UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

Redução de sobrecarga de monitoramento em ambientes virtualizados através da seleção de contadores de desempenho

Pedro Freire Popiolek

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande - FURG, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientador: Prof. Dr. Odorico Machado Mendizabal
Co-orientador: Profª. Drª. Karina dos Santos Machado

Rio Grande, 2018

Ficha catalográfica

P828r Popiolek, Pedro Freire.
Redução de sobrecarga de monitoramento em ambientes virtualizados através da seleção de contadores de desempenho / Pedro Freire Popiolek. – 2018.
89f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Computação, Rio Grande/RS, 2018.
Orientador: Dr. Odorico Machado Mendizabal.
Coorientadora: Dra. Karina dos Santos Machado.

1. Seleção de Contadores de Desempenho 2. Monitoramento de Sistemas 3. Otimização de Desempenho 4. Ambientes Virtualizados I. Mendizabal, Odorico Machado II. Machado, Karina dos Santos III. Título.

CDU 004

Catalogação na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

Banca examinadora:

Prof. Dr. Adriano Velasque Werhli

Profª. Drª. Thais Christina Webber dos Santos

Dedico esta dissertação a todas as pessoas que passaram pela minha vida deixando um pouco de si, contribuindo com o meu crescimento pessoal e intelectual. Em especial, dedico aos meus pais e professores, responsáveis por me fazer capaz de atingir este grande objetivo.

RESUMO

POPIOLEK, Pedro Freire. **Redução de sobrecarga de monitoramento em ambientes virtualizados através da seleção de contadores de desempenho**. 2018. 89 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

Infraestruturas computacionais modernas fazem uso de recursos virtualizados e são capazes de oferecer ambientes escaláveis e prover elasticidade ao se adaptar a mudanças repentinas de demanda aos sistemas. O uso de contadores de desempenho possibilita o monitoramento de recursos computacionais e contribui para um gerenciamento eficiente deles. A observação de eventos ocorridos no sistema através de contadores de desempenho possibilita uma visão detalhada de como diferentes aplicações impactam na utilização de recursos. Com isso, aplicações em desenvolvimento podem ser otimizadas para melhor aproveitar os recursos disponíveis e atender a requisitos associados ao consumo de energia, desempenho e escalabilidade. Contadores de desempenho também possibilitam o monitoramento de infraestruturas com objetivo de detectar gargalos e verificar se níveis de acordo de serviço são cumpridos. Este trabalho faz um levantamento do uso de contadores de desempenho em diferentes cenários, descreve os tipos de contadores disponíveis em diferentes plataformas, e apresenta algumas ferramentas de monitoramento baseadas em contadores de desempenho. Em adição, são discutidas algumas métricas de desempenho e interpretações para dados de monitoramento. Com base no conteúdo discutido, é proposto um método de seleção de contadores de desempenho para o monitoramento de sistemas. O método proposto possui como objetivo reduzir o custo de monitoramento, e traz como benefícios: um método automático para a seleção de contadores de desempenho; redução de sobrecarga gerada pela atividade de monitoramento ao descartar contadores de desempenho desnecessários. Os resultados experimentais obtidos neste trabalho caracterizam a sobrecarga gerada pela atividade de monitoramento em ambientes virtualizados. E, caracterizam também, a sobrecarga gerada pela atividade de monitoramento utilizando os contadores de desempenho selecionados pelo método proposto. Ademais é realizada uma avaliação qualitativa para os resultados obtidos através de experiência profissional documentada sobre o tema.

Palavras-chave: Seleção de contadores de desempenho, monitoramento de sistemas, otimização de desempenho, ambientes virtualizados.

