie a VM; (6) configurar a VM; (7) executar a aplicação; e (8) transferir os dados de saída da nuvem para a máquina local.

Uma região (1) é um ambiente de nuvem independente composta por um ou mais datacenters em uma dada região geográfica. Os grandes provedores de nuvem, como Amazon e Google, possuem diversas regiões ao redor do mundo. Por exemplo, atualmente a Amazon possui 11 regiões, sendo uma delas em São Paulo (sa-east-1). Após escolher a região, o usuário deve selecionar o tipo de instância (2), considerando requisitos como número mínimo de *cores*, quantidade mínima de memória, preço máximo, etc. Geralmente, existem muitas instâncias que atendem os requisitos e os usuários geralmente selecionam uma delas, de maneira *ad-hoc,* o que pode levar a alto custo financeiro. Depois da escolha da instância, o usuário deve escolher uma imagem de máquina virtual (VMI) (3), que depende basicamente do sistema operacional suportado pela aplicação. Após isso, o usuário seleciona o tipo de disco (4), caso deseje usá-lo. Tendo a região, o tipo da instância, a VMI e o tipo de disco, o usuário solicita que o provedor de nuvem crie a VM (5). Para tanto, o provedor de nuvem seleciona uma máquina física para abrigar a VM e copia a VMI para essa máquina, solicitando que o *layer* de virtualização inicialize a VM, atribuindo um IP à mesma e iniciando a cobrança por seu uso. Com a VM iniciada, os usuários podem configurá-la (6), instalando sua aplicação e movendo dados de entrada para a mesma. No passo (7), a aplicação é executada. Ao final da execução (8), o usuário transfere os dados de saída para a sua máquina local e libera a VM. Nesse ponto, a cobrança é interrompida.

Na próxima seção, será analisada a relação entre custo, número de *cores* de processamento e quantidade de memória dos diversos tipos de instância oferecidos por alguns provedores de nuvem atuais, de maneira a gerar informações que ajudem tanto os usuários na seleção do tipo da instância (passo 2) como aos provedores de nuvem no estabelecimento do custo de novos tipos de instância.

3. Análise de Dados dos Provedores Amazon EC2 e Google GCE

3.1. Hipótese da Função de Custo

O objetivo principal deste trabalho é identificar a relação entre o custo de uma instância e a quantidade de núcleos do processador e o tamanho da memória. Nossa primeira hipótese é de que essa relação é linear e, portanto, pode ser expressa conforme a equação (1).

$$Custo = \alpha + \beta * Processador + \gamma * Memória \quad (1)$$

Onde: *Custo* = custo em valores monetários (em $/hora);
Processador = quantidade de núcleos de processamento (escalar);
Memória = tamanho da memória (em GB);
α = coeficiente linear da equação;
β = coeficiente angular do fator Processador;
γ = coeficiente angular do fator Memória.

Do ponto de vista do provedor de nuvem, o estabelecimento da relação na Equação 1 busca contribuir para o aumento da eficiência na utilização dos recursos de infraestrutura, assim como para a evolução dos modelos econômicos e de precificação de recursos. Na perspectiva do usuário, o modelo pode ser utilizado em estratégias eficientes de seleção de tipos de recursos (i.e instâncias) e de modelos de precificação disponíveis, a depender das características de suas aplicações.

Como segunda contribuição, o artigo tem como objetivo identificar quais outros fatores podem influenciar nesta relação e se diferentes provedores praticam estratégias semelhantes de precificação para seus recursos. As técnicas de análise quantitativa [Jain 1991] aplicadas são regressão linear, regressão multilinear, análise de variância com intervalo de confiança e clusterização.

3.2. Coleta dos Dados

Os dados coletados se referem a medições de custos de utilização de instâncias de computação em nuvem, sendo que cada tipo de instância possui uma configuração específica de quantidade de núcleos de processamento e memória. Foram coletados dados dos provedores Amazon EC2 e Google GCE, que são originários de 3 fontes distintas. A origem nº 1 contém dados utilizados na pesquisa referenciada em [Leite 2015], coletados em 2014. A origem nº 2 contém dados de [Ostermann et al. 2009]. A origem nº 3 contém dados recentes da Amazon EC2 [Amazon 2016], coletados em junho de 2016.

3.3. Experimentos Executados

Os experimentos foram planejados de acordo com a origem dos dados, técnica de análise aplicada e provedor de computação em nuvem considerado. Em todos os experimentos foram considerados os custos no modelo de precificação pós-pago ou *on-demand*, para instâncias disponíveis na costa leste dos Estados Unidos (*US East*), utilizando o sistema operacional Unix/Linux. Na Tabela 1 é apresentado um subconjunto dos experimentos realizados. Nesses experimentos foram aplicadas as técnicas de regressão linear e multilinear, além da análise de variância com grau de confiança de 90%, tendo como objetivo a avaliação da relevância estatística do coeficiente linear α e dos coeficientes angulares β (núcleos de processamento) e γ (quantidade de memória). A técnica de clusterização foi utilizada com a finalidade de compor grupos representativos de dados. No caso, foi utilizado o algoritmo *Minimal Spanning Tree*, considerando a distância euclidiana mínima entre os pontos que farão parte de um grupo [Jain 1991].

