Análise de Duas Estratégias de Caracterização para Computação na Nuvem

Artur Baruchi, Edson Toshimi Midorikawa
Escola Politécnica da Universidade de São Paulo
Laboratory of Architecture and High Performance Computing - LAHPC
São Paulo, SP, Brasil
{artur.baruchi, emidorik}@usp.br

***Abstract*—Entre as principais motivações de adesão à Computação na Nuvem pode-se citar a otimização de recursos computacionais e controle de custos. A melhora no uso de recursos computacionais deve ser alcançada tanto da perspectiva do usuário como do provedor. Entretanto, diferente do que ocorre em Data Centers tradicionais, os recursos da Nuvem são compartilhados entre diferentes usuários e, em geral, o provedor de serviços possui pouco ou nenhuma informação sobre o tipo de carga de trabalho submetido nas máquinas virtuais. Esta cenário pode levar a uma situação de distribuição de carga ruim resultando em violações de SLA e QoS. Através de uma metodologia analítica, este artigo apresenta a avaliação de duas estratégias de caracterização de carga de trabalho, ambas baseadas em técnicas de Aprendizagem de Máquina (Naive Bayes e Árvores de Decisão). Além disso, este trabalho discute e apresenta alguns índices de carga que podem ser coletados por agentes SNMP, impondo pouca sobrecarga ao sistema (em torno de 2%). Os resultados mostram que as Árvores de Decisão são mais rápidas, mas mais sensíveis na variação das métricas. Já o Naive Bayes possui maior precisão em algumas situações, mas os dados precisam ser discretizados para que possam ser utilizados.**

Keywords—Cloud Computing; Workload Characterization; Virtualization;

I. INTRODUÇÃO

Em um ambiente de computação na nuvem, a caracterização da carga de trabalho é uma ferramenta de extrema importância. Por meio de estratégias de caracterização precisas, o provedor de serviços poderá criar políticas de distribuição e balanceamento de carga entre os servidores físicos e, dessa forma, evitar o desperdício de recursos. Além disso, uma alocação precisa é de extrema importância para o cumprimento de Acordos de Níveis de Serviço (SLA) e manutenção da Qualidade de Serviço (QoS).

Diversos problemas surgem ao lidar com técnicas de classificação em computação na nuvem. Ironicamente, a maior parte dos problemas é causada pelas propriedades deste paradigma. Entre as diversas características da computação na nuvem, a elasticidade [1] (capacidade de adaptação a uma determinada carga de trabalho sem *downtime*) é uma das mais importantes. Entretanto, a variação da quantidade de recursos computacionais alocada poderá causar situações de ambiguidades caso a estratégia de classificação utilizada não esteja pronta para lidar com este tipo de circunstância.

Além disso, a coleta e interpretação de métricas das Máquinas Virtuais (MV) também é um desafio não trivial. O problema de *Gap* Semântico [2] torna a coleta e a interpretação dos dados difícil e poderá levar à caracterização imprecisa ou errada.

A maioria das técnicas tradicionais de caracterização de carga é computacionalmente cara [3] e torna-se proibitiva em um ambiente de computação na nuvem com milhares de MVs sendo executadas simultaneamente. As técnicas recentes de caracterização também são muito complexas para serem implementadas dentro de um provedor de serviços, pois a vasta maioria baseia-se em métricas especializadas no Monitor de Máquinas Virtuais (MMV) [4], ou em alterações na própria MV e no MMV subjacente [5], [6]. Além disso, métricas especializadas em um determinado MMV acabam sendo impraticáveis em um ambiente composto por diferentes MMVs [7], [8].

A partir dos problemas discutidos anteriormente, foram definidas cinco atributos desejáveis que uma estratégia de caracterização de cargas deve possuir para ser bem-sucedida em um ambiente de computação na nuvem. Estes atributos são:

- Independência: As métricas coletadas devem ser independentes da tecnologia de virtualização empregada (ex.: Paravirtualização ou Tradução Binária), consequentemente, independente do MMV;
- Não Intrusivo: Os dados não podem ser gerados ou coletados por alterações no código-fonte ou no comportamento da MV e do MMV;
- Seguro: Com um ambiente compartilhado entre diversos usuários (além dos servidores físicos, a infraestrutura de rede também é compartilhada), os dados não podem ser observados ou alterados por terceiros;
- Baixa Complexidade: Os algoritmos de caracterização devem possuir baixa complexidade computacional (tempo linear ou, no pior caso, polinomial);
- Precisão: A caracterização deve ser a mais exata possível. Uma estratégia de caracterização imprecisa pode ser mais nocivo do que a ausência de caracterização.

Utilizando uma abordagem analítica, este trabalho fará uma comparação entre duas estratégias para caracterização de carga. Ambas são oriundas do campo da Aprendizagem de Máquina. A primeira técnica analisada será a NaiveBayes [9], que é baseada em probabilidades. A segunda técnica discutida será a Árvore de Decisão [9].

Além da avaliação das estratégias para caracterização, este trabalho apresentará alguns índices de carga que serão utilizados como dados de entrada para os algoritmos de

caracterização. Os índices de carga desenvolvidos são facilmente implementados e coletados por SNMP [10] e, por isso, podem ser utilizados em qualquer Sistema Operacional que tenha suporte a este protocolo.