ABSTRACT

POPIOLEK, Pedro Freire. **Reducing monitoring overhead in virtualized environments by selecting performance counters**. 2018. 89 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

Modern computing infrastructures make use of virtualized resources and are able to provide scalable environments and provide elasticity by adapting to sudden changes in demand to systems. The use of performance counters enables the monitoring of computational resources and contributes to their efficient management. The observation of events occurring in the system through performance counters provides a detailed view of how different applications impact the use of resources. Developing applications can be optimized to better leverage available resources and meet requirements associated with power consumption, performance, and scalability. Performance counters also enable monitoring of infrastructures to detect bottlenecks and check if service agreement levels are met. This work surveys the use of performance counters in different scenarios, describes the types of counters available on different platforms, and presents some monitoring tools based on performance counters. In addition, some performance metrics and interpretations for monitoring data are discussed. Based on the content discussed, a method of selecting performance counters for system monitoring is proposed. The proposed method aims to reduce the cost of monitoring, and brings as benefits: an automatic method for the selection of performance counters; reducing the overhead generated by the monitoring activity by discarding unnecessary performance counters. The experimental results obtained in this work characterize the overhead generated by the monitoring activity in virtualized environments. And, they also characterize the overhead generated by the monitoring activity using the performance counters selected by the proposed method. In addition a qualitative evaluation is carried out for the results obtained through documented professional experience on the subject.

Keywords: Selection of performance counters, systems monitoring, performance optimization, virtualized environments.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AGNES *Agglomerative Nesting*
APIC *Advanced Programmable Interrupt Controller*
CLR *Common Language Runtime*
CMASK *Counter Mask*
CPU *Central Process Unit*
DDR *Double Data Rate*
DIANA *Divise Analysis*
DPC *Deferred Procedure Calls*
E *Edge Detect*
EN *Enable Counters*
KDD *Knowledge-Discovery in Databases*
LBR *Last Branch Record*
MSR *Model-Specific Register*
MV Máquina Virtual
OS *Operating System Mode*
NoSQL *Not Only Structured Query Language*
PaaS *Platform as a Service*
PC *Pin Control*
PDF *Profile-Directed Feedback*
PEBS *Precise Event-Based Sampling*
PGO *Profile-Guided Optimization*
POGO *Profile-Guided Optimization*
PMU *Performance Monitor Unit*
FDO *Feedback-Directed Optimization*
HP *Hewlett-Packard Company*
HPC *High Performance Computing*

HTTP *Hypertext Transfer Protocol*

IaaS *Infrastructure as a Service*

IIS *Microsoft Internet Information Services*

INT *APIC Interrupt Enable*

I/O *Input/Output*

INV *Invert Counter Mask*

IP *Instruction Pointer*

JMX *Java Management Extension*

RAM *Random Access Memory*

RAID *Redundant Array of Independent Disks*

RDBMS *Relational Database Management Systems*

RISC *Reduced Instruction Set Computer*

rpm *Rounds per minute*

SaaS *Software as a Service*

SGBD *Sistema de Gerenciamento de Banco de Dados*

SLA *Service Level Agreement*

SQL *Structured Query Language*

SSE *Sum of Squares Errors*

TCP *Transmission Control Protocol*

TLB *Translation Lookaside Buffer*

TSS *Total sum of squares*

UMASK *Unit Mask*

USR *User Mode*

VCM Vazão Média de Experimento com Monitoramento

VSM Vazão Média de Experimento sem Monitoramento

YCSB *Yahoo! Cloud Serving Benchmark*

SUMÁRIO

1 INTRODUÇÃO

A utilização e gerenciamento de recursos de *hardware* por *software* encontra-se em expansão [7]. Com aplicações mais complexas e um número crescente de usuários de sistemas computacionais, o desenvolvedor e administrador de sistemas precisa preocupar-se com a otimização de desempenho e provisionamento de recursos eficiente [8]. No suporte a esses processos, unidades de monitoramento de desempenho (*PMU*, *Performance Monitor Unit*) são disponibilizadas por processadores de propósito geral [1]. Assim, possibilitando que ocorrências de eventos no processador possam ser mensuradas durante a execução de sistemas através de contadores de desempenho [9]. Além dos dados extraídos da *PMU*, também são utilizados para observação do comportamento de sistemas dados de contadores de desempenho implementados em sistemas operacionais, servidores de aplicação, aplicações, periféricos de I/O e *chipsets* (quando implementados fora do processador) [10].