Tabela 1. Experimentos Executados.

Id	Técnica Aplicada	Descrição do Experimento	Origem dos Dados
mlr	Regressão Multilinear (MLR)	Custo (USD / hora) em função de quantidade de núcleos de CPU e Memória (GB).	Amazon e Google [Leite 2015]
c-spt	Clusterização com filtro *spanning tree*	Custo (USD / hora) em função de quantidade de núcleos de CPU e Memória (GB), filtrando-se através da aplicação do algoritmo *Minimal Spanning Tree*.	Amazon [Leite 2015]

mlr-spt	Regressão Multilinear (MLR) com grupamento *spanning tree*	Custo (USD / hora) em função de quantidade de núcleos de CPU e Memória (GB), agrupando-se as instâncias com o algoritmo *Minimal Spanning Tree*.	Amazon [Leite 2015]
rl-linpack	Regressão Linear (LR)	Custo (USD / hora) em função de quantidade do benchmark LINPACK (GFLOPS).	Amazon [Ostermann 2009]
mlr-ecu	Regressão Multilinear usando também ECU	Custo (USD / hora) em função de quantidade de núcleos de CPU, Memória (GB) e ECU.	Amazon [Amazon 2016]
mlr-ecu-restricted	Regressão Multilinear	Custo (USD / hora) em função de quantidade de núcleos de CPU, Memória (GB) e ECU, somente para instâncias *compute-optimized* e *memory-optimized*.	Amazon [Amazon 2016]
mlr-restricted-no-core	Regressão Multilinear	Custo (USD / hora) em função de quantidade de Memória (GB) e ECU, somente para instâncias *compute-optimized* e *memory-optimized*.	Amazon [Amazon 2016]

3.4. Resultados Experimentais

Os resultados dessa seção são mostrados no formato de quadro-resumo, que fornece a identificação, descrição e observações sobre o experimento. São mostrados também a equação linear, o coeficiente de determinação e os dados relacionados à relevância estatística obtidos por meio da análise de variância e aplicação do *F-Test*. Para análise visual, são mostrados 2 tipos de gráfico: Quantis Normal X Residual (erro) e Homocedasticidade. O primeiro gráfico deve se assemelhar a uma reta, de forma a demonstrar que existe uma relação linear entre os parâmetros. O segundo gráfico evidencia a dispersão dos pontos da regressão dentro de bons limites.

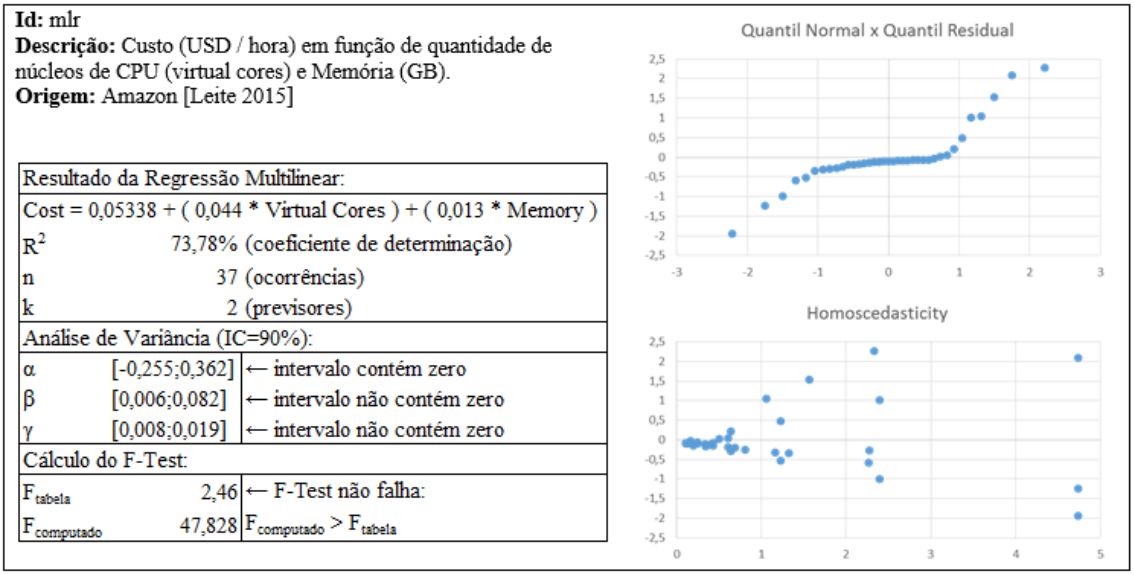


Figura 2. Quadro-resumo do experimento mlr com dados da Amazon EC2 coletados em [Leite 2015].