O restante do artigo segue organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados, com foco, principalmente, em trabalhos que abordam caracterização de cargas de trabalho em MVs e Computação na Nuvem. Os índices de carga elaborados serão discutidos e apresentados na Seção 3. A Seção 4 abordará os algoritmos de caracterização que serão avaliados. Os resultados das comparações entre os dois algoritmos, utilizando *benchmarks* e uma aplicação científica, são apresentados na Seção 5 e, por fim, a conclusão e os trabalhos futuros compõem a Seção 6.

II. TRABALHOS RELACIONADOS

Na literatura, é possível encontrar uma boa variedade de estratégias de caracterização de cargas de trabalhos. Entretanto, de forma geral, o assunto dificilmente é tratado como foco principal (particularmente em computação na nuvem). As técnicas de caracterização acabam ficando em segundo plano, pois são apenas parte de estratégias mais complexas e elaboradas de políticas de balanceamento de carga.

Para um melhor entendimento, esta seção foi dividida em duas subseções. Primeiramente, os trabalhos relacionados à caracterização de carga em MVs serão apresentados e, em seguida, trabalhos referentes à caracterização de cargas em ambientes de Computação na Nuvem.

A. Caracterização em Máquinas Virtuais

No artigo [11], os autores têm como principal objetivo desenvolver uma metodologia para coletar métricas de um ambiente virtualizado. O principal argumento do trabalho é que as métricas para caracterização devem ser coletadas na camada do MMV, com alguma ajuda da MV.

Apesar dos resultados obtidos serem muito precisos, a sobrecarga imposta pela estratégia apresentada superou 20% em alguns *benchmarks*. Com este resultado, a conclusão do trabalho é que novas métricas devem ser elaboradas e, preferencialmente, com nenhuma ou muito pouca alteração no MMV. O presente trabalho tem como premissa o uso de métricas que causam pouca sobrecarga e são coletadas a partir do Sistema Operacional convidado.

O trabalho apresentado em [6] também baseia-se em eventos ou métricas coletadas na camada do MMV. Entretanto, diferente da estratégia do trabalho anterior, os autores não fazem alterações no MMV e utilizam um *software* de *tracing* para caracterizar a carga de trabalho.

Algumas similaridades com o presente trabalho, como o princípio de evitar alterações na MV e no MMV e também o uso de um algoritmo baseado em técnicas de Aprendizagem de Máquina (os autores usam correlação), são observadas. Entretanto, as métricas utilizadas são restritas a um único MMV, limitando seu uso a ambientes homogêneos. Além da baixa sobrecarga, os índices definidos nesta pesquisa podem ser usados independente da tecnologia de virtualização.

Em [12], foi proposta uma metodologia de caracterização de carga como parte de uma estratégia para consolidação de MVs. O objetivo principal dos autores é a diminuição do consumo de energia por meio da consolidação.

Este é um dos poucos trabalhos que não faz uso de métricas especializadas do MMV para realizar a caracterização e, por consequência, é a principal semelhança com o trabalho apresentado neste artigo. Entretanto, a complexidade da estratégia apresentada em [12] limita de forma significativa o trabalho. Por esta razão, os autores utilizam apenas duas categorias de classificação (*Data-Intensive* ou *CPU-Intensive*). A adição de novas classes torna a complexidade da classificação inviável (os autores utilizam uma variação do "problema do mochileiro" para a consolidação). A complexidade computacional é uma preocupação neste trabalho devido à grande quantidade de MVs presente em um ambiente de computação na nuvem. A adição de novas MVs não pode representar aumento de tempo ou perda de desempenho durante o processo de caracterização de carga de trabalho.

B. Caracterização em Computação na Nuvem

O trabalho apresentado em [13] tem como proposta a alocação de MVs em um mesmo servidor físico. Entretanto, o trabalho argumenta que, ao alocar diferentes MVs em um mesmo servidor, é desejável que as cargas de trabalho sejam diferentes (ou complementares), de tal forma que a concorrência para utilizar um determinado recurso seja menor. Os autores não apresentam detalhes sobre a coleta de métricas, mas o artigo versa sobre a importância da caracterização de cargas de trabalho na computação na nuvem e fazem uma ampla discussão sobre o tópico.

Um trabalho interessante sobre identificação de carga em um ambiente de computação na nuvem é apresentado em [14]. Neste artigo, os autores têm como principal objetivo a identificação de picos de utilização cíclicos (por meio de análises temporais, como Média Móvel Exponencial e Transformada de Fourier). Apesar de haver algumas similaridades com o presente artigo, como métricas independentes do MMV e a comparação entre duas estratégias, o foco maior dos autores é a predição do comportamento da carga com análises temporais, que está fora do escopo deste trabalho, mas poderia ser adicionado às estratégias apresentadas neste trabalho.

A proposta apresentada em [15] tem como objetivo a criação de uma metodologia capaz de identificar e predizer a carga de um conjunto de MVs. Neste trabalho, os autores argumentam que, de forma geral, o comportamento de uma determinada MV está correlacionado com o comportamento de outras MVs (conclusão que os autores chegaram após observar a carga de diversas MVs de um conjunto de usuários da computação na nuvem). Este comportamento se deve, principalmente, ao tipo de negócio do usuário e também ao tipo de arquitetura *Multi-Tier*, que é muito comum em ambientes de computação na nuvem. A predição de carga não está no escopo deste trabalho, mas poderá ser adicionada futuramente com o uso de análises temporais (como média móvel exponencial). A correlação entre cargas de trabalho é algo que tem sido bastante usado nos trabalhos similares e

também é uma estratégia que poderá ser utilizada para complementar as estratégias discutidas neste artigo.