Com base nos dados coletados pelos contadores de desempenho, desenvolvedores e administradores de sistemas são capazes de avaliar e efetuar ajustes e otimizações no código-fonte ou nas configurações de aplicações para resolver problemas de desempenho. Por exemplo, é possível avaliar se um aplicativo consegue fazer bom uso da hierarquia de memória do sistema observando ocorrências de *cache miss* e *cache hit*. Outra possibilidade é avaliar o aproveitamento de técnicas como o *pipeline*. Informações como ocorrências de bolhas no *pipeline* e previsões de desvio incorretas são de grande interesse para diagnosticar problemas de desempenho [11]. Contadores de desempenho também são importantes para o projeto de compiladores que preveem otimização de desempenho baseados em dados fornecidos pelos contadores [1] [12].

Semelhante ao processo realizado pelo desenvolvedor, que avalia uma aplicação em desenvolvimento, o administrador de sistemas avalia o desempenho e a disponibilidade de sistemas através da leitura de contadores de desempenho. Ao observar o uso de recursos do sistema para diferentes cargas de trabalho, pode-se identificar gargalos e descobrir quais as condições de utilização suportadas pela infraestrutura computacional [2]. O monitoramento de recursos é necessário mesmo após a implantação do sistema. Mudanças na demanda de serviços oferecidos podem acarretar na necessidade de ampliar os recursos

computacionais. Ainda nesse cenário, muitas aplicações oferecem meio de configuração capaz de regular o consumo de recursos do sistema, sendo necessária a leitura de contadores para verificar possíveis gargalos de desempenho.

Diante das oportunidades que a microeletrônica proporciona pelo contínuo ganho computacional, soluções de virtualização têm sido prática comum no processo de consolidação de servidores. Benefícios são trazidos pelo uso de virtualização, como por exemplo, a redução do consumo de energia e serviços de manutenção [13]. No entanto, cada vez mais servidores virtuais são suportados em um único servidor físico. Com isso, é elevada a complexidade de gerenciamento de recursos por serem submetidos a cargas de trabalho variáveis provenientes das máquinas virtuais [14].

Assim como na consolidação de servidores, plataformas de nuvem computacional também utilizam tecnologias de virtualização. Nesse caso, existe uma cobrança maior com suporte a requisitos de desempenho. Nuvens computacionais ofertam serviços (Software as a Service – SaaS, Platform as a Service – PaaS e Infrastructure as a Service – IaaS) com garantias firmadas junto ao usuário através de acordo de níveis de serviço, *SLA* (do inglês, *Service Level Agreement*) [13]. Por isso, técnicas de provisionamento de recursos eficiente são de grande importância, tanto para o provedor quando para os clientes. A alocação dinâmica de máquinas virtuais entre os servidores físicos em tempo de execução proporciona ao provedor um uso eficiente da infraestrutura e, ao mesmo tempo, garante ao usuário os acordos pré-estabelecidos [14].

Contadores de desempenho possuem papel relevante em tarefas dedicadas a otimização de desempenho de aplicações, ajuste de desempenho de servidores, e monitoramento de recursos para diversos fins [15] [16]. Porém, em contrapartida à motivação pela utilização de contadores de desempenho, existe um custo computacional associado à utilização desses recursos [17]. E por consequência, ocorre consumo adicional de energia elétrica, e competição pelo uso de recursos computacionais que poderiam estar disponíveis para as tarefas de propósito do sistema e aplicações. Esses efeitos são mais aparentes quando há um número elevado de contadores sendo monitorados [18] [19]. Em adição, a seleção de contadores de desempenho importantes para a avaliação de um sistema é dependente de um profissional especializado. Mesmo sob essa condição, a seleção e avaliação dos contadores de desempenho estão sujeitas a subjetividade e experiência do profissional responsável.