A Figura 2 mostra o quadro-resumo com os resultados da regressão multilinear **mlr**, que define Custo (USD / hora) em função de quantidade de núcleos de CPU (*virtual cores*) e Memória (GB), para tipos de instâncias da Amazon EC2 coletados em [Leite 2015]. Nesse experimento, o coeficiente de determinação (R^2) é igual a 73,78%. O coeficiente linear α não é estatisticamente relevante na análise de variância, pois contém valor zero no intervalo, ou seja, não se pode afirmar com 90% de confiança que

o coeficiente representa o comportamento linear esperado. Outro problema apresentado é o gráfico de homocedasticidade, que apresenta uma tendência de aglomeração no formato de cone horizontal, sugerindo a necessidade de uma transformação dos dados (e. g. logarítmica). O gráfico dos quantis evidencia que a relação não é linear. Apesar de passar no *F-Test*, conclui-se que a regressão possui variáveis externas não consideradas. Dessa forma, o custo de um tipo de instância não deve ser calculado considerando apenas número de núcleos e quantidade de memória para o provedor Amazon EC2.

Id: mlr
Descrição: Custo (USD / hora) em função de quantidade de núcleos de CPU (virtual cores) e Memória (GB).
Origem: Google [Leite 2015]

Resultado da Regressão Multilinear:		
Cost = -0,006 + (0,04 * Virtual Cores) + (0,007 * Memory)		
R^2	99,50% (coeficiente de determinação)	
n	15 (ocorrências)	
k	2 (previsores)	
Análise de Variância (IC=90%):		
α	[-0,027;0,015]	← intervalo contém zero
β	[0,037;0,044]	← intervalo não contém zero
γ	[0,006;0,008]	← intervalo não contém zero
Cálculo do F-Test:		
$F_{computado}$	1195,249	← F-Test não falha:
F_{tabela}	2,81	$F_{computado} > F_{tabela}$

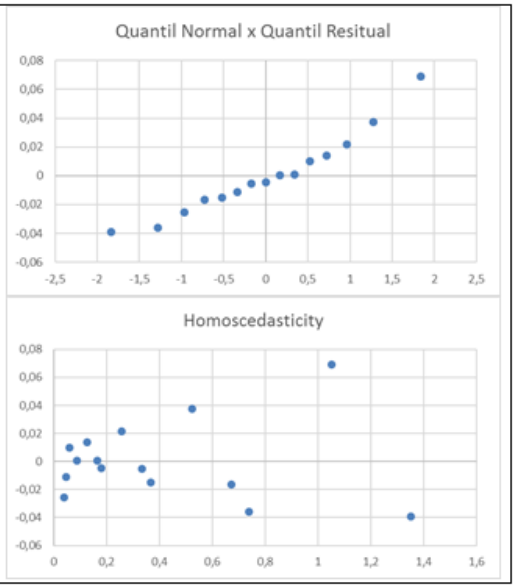


Figura 3. Quadro-resumo do experimento mlr com dados do Google GCE coletados em [Leite 2015].

Análise semelhante foi realizada com dados do Google, conforme mostrado na Figura 3. Os resultados obtidos para as instâncias Google GCE foram melhores, do ponto de vista estatístico, do que os obtidos para as instâncias Amazon EC2, principalmente em relação ao coeficiente de determinação. Entretanto, permanecem os problemas relacionados à relevância estatística do coeficiente linear α e ao gráfico de homocedasticidade. Sendo assim, não é possível afirmar que há uma relação de linearidade entre os fatores para se estimar custos de instâncias nesse provedor.

Assim, com o objetivo de estruturar melhor os fatores, foi realizada uma caracterização de carga com base nos tipos de instâncias da Amazon EC2. Para tanto, foi aplicado o algoritmo de clusterização *Minimal Spanning Tree* no conjunto de 37 tipos de instâncias. A Tabela 2 mostra os dados dos centroides obtidos para cada um dos 6 grupos representativos e a Figura 4 exibe o dendograma resultante dos agrupamentos.

Tabela 2. Grupos de Instâncias Amazon EC2 após aplicação do algoritmo de clusterização *Minimal Spanning Tree*.

Cluster	Instance Types	Virtual Cores	Memory (GB)	Cost / Hour
A	c3.8xlarge,cc2.8xlarge	32,00	60,25	1,84
B	i2.8xlarge,r3.8xlarge,cr1.8xlarge	32,00	244,00	4,99
C	hs1.8xlarge,r3.4xlarge,i2.4xlarge	16,00	119,50	3,50
D	hi1.4xlarge,m2.4xlarge,r3.2xlarge,i2.2xlarge	12,00	62,60	2,10

E	c3.2xlarge,g2.2xlarge,m2.xlarge,r3.large,m3.xlarge,m1.xlarge,m3.medium,m1.medium,c3.large,t2.medium,t2.micro,t1.micro,c1.medium,t2.small,m1.small,c1.xlarge,c3.xlarge,m3.large,m1.large	4,34	10,21	0,30
F	m3.2xlarge,m2.2xlarge,r3.xlarge,i2.xlarge,c3.4xlarge,cg1.4xlarge	11,00	28,71	1,01