III. ÍNDICES DE CARGA

De acordo com a definição dada em [16], um índice de carga é uma métrica cujo objetivo é quantificar o estado da carga em um dado instante ou em um intervalo de tempo que dependerá da frequência em que a métrica é atualizada. São definidas também algumas características importantes para um índice de carga, citadas a seguir[16]:

- A métrica deve refletir com certa precisão o real estado do sistema;
- Deve ser facilmente calculado ou coletado, gerando o mínimo de sobrecarga no sistema;
- O índice deve ser capaz de absorver e representar de maneira correta picos de carga;
- Capacidade de medir, com a mesma precisão, sistemas heterogêneos.

Um índice de carga poderá ser classificado como simples ou composto, que dependerá da extensão e composição da métrica. Os índices classificados como simples são baseados em apenas um valor e, geralmente, representa o uso de apenas um recurso computacional (ex.: o uso de memória ou I/O). Em contrapartida, os índices compostos representam a combinação de uso de diversos recursos em um único valor.

Neste trabalho, foram desenvolvidos índices de cargas simples que são submetidos aos algoritmos de caracterização. Dessa forma, o resultado final da caracterização leva em conta todos os índices coletados e não apenas um único índice, melhorando a precisão dos resultados. Foi verificado, também, que a coleta dos índices de carga apresentados causam uma sobrecarga média de apenas 2% ±1.

Os índices de carga utilizados estão relacionados a dois recursos computacionais, Processador e Memória. Entretanto, é importante salientar que, apesar dos índices estarem associados somente a dois recursos computacionais, é possível identificar outros tipos de cargas (como I/O).

A. Índices de Carga de Processador

Para avaliar o uso do processador, são coletados cinco índices distintos. A Tabela I apresenta uma breve descrição dos índices que são coletados nas MVs e o intervalo dos valores que o índice retorna. Note que as métricas usadas são amplamentes disponíveis na maioria dos Sistemas Operacionais de propósito geral.

Os índices P1 e P3 foram escolhidos por indicar uso do processador em períodos de uso intensivo do Processador. As cargas com esta característica forçam outros processos a esperarem por uma liberação do Processador e, com isso, aumentam a fila de processos (*run queue*).

Os índices P2 e P4, apresentados na Tabela I, tem como objetivo inferir cargas de IO ou uso intensivo de Memória. Estes índices medem o tempo gasto em processamento de rotinas do *Kernel* e esperando por operações de I/O (como acesso a disco ou à memória). O índice P5 tem a função de monitorar o uso do processador e ativar os mecanismos de caracterização.

TABELA I. ÍNDICES DE CARGA DE PROCESSADOR

ID. Índice	Descrição	Valor (Intervalo)
P1	Mede a razão entre o tamanho da fila de processos (*run queue*) e a quantidade de processadores.	$[0, \infty_+)$
P2	Mede a proporção entre o tempo gasto de processamento em rotinas do Kernel (*sys*) e o tempo total.	[0, 1]
P3	Mede a proporção entre o tempo gasto de processamento do usuário (*usr*) e o tempo total.	[0, 1]
P4	Mede a proporção entre o tempo gasto de processamento esperando por operações de I/O (*I/O Wait*) e o tempo total.	[0, 1]
P5	Mede a proporção entre o tempo gasto de processamento ocioso (*Idle*) e o tempo total.	[0, 1]

B. Índices de Carga de Memória

Ao todo, serão avaliados seis índices distintos relacionados à Memória. Os índices, apresentados na Tabela II, indicam a intensidade do uso de memória e também a forma como a memória está sendo usada no Sistema Operacional.

TABELA II. ÍNDICES DE CARGA DE MEMÓRIA

ID. Índice	Descrição	Valor (Intervalo)
M1	Mede a proporção entre a quantidade de Páginas Modificadas (*dirty pages*) e a quantidade total de páginas.	[0, 1]
M2	Mede a proporção entre a quantidade de memória usada para *Cache* e a quantidade total de memória.	[0, 1]
M3	Mede a proporção entre a quantidade de memória usada para *Buffer* e a quantidade total de memória.	[0, 1]
M4	Mede a proporção entre a quantidade de memória Livre (*Free*) e a quantidade total de memória.	[0, 1]
M5	Mede a proporção entre a quantidade de memória Ativa (memória **Não** sujeita à paginação) e a quantidade total de memória.	[0, 1]
M6	Mede a proporção entre a quantidade de memória Inativa (memória sujeita à paginação) e a quantidade total de memória.	[0, 1]

Assim como ocorre com os índices de Processador, os índices que estimam o comportamento e uso da Memória podem ser facilmente coletados na maioria dos Sistemas Operacionais de propósito geral. O intervalo e a forma de coleta dos índices pode variar de acordo com a necessidade de cada situação. Neste trabalho, os índices serão coletados a cada 10 segundos via protocolo SNMP.

Para inferir cargas que dependem de muito acesso e atualização da memória, os índices M1 e M5 são utilizados. Estes índices indicam a quantidade de memória que teve alguma alteração (índice M1) e a quantidade de memória que está sendo usada pelos processos (índice M5). Já os índices M2, M3 e M6 são utilizados para inferir cargas de IO intensivo, pois indicam a quantidade de memória usada para

Caches e *Buffers* de operações de I/O. O índice usado para aferir a quantidade de memória livre, M4, tem como objetivo disparar os algoritmos de caracterização (análogo ao índice P5).

IV. Estratégias de Caracterização

Nesta seção, serão apresentados os dois algoritmos utilizados para caracterizar a carga de trabalho. Ambos os algoritmos são amplamente utilizados na área de Mineração de Dados [9] [17]. A escolha dos algoritmos deste trabalho deve-se, principalmente, à baixa complexidade computacional, característica de grande importância para um ambiente de computação na nuvem.