1.1 Objetivos

Em consideração a importância das atividades que empregam a utilização de contadores de desempenho, este trabalho tem como objetivo geral criar uma metodologia capaz de proporcionar a seleção de contadores de desempenho de forma eficiente. Ou seja, reduzir a quantidade de contadores de desempenho utilizados no monitoramento de sistemas com

pouca perda de informação sobre o estado do sistema monitorado. Com esse propósito, este trabalho possui como objetivos específicos:

- Obter, de forma automática, subconjuntos de contadores de desempenho mais relevantes para diferentes cenários;
- Dispensar o monitoramento de algum subconjunto de contadores de desempenho pouco relevantes ou redundantes;
- Facilitar a escolha de contadores de desempenho para o monitoramento de sistemas.

Em busca dos objetivos estabelecidos neste trabalho, realizou-se uma pesquisa experimental. Esse tipo de pesquisa é baseado na reprodução de um fato sob condições controladas, de forma que seja possível observar e estudar o fato [20]. Neste trabalho o fato observado é a utilização de recursos computacionais do sistema. A observação é realizada através de dados disponibilizados por contadores de desempenho. A reprodução de comportamentos de utilização de recursos é obtida pela execução de *benchmarks* com o intuito de estimular recursos específicos do sistema ou simular aplicações, como sistemas NoSQL (do inglês, *Not only Structured Query Language*) [5].O estudo do fato é realizado através da avaliação dos dados obtidos pelos contadores de desempenho, considerando cada contador de desempenho como uma variável contendo uma série temporal. Assim, viabilizando um estudo de causas e efeitos.

Com base na relação existente entre as variáveis, são encontrados os coeficientes de correlação linear de Pearson para cada uma das possíveis combinações de duas variáveis. O método proposto avalia os coeficientes utilizando abordagem de descoberta de conhecimento em banco de dados. Através da aplicação do método proposto, é possível verificar:

- Ocorrência de correlações em diferentes cenários: diferentes cargas de trabalho são geradas para a realização de experimentos. Em cada cenário de experimentação são verificadas correlações. São verificadas as correlações que aparecem em cenários específicos e as que se repetem em diferentes cenários.
- Possibilidade de redução do número de variáveis: são formados grupos de variáveis que possuem um alto grau de correlação entre si. Dessa forma, uma dessas variáveis pode ser selecionada dispensando as demais. São considerados apenas os grupos que são mantidos para diferentes cenários de utilização de recursos.
- Variáveis que melhor representam o seu grupo de origem: é possível selecionar a variável representativa de cada grupo de variáveis verificando a correlação entre as mesmas, com o objetivo de selecionar aquela que melhor representa as demais. Dessa forma, é composto novo conjunto de variáveis que serão utilizadas para descrever o estado de utilização de recursos do sistema.

- Redução de sobrecarga: ocorre redução de sobrecarga ao reduzir a quantidade de contadores de desempenho utilizados para o monitoramento do sistema.

- Liberação de recursos: através da redução de sobrecarga recursos computacionais tornam-se disponíveis para a aplicação do sistema. Por consequência, ocorre aumento da vazão de operações realizadas pela aplicação.

1.2 Justificativa

A realização desse trabalho justifica-se pela relevância da utilização de contadores de desempenho e pela contribuição em forma de conhecimento científico gerado sobre os dados fornecidos por contadores de desempenho. O conhecimento gerado serve de embasamento para a utilização consciente de contadores de desempenho. Possibilita melhor entendimento sobre o comportamento do sistema quando submetido a diferentes cargas de trabalho. Em adição, pode contribuir para o aprimoramento de técnicas relacionadas ao monitoramento de sistemas. E, além disso, amplia a literatura que expõe na prática conceitos teóricos relacionados a sistemas operacionais e arquitetura de computadores.