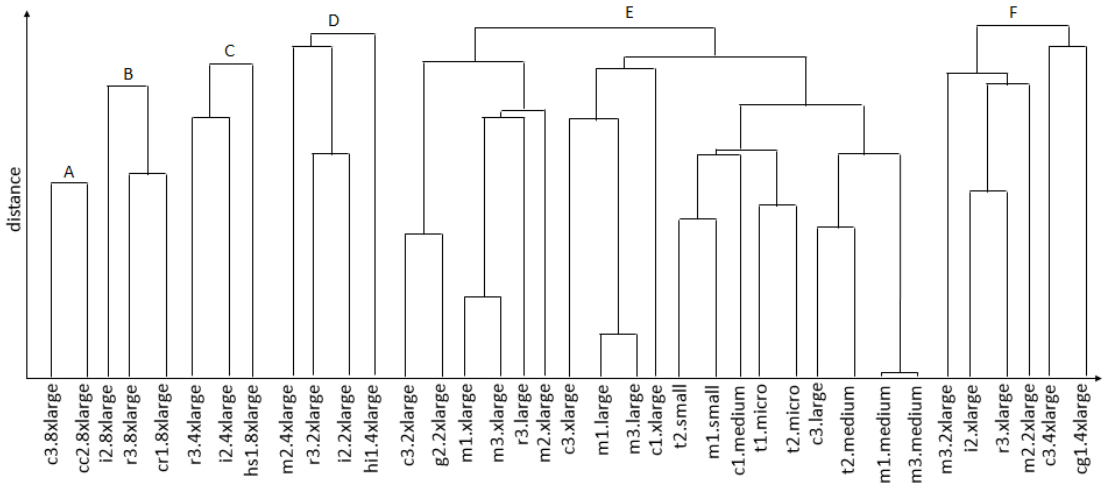


Figura 4. Dendograma de representação dos grupos em função da distância entre instâncias, após aplicação do algoritmo de clusterização *Minimal Spanning Tree*.

Com base nessa nova configuração, buscou-se então analisar uma possível regressão de linearidade entre os fatores. Os resultados são apresentados na Figura 5.

Id: mlr-spt
Descrição: Custo (USD / hora) em função de quantidade de núcleos de CPU e Memória (GB), agrupando-se instâncias após aplicação do algoritmo de clusterização *Minimal Spanning Tree*.
Origem: Amazon [Leite 2015]

Resultado da Regressão Multilinear:		
Cost = 0,5609 + (0,00245 * Virtual Cores) + (0,01924 * Memory)		
R^2	94,41% (coeficiente de determinação)	
n	6 (ocorrências)	
k	2 (previsores)	
Análise de Variância (IC=90%):		
α	[-0,425;1,547]	← intervalo contém zero
β	[-0,063;0,067]	← intervalo contém zero
γ	[0,01;0,028]	← intervalo não contém zero
Cálculo do F-Test:		
$F_{computado}$	25,338	← F-Test não falha:
F_{tabela}	9,16	$F_{computado} > F_{tabela}$

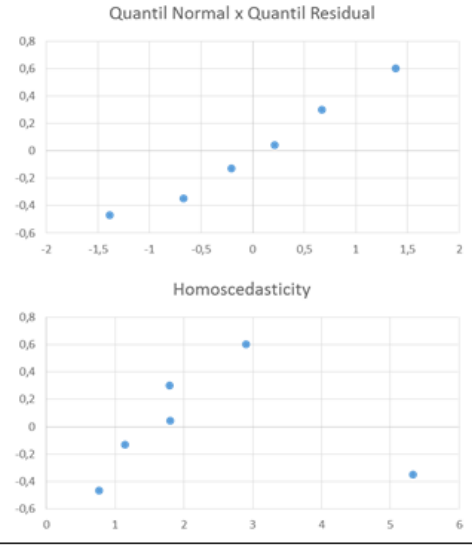


Figura 5. Quadro-resumo do experimento mlr-spt com dados da Amazon EC2 coletados em [Leite 2015], após aplicação do algoritmo de clusterização *Minimal Spanning Tree*.

Os resultados ainda não foram motivadores (Figura 5), uma vez que o coeficiente linear α e o coeficiente angular β não são estatisticamente relevantes devido à presença do valor zero em seus intervalos. Além disso, não há como determinar homocedasticidade dos dados, por serem poucos pontos no gráfico, apesar de haver uma melhora significativa do coeficiente de determinação em comparação com o experimento **mlr** sem uso de clusterização. Apesar disso, não é possível uma avaliação

conclusiva, não sendo recomendável o uso desta regressão para se estimar custos de instâncias.

O experimento de regressão linear **rl-linpack** utilizou dados do benchmark LINPACK aplicados em instâncias da Amazon provenientes de [Ostermann 2009]. Não foram obtidos resultados significativos, visto que apenas 5 tipos de instâncias foram considerados no experimento, conforme pode ser visualizado na Figura 6.