Além da baixa complexidade, os dois algoritmos compartilham o método de Aprendizagem Supervisionada, isto significa que é necessário submeter os algoritmos a uma massa de dados previamente rotuladas. Para a geração dos dados de aprendizagem, foi utilizado o *Isolation Benchmark*, já avaliado e utilizado em trabalhos anteriores [12] [18].

A. Naive Bayes

O algoritmo de classificação Naive Bayes (NB) é baseado no teorema de Bayes, que é amplamente utilizado na área de probabilidade. O termo *Naive* (em Português ingênuo) indica que o cálculo das probabilidades assume que os eventos (ou, no caso do algoritmo de classificação, as características) são independentes.

Os classificadores Bayesianos (como são chamados os algoritmos baseados no teorema de Bayes) tem como objetivo estimar a classe mais provável de um conjunto de características utilizando as probabilidades conhecidas *a priori*, calculadas a partir dos dados de aprendizagem. O teorema de Bayes exige, no mínimo, três termos – uma probabilidade condicional e duas probabilidades incondicionais – para calcular uma terceira probabilidade condicional [19]. Formalmente, pode-se escrever o teorema de Bayes como:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}, \text{ onde:} \qquad (1)$$

P(Y|X): Probabilidade do evento Y ocorrer dado que X ocorreu;

P(X|Y): Probabilidade do evento X ocorrer dado que Y ocorreu;

P(Y): Probabilidade do evento Y ocorrer;

P(X): Probabilidade do evento X ocorrer.

Uma característica muito importante de classificadores Bayesianos é o resultado quantitativo. Ao submeter uma entrada para classificação, o algoritmo retornará à classe mais provável que aquela entrada possui e sua probabilidade. Dessa forma, é possível tratar possíveis desvios ou elaborar mecanismos de otimizações.

1) Discretização Entrópica

Apesar da boa precisão do algoritmo NB, foi observado que, ao utilizar valores discretos, a precisão melhorou consideravelmente. O processo de discretização consiste, basicamente, em atribuir um número inteiro para um determinado intervalo de valores.

Um dos desafios das técnicas de discretização é a definição de critérios para dividir o conjunto de dados da melhor forma possível. Para este fim, foi utilizada a técnica de discretização entrópica (conhecida também como discretização baseada em entropia) [9]. Em um primeiro momento, o algoritmo de discretização seleciona alguns candidatos a serem utilizados como limite inferior e superior do intervalo. Para essa seleção, o algoritmo utiliza uma medida chamada de "entropia de classe", que mede a quantidade de informação necessária para descrever uma determinada classe.

Em geral, o primeiro intervalo escolhido pelo algoritmo é bastante amplo. Utilizando recursão, o algoritmo particiona o intervalo inicial em partições menores até que uma condição de parada seja satisfeita. Estas condições podem variar bastante de acordo com o algoritmo. Neste trabalho, o critério de parada utilizado (versão implementada no *software* de Aprendizagem de Máquina Weka [20]) é o *Minimum Description Lenght* (MDL), que utiliza como critério a quantidade mínima de atributos para definir uma determinada classe.

Para demonstrar a melhora obtida com a discretização no algoritmo NB, foi construída uma matriz de confusão, apresentada na Tabela III. A partir dos dados de treinamento do algoritmo, foram selecionados aleatoriamente 1014 amostras (em torno de 33% do total) de cada classe e submetidas ao crivo do algoritmo. Observa-se que a precisão com a discretização (CD) é consideravelmente melhor do que a caracterização sem a discretização (SD).

TABELA III. Matriz De Confusão (Com e Sem Discretização).

	CPU (Predição)	MEM (Predição)	IO (Predição)	Precisão (%)
CPU (Real)	1013 (SD)	0 (SD)	1 (SD)	99,9 (SD)
	1014 (CD)	0 (CD)	0 (CD)	100 (CD)
MEM (Real)	0 (SD)	1014 (SD)	0 (SD)	100 (SD)
	0 (CD)	1014 (CD)	0 (CD)	100 (CD)
IO (Real)	1 (SD)	43 (SD)	957 (SD)	95,6 (SD)
	2 (CD)	1 (CD)	1011 (CD)	99,7 (CD)

Aparentemente, a classificação de cargas do tipo I/O intensivo (I/O) são as mais difíceis. Sem a discretização, o algoritmo classificou uma amostra do tipo I/O como sendo uma carga do tipo processador intensivo (CPU) e quarenta e três amostras foram classificadas como memória intensiva (MEM).

B. Árvores de Decisão

As árvores de decisão (DT) pertencem a uma classe de técnicas de Mineração de Dados que são baseadas em disciplinas estatísticas, como a regressão linear [17]. O

objetivo das árvores de decisão é encontrar uma relação entre os dados de entrada e o conjunto dos dados utilizados para construir a árvore (os dados de aprendizagem).

Entre as principais características das árvores de decisão, pode-se citar; (1) a facilidade em produzir, interpretar e utilizar as árvores de decisão, e (2) rapidez de adaptação a novos dados, isto é, as árvores podem ser expandidas ou reduzidas com a contabilização de novos dados de aprendizagem. Estas características tornam essa estratégia de caracterização uma opção bastante atrativa em um ambiente de computação na nuvem.