A análise de dados fornecidos por contadores de desempenho propicia tomadas de decisão que têm por objetivo: otimização de desempenho de sistemas; e provisionamento eficiente de recursos de infraestruturas computacionais. A grande quantidade de contadores de desempenho disponíveis no sistema permite reproduzir um retrato detalhado do estado do sistema em intervalos de tempo [16]. Porém, torna-se um desafio: a seleção dos contadores de desempenho relevantes para a atividade de monitoramento do sistema; a interpretação de grande quantidade de dados fornecidos pelos contadores; e o entendimento de como os dados podem estar correlacionados. Em contrapartida à utilização de contadores de desempenho, ocorre a geração de sobrecarga ao sistema e, consequentemente, o aumento do consumo de energia e recursos [21].

A identificação detalhada dos custos originados pela utilização de contadores de desempenho implementados pelo sistema operacional, servidores de aplicação e aplicações possui descrição escassa em literatura. Porém, ao utilizar esses contadores de desempenho, assim como ao executar qualquer outro processo no sistema, ocorre a criação de tarefas adicionais que competem pelos recursos disponíveis. As tarefas do processo de monitoramento – juntamente com as demais – partilham o tempo de processador, hierarquia de memória, barramentos, e geram interrupções no sistema [22] [23]. Em adição, recursos de rede podem ser demandados quando os dados fornecidos pelos contadores de desempenho são coletados por um sistema remoto. Quanto maior é a quantidade de contadores utilizados, maior é a quantidade de dados que necessita ser gerada, armazenada ou enviada. A frequência de amostragem utilizada para a coleta de dados de monitoramento também influencia na quantidade de dados gerada [2]. Logo, maior quantidade de

contadores utilizados e maior frequência de amostragem são fatores que contribuem para uma atividade mais intensa do sistema para realizar as tarefas de monitoramento.

Jiang et al. [24] declaram a dificuldade em detectar problemas de desempenho em testes de carga. A dificuldade está em analisar grande quantidade de dados proveniente dos contadores de desempenho disponíveis. Outro desafio está em lidar com a sobrecarga decorrente da atividade de monitoramento do sistema. A sobrecarga por ser elevada torna os dados inválidos, impedindo a sua utilização. Ou seja, os dados também refletem a atividade de monitoramento realizada no sistema, uma interferência do instrumento de medição. Por consequência, torna a análise de desempenho um processo custoso em termos de tempo, devido a quantidade de experimentos necessários, com poucos contadores, para coletar dados válidos para realizar análise de desempenho.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte maneira. No Capítulo 2 é explorada a fundamentação teórica da dissertação. No Capítulo 3 são discutidos os trabalhos relacionados. No Capítulo 4 é descrito o método proposto para seleção de contadores de desempenho. No Capítulo 5 são expostas avaliações experimentais de sobrecarga gerada pela atividade de monitoramento e os efeitos causados pela aplicação do método proposto, assim como, a sua avaliação qualitativa. No Capítulo 6 são realizadas as considerações finais e levantados os trabalhos futuros.

2 FUNDAMENTOS

Este capítulo apresenta conceitos fundamentais para a compreensão do trabalho desenvolvido. Na Seção 2.1 é apresentada uma visão geral sobre contadores de desempenho. Na Seção 2.2 é explorado o emprego de contadores de desempenho. Na Seção 2.3 são descritas informações obtidas através de dados fornecidos por contadores de desempenho. Na Seção 2.4 são descritas técnicas de análise multivariada de dados e, na Seção 2.5, é apresentado o processo de descoberta de conhecimento em banco de dados.