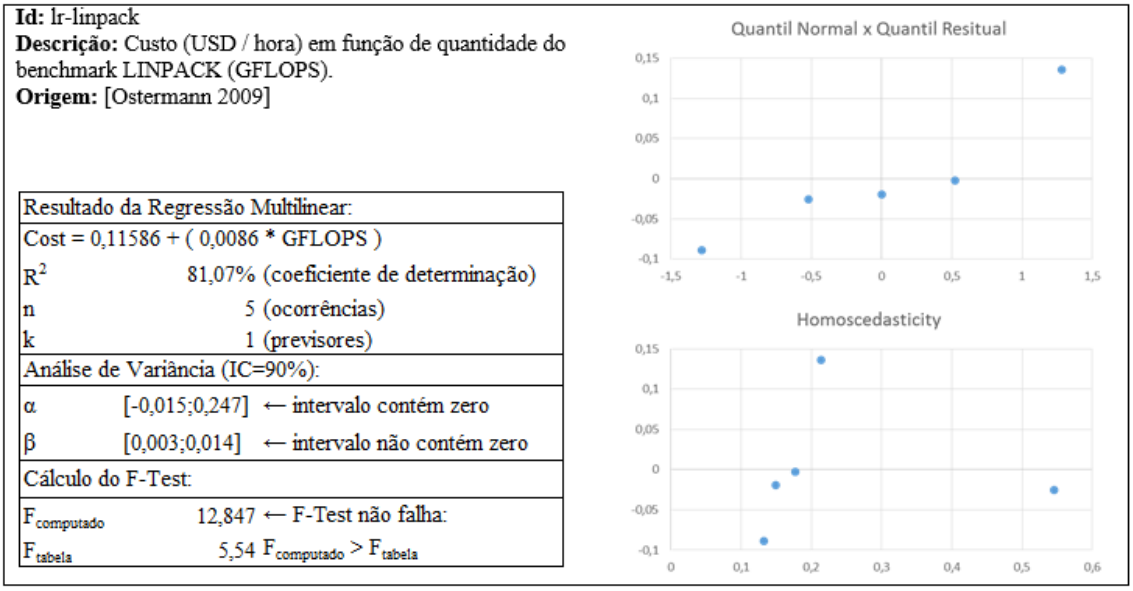


Figura 6. Quadro-resumo do experimento rl-linpack com dados da Amazon EC2 de [Ostermann 2009] provenientes da aplicação do benchmark LINPACK.

No experimento da Figura 6, o coeficiente linear α não é estatisticamente relevante e o gráfico da homocedasticidade revela uma tendência de não espalhamento. Outro problema encontrado é a quantidade de ocorrências reduzida (n=5) para realização do experimento. Sendo assim, não é recomendável o uso da regressão e novos experimentos com mais dados devem ser realizados para comprovação desta hipótese.

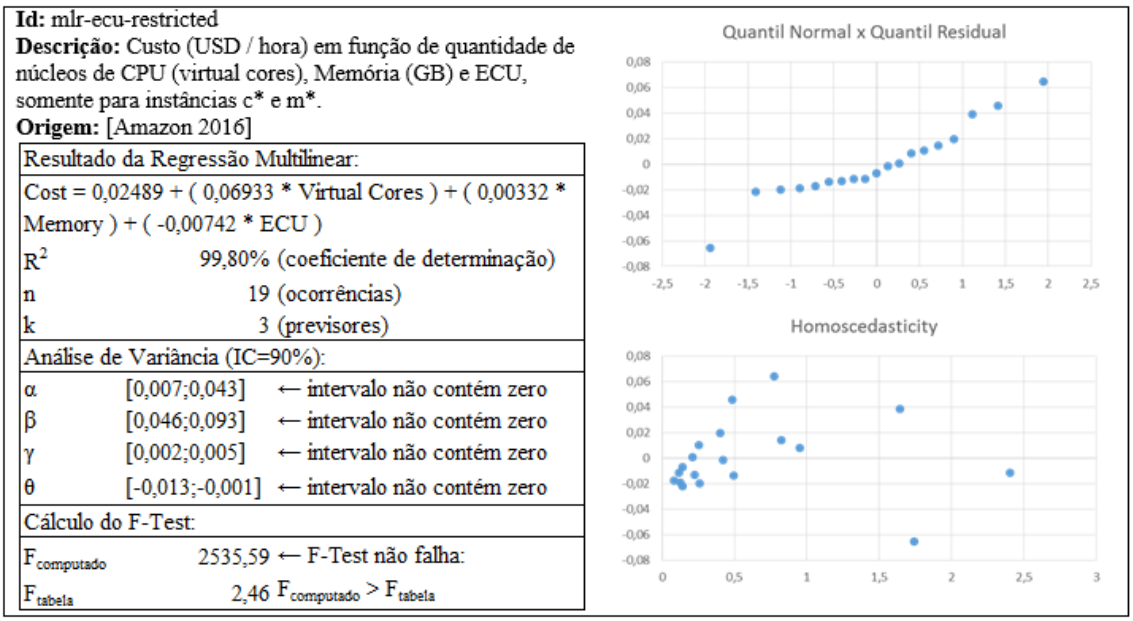
Os melhores resultados foram obtidos nos experimentos que utilizaram os dados mais recentes da Amazon EC2, coletados de [Amazon 2016]. O experimento **mlr-ecu-restricted** aplica a técnica de regressão multilinear, considerando um subconjunto de 19 tipos de instâncias cujo propósito é a utilização de processamento ou memória (*compute-optimized* e *memory-optimized*), ou seja, instâncias do tipo c.* ou m.*, respectivamente. Nesse subconjunto, fica clara a alta influência das quantidades de núcleos de CPU e de memória na composição do custo através do valor do coeficiente de determinação R^2 obtido.