Uma árvore de decisão pode ser dividida em três partes: o nó raiz, que não possui nenhuma aresta de entrada e, pelo menos, uma aresta de saída; os nós que possuem os testes condicionais são chamados nós internos e possuem, pelo menos, uma aresta de entrada e uma de saída; e, por fim, o nó folha que possui a classificação final de acordo com os parâmetros de entrada e as condições satisfeitas nos nós internos. O nó folha não possui nenhuma aresta de saída e, pelo menos, uma aresta de entrada.

Existem diversas implementações e metodologias para se criar uma árvore de decisão. O algoritmo utilizado neste trabalho é o C4.5 (versão implementado no *software* de Aprendizagem de Máquina Weka). Entretanto, diferente do que ocorre com o NB, as árvores de decisão mantiveram a precisão sem necessidade de submeter os dados a um processo de discretização.

V. RESULTADOS

Para avaliar os algoritmos de caracterização, foram definidos dois *benchmarks* e uma aplicação científica, que são descritos na Tabela IV. Os *benchmarks* possuem um comportamento mais constante no decorrer da execução e, por isso, são bons candidatos para observar a confiabilidade e a precisão do algoritmo. Já a aplicação científica, apresenta mudanças de comportamento ao longo de sua execução, possibilitando a verificação da sensibilidade dos algoritmos em situações em que o tipo de carga é alterado.

Os *benchmarks* e a aplicação foram executados em diferentes configurações de MVs (ao todo, quatro configurações distintas, conforme descrição na Tabela V). Em cada uma das configurações, a execução dos *benchmark*s e da aplicação foi repetida por dez vezes e, no decorrer da execução, foram coletadas informações de uso de processador, memória e disco. As MVs foram configuradas em um *hardware* composto por um processador Intel Core2 Quad de 2.66GHz, 2 GB de memória RAM e um disco com capacidade de 500GB de 5400RPM e vazão nominal de 3Gb/s. Na configuração do *software*, foi utilizado como MMV o Xen 4.1.3 [24] e o Sistema Operacional, na máquina física, o OpenSuse 12.1 com o Kernel 3.1.10. Nas MVs foram instaladas imagens do CentOS 5.9, com o Kernel 2.6.18.

TABELA IV. DESCRIÇÃO DOS *BENCHMARKS* E APLICAÇÃO.

Aplicação / *Benchmark*	**Descrição**
SPEC CPU2000 [21]	Um *Benchmark* amplamente utilizado para comparação de sistemas computacionais. Possui diversos subprogramas que estressam o processador. Dentre os programas que compõem o SPEC, neste trabalho, foi utilizado o MCF, que é um programa de otimização combinatória.
LAME [22]	O LAME é um codificador MP3 utilizado como *Benchmark*. Recebe como entrada arquivos em diversos formatos e transforma para o formato MP3. Para os testes apresentados, o arquivo de entrada continha algo em torno de 2,8GB.
openModeller [23]	É uma aplicação científica utilizada na Biologia. Esta aplicação faz a modelagem de distribuição de espécies baseado em pré-requisitos ecológicos (como a composição da vegetação, altitude, etc.) e extrapola o modelo para outras regiões do planeta.

Os resultados das classificações e o uso médio (com desvio-padrão em parênteses) dos recursos computacionais estão sumarizados na Tabela VI. Nesta tabela, as últimas colunas mostram a caracterização pelo algoritmo NaiveBayes (NB) e pelas Árvores de Decisão (DT). A classificação é realizada no decorrer da execução do *Benchmark* ou da Aplicação e, por isso, ocorreram variações de acordo com a carga. Para apresentar a classificação inferida pelos algoritmos, foi elaborada uma denotação que ilustra a carga predominante e a carga secundária (quando houver) da seguinte forma:

CP/CS, onde:

CP: Classificação Predominante;

CS: Classificação Secundária.

É importante observar também que, para cada algoritmo, existe uma classificação de acordo com os índices de carga de processador (coluna CPU) e com os índices de carga de Memória (coluna MEM). Esta discriminação é fundamental, pois permite que os algoritmos de caracterização consigam identificar cargas de trabalho mistas. Isto ocorre, pois algumas cargas podem ficar mais evidentes por meio dos índices de carga de Processador ou Memória.

TABELA V. DESCRIÇÃO DAS CONFIGURAÇÕES DE MVS.

ID. Configuração	**Processador (Quantidade)**	**Memória (GB)**
C1	1	1
C2		2
C3	2	1
C4		2

A coluna que mostra o uso de Processador é a média da utilização total, sem distinção entre processamento na camada do usuário ou de sistema. A utilização da memória não contabiliza o uso de memória em *Cache* e *Buffer*, pois este tipo de memória é o primeiro a ser descartado quando necessário [25].

TABELA VI. SUMÁRIO DAS CARACTERIZAÇÕES OBTIDAS.

Utilização Média Recursos Computacionais					Caracterização			
					NB		DT	
BENCHMARK	CONF. ID	CPU (%)	MEM (%)	IOPS	CPU	MEM	CPU	MEM
SPEC	C1	96 (±18)	17 (±5)	85771 (±851811)	CPU/IO	CPU/MEM	CPU/IO	CPU/MEM
	C2	96 (±18)	9 (±2)	93674 (±898949)	CPU/IO	MEM	CPU/IO	IO/MEM
	C3	49 (±12)	17 (±5)	93226 (±699789)	IO	IO/MEM	IO	MEM/CPU
	C4	49 (±11)	10 (±3)	93830 (±931227)	IO/MEM	IO	IO	IO/MEM
LAME	C1	98 (±15)	7 (±1)	7646221 (±2680530)	CPU/IO	IO/CPU	CPU/IO	IO
	C2	98 (±15)	5 (±1)	7630012 (±2650733)	CPU/IO	IO	CPU/IO	IO
	C3	50 (±11)	7 (±1)	7810689 (±2968733)	IO	IO	IO	IO
	C4	51 (±11)	5 (±1)	7585084 (±2774450)	IO	IO	IO	IO
OpenModeller	C1	100 (±5)	15 (±1)	490064 (±733006)	CPU/IO	IO	CPU/IO	IO
	C2	99 (±11)	8 (±1)	533604 (±1121545)	CPU/IO	IO	CPU/IO	IO
	C3	51 (±11)	15 (±1)	573838 (±1162654)	IO/MEM	IO	IO	IO
	C4	51 (±11)	9 (±1)	568933 (±1177646)	IO/MEM	IO	IO	IO

A métrica utilizada para representar as operações de I/O utilizada neste trabalho é a quantidade de IO realizada no intervalo de um segundo (na sigla, em Inglês, *I/O Operations Per Second* – IOPS). O cálculo dessa métrica sintetiza operações de escrita e leitura em uma única métrica [26].

Nas classificações do SPEC, utilizando a configuração C1 e C2, em que a MV possui apenas um processador configurado, ambos os algoritmos classificam, majoritariamente, como sendo uma carga de CPU intensivo, com algumas flutuações para I/O e Memória. A classificação como I/O deve-se ao fato de que, em alguns momentos durante a execução do *benchmark*, o SPEC contabiliza uma série de informações em um arquivo (com maior intensidade no final, mas ocorre com bastante frequência durante a execução). A classificação como uso intensivo de memória era esperado, pois o algoritmo escolhido dentre os diversos disponíveis no SPEC é o que possui maior dependência do tamanho da memória (fato que pode ser observado na coluna que mostra o consumo da memória, pois é o *benchmark* com maior uso deste recurso).

A Fig. 1 mostra evolução da classificação de cada algoritmo de forma sobreposta com o uso dos recursos no decorrer da execução. Dentre todas as classificações, esta foi a que os algoritmos mais divergiram. As classificações utilizando os índices de CPU ficaram muito parecidos, entretanto, nas classificações que utilizam os índices de Memória, houve discordância na classificação majoritária. Não foi possível identificar, ainda, a razão para esta diferença, mas a principal suspeita é de que, ao discretizar os valores para submeter ao algoritmo NB, os valores para este *benchmark* (nesta configuração) acabam ficando no limiar entre a categoria de uso intensivo de Memória e uso intensivo de I/O, e, consequentemente, acabam sendo classificados como I/O.

A classificação do Lame foi a que representou menor dificuldade para os algoritmos. Este *benchmark* faz a decodificação de arquivos de áudio para o formato MP3, e possui um comportamento de CPU e I/O intensivo. A decodificação é um processo que exige bastante uso de processador e, conforme a decodificação ocorre, o Lame escreve o arquivo MP3 de saída, gerando bastante operações de IO (conforme a a Tabela VI, é o *benchmark* com maior índice de IOPS).

O uso de CPU fica mais evidente na configuração C1 e C2 (isso acaba ocorrendo em todos os *benchmark*s e no OpenModeller), pois, nesta configuração, há apenas um processador disponível. Entretanto, nas configurações C3 e C4, a predominância da classificação de I/O deve-se à ociosidade do segundo processador e, dessa forma, os algoritmos acabam por classificar a carga como I/O intensivo.

Em seguida, a caracterização do OpenModeller que representa uma classe de aplicação científica. Observa-se, pela Tabela VI, que o OpenModeller tem uma característica de uso bastante intensivo de processador, com alguma dependência de memória e com diversas operações de I/O no decorrer de sua execução. Ambos os algoritmos conseguem classificar de forma bastante precisa o OpenModeller, a classificação predominante como CPU intensivo é em decorrência da configuração. Entretanto, nas configurações C3 e C4, em que há dois processadores e quantidade de memória suficiente, o OpenModuler é classificado como carga de uso Intensivo de Memória e I/O.