Outro diferencial do experimento **mlr-ecu-restricted** foi o uso de mais um previsor, a medida de ECU (EC2 Compute Unit), onde 1 ECU é equivalente à capacidade de um processador Opteron 2007 ou Xeon 2007, com velocidade de 1.0 a 1.2 GHz [Amazon 2016]. Dessa forma, a equação (1) é modificada para considerar o valor do previsor escalar ECU, cujo coeficiente é θ, sendo assim definida a Equação 2.

$$Custo = \alpha + \beta * Processador + \gamma * Memória + \theta * ECU \quad (2)$$

Onde: *Custo* = custo em valores monetários (em $/hora);
Processador = quantidade de núcleos de processamento (escalar);
Memória = tamanho da memória (em GB);
ECU = quantidade de ECU (escalar);
α = coeficiente linear da equação;
β = coeficiente angular do fator Processador;
γ = coeficiente angular do fator Memória;
θ = coeficiente angular do fator ECU.

A Figura 7 mostra os resultados iniciais obtidos no experimento **mlr-ecu-restricted**. Apesar do coeficiente de determinação R^2 obtido foi alto (99,8%) e significância estatística de todos os fatores, o gráfico da homocedasticidade contém uma tendência de aglomeração. Dessa forma, não é possível afirmar que os dados são homocedásticos e nem tão pouco a distribuição dos resíduos segue uma normalidade.



Id: mlr-ecu-restricted
Descrição: Custo (USD / hora) em função de quantidade de núcleos de CPU (virtual cores), Memória (GB) e ECU, somente para instâncias c* e m*.
Origem: [Amazon 2016]

Resultado da Regressão Multilinear:		
Cost = 0,02489 + (0,06933 * Virtual Cores) + (0,00332 * Memory) + (-0,00742 * ECU)		
R^2	99,80% (coeficiente de determinação)	
n	19 (ocorrências)	
k	3 (previsores)	
Análise de Variância (IC=90%):		
α	[0,007;0,043]	← intervalo não contém zero
β	[0,046;0,093]	← intervalo não contém zero
γ	[0,002;0,005]	← intervalo não contém zero
θ	[-0,013;-0,001]	← intervalo não contém zero
Cálculo do F-Test:		
$F_{computado}$	2535,59	← F-Test não falha:
F_{tabela}	2,46	$F_{computado} > F_{tabela}$

Figura 7. Quadro-resumo do experimento mlr-ecu-restricted com dados da Amazon EC2 de [Amazon 2016] com uso da métrica ECU.

Assim, buscando amenizar a tendência revelada no gráfico de homocedasticidade e considerando a alta correlação entre os fatores *Virtual Cores* e ECU, foi realizado o experimento **mrl-restricted-no-core** considerando apenas os fatores *Memory* e ECU. Outro ponto considerado foi a característica de progressão geométrica de razão 2 no fator ECU, o que indica uma potencial necessidade de uso de uma transformação logarítmica. Foi também utilizada a transformação logarítmica na variável resposta Custo, obtendo-se a Equação 3. Os resultados do experimento podem ser vistos na Figura 8.

$$ln(Custo) = \alpha + \gamma * Memória + \theta * ln(ECU) \quad (3)$$

Id: mlr-restricted-no-core
Descrição: Custo (USD / hora) em função de quantidade de núcleos de Memória (GB) e ECU, somente para instâncias c* e m*.
Origem: [Amazon 2016]

Resultado da Regressão Multilinear:		
ln(Cost) = -3,73522 + (0,004 * Memory) + (0,842 * ln(ECU))		
R^2	98,30% (coeficiente de determinação)	
n	19 (ocorrências)	
k	2 (previsores)	
Análise de Variância (IC=90%):		
α	[-3,948;-3,523]	← intervalo não contém zero
β	[0,002;0,006]	← intervalo não contém zero
γ	[0,762;0,923]	← intervalo não contém zero
Cálculo do F-Test:		
$F_{computado}$	463,718	← F-Test não falha:$F_{computado} > F_{tabela}$
F_{tabela}	2,67	