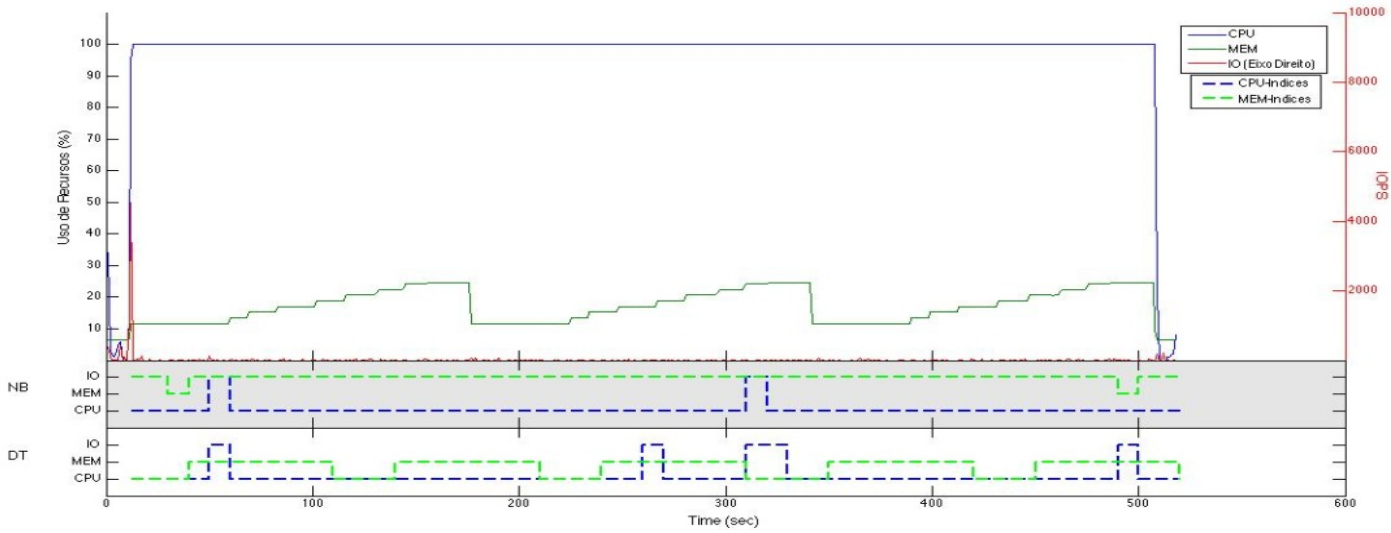

Fig. 1. Caracterização do *Benchmark* SPEC na configuração C1.

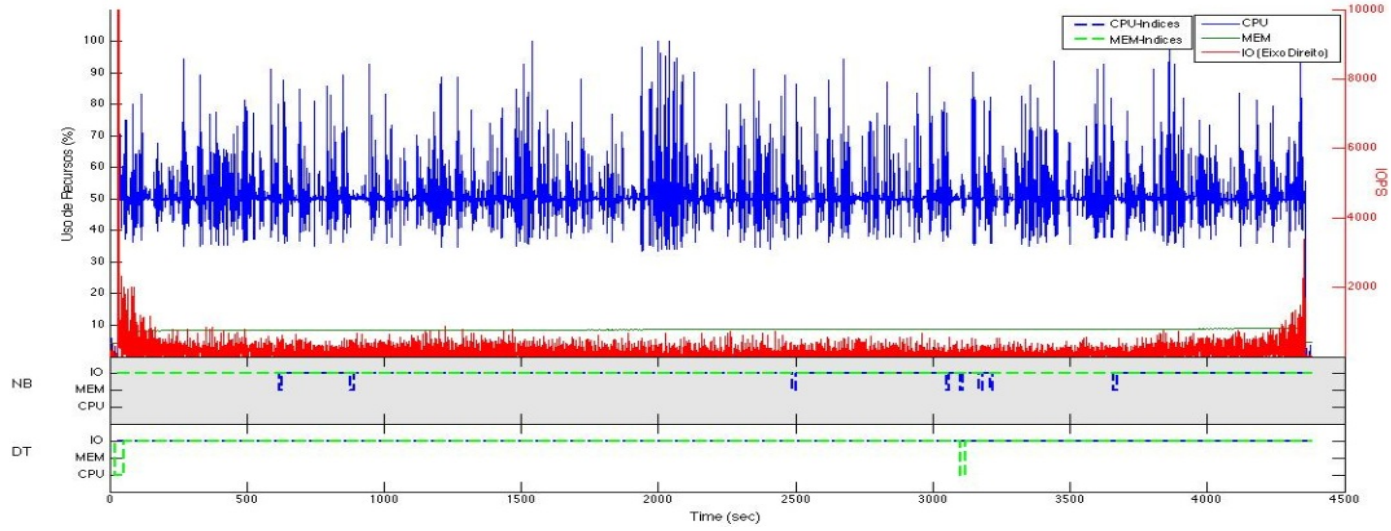

Fig. 2. Caracterização do OpenModeller na configuração C4.

A caracterização realizada pelos dois algoritmos apresentados na Fig. 2 mostra as poucas variações que ocorreram nesta aplicação na configuração C4. Nesta configuração (com maior quantidade de processador e memória), o OpenModeller acaba ficando mais dependente das operações de I/O e, dessa forma, sua carga é predominante I/O intensivo. A Fig. 2 mostra também que o openModeller realiza um grande número de operações de I/O no início e no final da execução. Isto ocorre devido à leitura dos arquivos de entrada e, posteriormente, à escrita dos resultados da modelagem.

VI. CONCLUSÃO

Foi apresentada neste trabalho uma comparação entre dois algoritmos de caracterização de carga de trabalho com foco em um ambiente de computação na nuvem. Para isso, foram definidos índices de carga que podem ser facilmente coletados das MVs e, principalmente, não dependem de modificação no MMV ou de uma determinada tecnologia de virtualização para funcionar. Além disso, este trabalho definiu as principais características que uma estratégia de caracterização de carga deve possuir para um ambiente de computação na nuvem.

Os dois algoritmos de caracterização analisados neste trabalho, obtiveram um bom desempenho na classificação das cargas de trabalho dos *benchmarks* e do openModeller. A caracterização com as Árvores de Decisão possui maior sensibilidade a picos de utilização e sofre maior variação no decorrer da execução das cargas de trabalho. Já a caracterização NaiveBayes possui maior estabilidade devido ao processo de discretização que os dados são submetidos antes de serem usados no algoritmo.