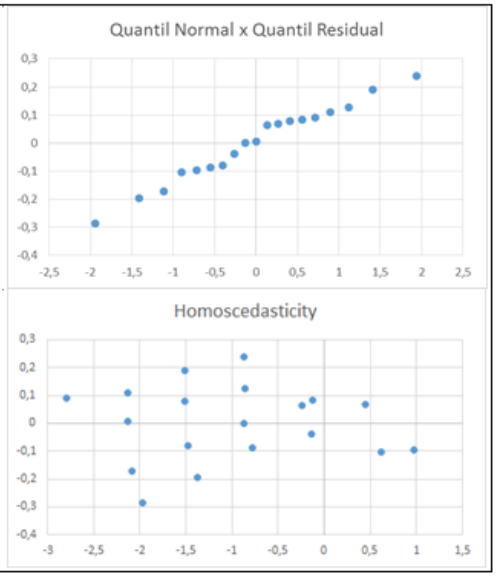


Figura 8. Quadro-resumo do experimento mlr-estricted-no-core com dados da Amazon EC2 de [Amazon 2016], desconsiderando o fator *Virtual Cores*.

Finalmente, foi aplicada a equação resultante do experimento **mlr-ecu-restricted** (Figura 7) em outros tipos de instâncias *(memory_optimized_large*) Amazon EC2, para tentar determinar a aplicabilidade da mesma para tipos de instâncias com propósitos semelhantes ao da equação, ou seja, para uso de processador ou memória. Essa comprovação pode ser visualizada na Tabela 3, especificamente nas colunas que demonstram o erro (escalar e percentual), ou seja, a diferença entre o valor real cobrado pela Amazon e o valor estimado usando a equação. Como pode ser visto, nossa fórmula possui um erro bem pequeno (<5%) para a maior e menor instância consideradas (*x1.32xlarge* e *r3.large*) e erro razoavelmente pequeno (<15%) para as demais instâncias.

Tabela 3. Aplicação da equação da regressão multilinear do experimento mlr-ecu-restricted em tipos de instância Amazon EC2 *memory optimized*.

Instance Purpose	Instance Type	vCPU	ECU	Memory (GiB)	On-Demand ($/hour)	Estimado ($/hour)	Erro (Escalar)	Erro (%)
Memory Optimized	x1.32xlarge	128	349	1952,00	13,338	12,79398	0,54402	4%
Memory Optimized	r3.large	2	6,5	15,00	0,166	0,165142	0,00086	1%
Memory Optimized	r3.xlarge	4	13	30,50	0,333	0,307052	0,02595	8%
Memory Optimized	r3.2xlarge	8	26	61,00	0,665	0,58921	0,07579	11%
Memory Optimized	r3.4xlarge	16	52	122,00	1,330	1,153528	0,17647	13%
Memory Optimized	r3.8xlarge	32	104	244,00	2,660	2,282162	0,37784	14%

Entretanto, é importante mencionar que a validade de generalização dessa abordagem requer futuros estudos para tipos de instâncias com outros propósitos, como *GPU Instances* e *Storage Optimized*, visto que, nestes casos, outras variáveis devem ser consideradas na composição dos custos dos tipos de instâncias.

4. Trabalhos Relacionados

[Lee et. al, 2011] publicaram um dos primeiros trabalhos a analisar características de instâncias oferecidas pelo Amazon EC2. O objetivo era escalonar uma aplicação composta de tarefas independentes em um ambiente heterogêneo e compartilhado. Em 2011, existiam somente 11 tipos de instâncias EC2 e, mesmo com esse número

reduzido, os autores concluem que a relação preço vs. desempenho vs. quantidade de memória é complexa no EC2 e sugerem uma abordagem baseada em *ProgressShare* para o escalonamento.

[Zhang et. al, 2012] propuseram o CloudRecommender, uma abordagem declarativa baseada em ontologia com dois objetivos: unificar as especificações de recursos de diversos provedores de nuvem e sugerir instâncias apropriadas para os requerimentos das aplicações. São considerados os seguintes aspectos: computação, rede e *storage* em diversos provedores (Amazon EC2, Microsoft Azure, GoGrid, RackSpace e outros). Com base nos requisitos e na ontologia, o CloudRecommender seleciona os provedores que atendem aos requisitos e os organiza por menor custo agregado.

Em [Arevalos 2016] é realizado um estudo comparativo de modelos de previsão de preços seguindo a política spot (*spot price prediction*). Nesta política, os preços variam de acordo com a oferta e demanda, sendo que o usuário tem sua instância garantida até preço ofertado. No estudo, são contempladas as particularidades de um modelo de mercado, especificamente do modelo de leilão. São utilizadas técnicas de regressão e de análise comparativa dos algoritmos dos diferentes modelos, com o objetivo de se identificar os melhores recursos em um ambiente de mercado altamente dinâmico.