Analisando a complexidade assintótica das duas estratégias, ambas possuem complexidade linear. As Árvores de Decisão têm complexidade dada por $\Theta(i)$, onde i é a quantidade de nós. Já ao algoritmo NaiveBayes, deve-se contabilizar a complexidade da discretização e do cálculo das probabilidades. A discretização dos índices coletados é apenas uma sequência de comparações que deverá ser feita

para cada uma das métricas, portanto, a complexidade do processo de discretização é de $\Theta(k)$, onde k é a quantidade de índices a serem discretizados e a complexidade do cálculo de probabilidades é em função da quantidade de classes e é dada por $\Theta(n)$, sendo que n é o número de classes que terão a probabilidade calculada. Com isso, a complexidade total da estratégia NaiveBayes é $\Theta(n+k)$. Contudo, como as árvores desenvolvidas neste trabalho possuem uma quantidade menor de nós do que a soma da quantidade de índices e de classes, as árvores acabam sendo ligeiramente mais rápidas para se inferir uma classificação sobre o algoritmo NB.

As estratégias de caracterização apresentadas satisfazem as características discutidas na Introdução e, dessa forma, viabilizam a caracterização em grande escala de MVs. Além disso, será possível desenvolver ou melhorar políticas de consolidação de MVs, que poderiam levar em conta as variações de carga de uma determinada carga e, através de análises temporais, realizar predições sobre o comportamento da carga.

REFERÊNCIAS

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, “Above the Clouds: A View of Cloud Computing,” 2010.

[2] L. Cherkasova, D. Gupta, and A. Vahdat, “When virtual is harder than real: Resource allocation challenges in virtual machine based it environments,” *Hewlett-Packard Labs, Tech. Rep. HPL-2007-25*, 2007.

[3] O. ElMoustapha, W. James, and Y. Charles, “On the Comparison of Regression Algorithms for Computer Architecture Performance Analysis of Software Applications,” presented at the First Workshop on Statistical and Machine learning approaches applied to ARchitectures and compilaTion (SMAT07), Ghent, Belgica, 2007.

[4] Irfan Ahmad. 2007. Easy and Efficient Disk I/O Workload Characterization in VMware ESX Server. In Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization

[5] Jiaqing Du, Nipun Sehrawat, and Willy Zwaenepoel. 2011. Performance profiling of virtual machines. In Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '11). ACM, New York, NY, USA, 3-14.

[6] F. Azmandian, M. Moffie, J. G. Dy, J. A. Aslam, and D. R. Kaeli, “Workload Characterization at the Virtualization Layer,” presented at the Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 63–72.

[7] M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har’El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour, “The Turtles project: Design and implementation of nested virtualization,” pp. 1–6, 2010.

[8] F. Zhang, J. Chen, H. Chen, and B. Zang, “CloudVisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization,” pp. 203–216, 2011.

[9] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining Practical Machine Learning Tools and Techniques*, 3rd ed. Morgan Kaufmann, 2005.

[10] D. R. Mauro and K. J. Schmidt, “Essential SNMP, Second Edition,” *Essential SNMP, Second Edition*, Sep. 2005.

[11] H. Mousa, K. Doshi, and E. Ould-Ahmed-Vall, “Characterizing performance in virtualized execution,” *Department of Computer Science, University of California, Santa Barbara, CA*, vol. 93106, 2008.

[12] Jyun-Shiung Yang; Pangfeng Liu; Jan-Jan Wu, "*Workload characteristics-aware virtual machine consolidation algorithms,*" Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conf. vol., no., pp.42,49, 3-6 Dec. 2012

[13] J. Wan, F. Pan, and 2. I. 2. I. Congfeng Jiang Parallel and Distributed Processing Symposium Workshops PhD Forum IPDPSW, “Placement Strategy of Virtual Machines Based on Workload Characteristics,” *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*.

[14] I. Davis, H. Hemmati, R. Holt, M. Godfrey, and D. Neuse, “Storm Prediction in a Cloud,” *plg.uwaterloo.ca.*

[15] A. Khan, X. Yan, S. Tao, and Nikos, Anerousis, “Workload characterization and prediction in the cloud: A multiple time series approach,” *Network Operations and Management Symposium (NOMS), 2012 IEEE*.

[16] Branco, Kalinka Regina Lucas Jaquie Castelo.*"Índices de carga e desempenho em ambientes paralelos/distribuídos - modelagem e métricas.* 2004. Tese (Doutorado em Ciências de Computação e Matemát"ica Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2004.

[17] Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Boston: Pearson Addison Wesley, 2005. Print.

[18] A. Baruchi and E. T. Midorikawa, “Hypervisor Agnostic Workload Characterization Of Virtual Machines,” presented at the Parallel and Distributed Computing Systems (PDCS 2012), Las Vegas, 2012, pp. 1–8.

[19] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*, vol. 74. Prentice hall Englewood Cliffs, 1995.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: an update,” *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[21] S. Sair and M. Charney, “Memory behavior of the SPEC2000 *Benchmark* suite,” *IBM TJ Watson Research Center Technical Report*, 2000.

[22] J. D. Johnston and A. J. Ferreira, “Sum-difference stereo transform coding,” presented at the Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on, 1992, vol. 2, pp. 569–572.

[23] M. E. Souza Muñoz, R. Giovanni, M. F. Siqueira, T. Sutton, P. Brewer, R. S. Pereira, D. A. L. Canhos, and V. P. Canhos, “openModeller: a generic approach to species’ potential distribution modelling,” *Geoinformatica*, vol. 15, no. 1, pp. 111–135, Aug. 2009.

[24] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization,” *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[25] D. P. Bovet and M. Cesati, *Understanding the Linux Kernel*. O'Reilly Media, 2008.

[26] A. Gulati, I. Ahmad, and C. A. Waldspurger, “PARDA: proportional allocation of resources for distributed storage access,” pp. 85–98, 2009.