Em [Tanaka 2014] também é analisado um modelo de mercado baseado em leilões invertidos como estratégia para composição de preços dos recursos em nuvem. É proposto um algoritmo para seleção de recursos em função de custos e qualidade de serviços (problema NP-difícil), com tempo próximo ao polinomial.

A nosso conhecimento, não foi ainda realizado estudo estatístico que consiga determinar a relação entre número de *cores*, quantidade de memória e preços das instâncias em provedores de nuvem públicos. Caso estabelecida, essa relação poderá ser usada de maneira a simplificar bastante as fórmulas de cálculo de custo.

5. Conclusões e Trabalhos Futuros

O presente artigo apresentou um estudo estatístico sobre a relação entre o número de *cores* e quantidade de memória na estipulação do preço das instâncias *on-demand* do provedor Amazon EC2. As equações resultantes das regressões multilineares dos experimentos da Figura 7 e Figura 8 podem ser utilizadas em estratégias de seleção de recursos no provedor Amazon EC2, para instâncias cujo propósito é o uso intensivo de processador ou memória e o modelo de precificação escolhido é o *on-demand*.

A influência dos recursos de processador e memória no custo dos tipos de instâncias existe, mas há outras variáveis que devem ser consideradas. Na Amazon EC2, o fator ECU deve ser considerado. De forma semelhante, o Google GCE utiliza uma métrica que avalia desempenho de processamento, denominada *Google Compute Engine Units* [Google 2016]. Esta métrica será considerada em experimentos futuros.

Ainda como trabalhos futuros, as seguintes possibilidades podem ser exploradas para obtenção de melhores resultados: (a) analisar outros fatores na composição do custo, tais como: largura de banda, espaço de armazenamento, velocidade e tipo de memória, dentre outros; (b) realizar a execução do benchmark LINPACK em vários tipos de instâncias da Amazon EC2 e verificar se pode ser definida uma relação linear

entre custo e o benchmark; (c) analisar comparativamente os resultados dos provedores Amazon EC2 e Google GCE; (d) analisar dados da Amazon EC2 no modelo de precificação *spot* [Amazon 2016], em que o usuário tem a instância garantida até o preço ofertado pelo usuário; (e) analisar dados de outros provedores de nuvem, como Digital Ocean, GoGrid, Rackspace Cloud e Windows Azure.

Agradecimentos

Este trabalho é parcialmente financiado pelo projeto Capes/PROCAD 183794/2013.

Referências

Amazon (2016), *Amazon EC2 Instance Pricing*. http://aws.amazon.com/ec2/pricing/.

Arevalos, S., López-Pires, F., Baran, B. (2016). A Comparative Evaluation of Algorithms for Auction-Based Cloud Pricing Prediction. *IEEE International Conference on Cloud Engineering*, pages 99-108.

Foster, I., Zaho Y., Raicu, I., and Lu. S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop*, pages 1–10.

Google (2016), *Google Compute Engine Pricing*. http://cloud.google.com/compute/pricing.

Jain. R. (1991) *The Art of Computer System Performance Analysis: Techniques for Experimental Design*, *Measurement, Simulation and Modeling,* John Wiley & Sons.

Kheradmand, S., Meybodi, M. R. (2014). Costumer Needs Aware Pricing Strategy for a Cloud Provider. *4th IEEE International Conference on Computer and Knowledge Engineering*, pages 334-339.

Lee, G., Chun, B., Katz, R. H. (2011) Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud, *Proceedings of the 3rd USENIX conference on Hot topics in cloud (HotCloud),* pages 1-10.

Leite, A. F. (2014), *A User-Centered and Autonomic Multi-Cloud Architecture for High Performance Computing Applications*, PhD Thesis, Université Paris-Sud and Universidade de Brasília, available at //hal.inria.fr/tel-01097295.

Leite, A. F., Alves, V., Rodrigues, G. N., Tadonki, C., Eisenbeis, C. and Melo, A. C. M. A. (2015), Automating Resource Selection and Configuration in Inter-clouds through a Software Product Line Method. *8th IEEE International Conference on Cloud Computing*, pages 726-733, 2015.

Mell, P. and Grance, T. (2011) The NIST definition of Cloud Computing, National Institute of Standards and Technology. *Technical Report SP800-145, NIST Information Technology Laboratory*.

Ostermann S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T. and Epema, D. (2009). A performance analysis of EC2 cloud computing services for scientific computing. *In International Conference on Cloud Computing*, pages 115–131.

Sotomayor B., Montero, R. S., Llorente, I. M. and Foster (2009), I. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*,13(5), pages 14–22.

Tanaka, M., Murakami, Y. (2014) Strategy-proof Pricing for Cloud Service Composition. *IEEE Transactions on Cloud Computing*, 99(PP), pages 1-15.

Zhang, M., Ranjan, R., Nepal S., Menzel, M. and Haller (2012), A., A Declarative Recommender System for Cloud Infrastructure Services Selection, *9th Int. Conf. on Economics of Grids, Clouds, Systems, and Services* (GECON), pages 102-113.