2.1 Contadores de desempenho

Contadores de desempenho são recursos presentes em plataformas computacionais e em aplicações. Contadores de desempenho revelam a utilização de recursos do sistema através de métricas (medições que quantificam a ocorrência de atividades ou eventos), e possuem como principal finalidade o fornecimento de dados numéricos para o suporte a análises de desempenho. Esses dados também são capazes de fornecer indícios sobre o estado ou a saúde de sistemas [2]. Números de trocas de contexto por segundo, medidas pelo sistema operacional, é um exemplo de métrica fornecida por contadores de desempenho [16].

Ibidunmoye et al. em [2] classificam a origem dos dados coletados por contadores de desempenho em nível de sistema ou em nível de aplicação. Nessa classificação, o nível de sistema compreende a plataforma computacional (arquitetura de *hardware* e sistema operacional). Logo, o nível de sistema pode ser dividido nas camadas de *hardware* e de sistema operacional. No nível de aplicação, as aplicações são normalmente serviços, como por exemplo: servidores HTTP, plataformas de tempo de execução e sistemas de gerenciamento de banco de dados.

Abstrações para utilização de contadores de desempenho são oferecidas aos usuários através de ferramentas. Porém, o acesso aos dados fornecidos por contadores de desempenho também pode ser realizado, dependendo do nível de implementação dos contadores, através de: interrupções; arquivos virtuais; bibliotecas; ou por meios específicos implementados pelas aplicações que disponibilizam contadores de desempenho, como

por exemplo, consultas HTTP para servidores Web e consultas em tabelas para SGBDs (sistemas de gerenciamento de banco de dados).

2.1.1 Nível de sistema

Em plataformas computacionais, contadores de desempenho estão implementados em diferentes camadas, sendo capazes de fornecer dados provenientes de diferentes níveis de abstração. Contadores de desempenho implementados no *hardware* são capazes de fornecer dados de mais baixo nível, relacionados ao processamento de microinstruções e ao acesso a memória cache, por exemplo. Contadores de desempenho implementados pelo sistema operacional são capazes de fornecer dados de nível intermediário de abstração, como por exemplo os relacionados ao escalonamento de processos e ao enfileiramento de processos esperando pelo atendimento de periféricos. As seções que seguem exploram os contadores de desempenho nessas diferentes camadas.

2.1.1.1 Camada de hardware

Processadores de propósito geral possuem recursos capazes de mensurar e disponibilizar à camada lógica do sistema métricas sobre eventos ocorridos no próprio processador. Esses recursos são chamados de contadores de desempenho de *hardware* e são disponibilizados através de registradores privilegiados de propósito específico [1] [25]. Recursos similares também são oferecidos por *chipsets* e periféricos de I/O, como controladoras RAID [10].

Em processadores Intel, existe uma unidade chamada PMU (*Performance Monitor Unit*), responsável por implementar contadores de desempenho. O primeiro processador a oferecer recursos de monitoramento foi o processador Pentium, sendo dada sequência de implementação e aperfeiçoamento da unidade aos processadores subsequentes lançados pelo fabricante [1]. A PMU era, inicialmente, um recurso para realização de depuração no processador. A partir do processador Itanium, a PMU passou a disponibilizar recursos de monitoramento aos usuários finais [10]. Em processadores atuais, essa unidade encontra-se distribuída pelo processador. Embora o termo PMU não seja encontrado em manuais de processadores de outros fabricantes, esse termo é empregado de maneira irrestrita na literatura [26].

2.1.1.1.1 PMU em arquitetura Intel 64

Em processadores de arquitetura Intel 64 existem dois grupos de contadores de desempenho: contadores que monitoram eventos pré-estabelecidos pela arquitetura, de função fixa; e contadores de propósito geral, que monitoram determinados eventos selecionados dentre os eventos suportados. Cada contador pode monitorar um evento. A 6ª geração de processadores Intel possui mais de 200 eventos que podem ser monitorados e incluem diversos aspectos do processador, como *pipeline*, memória *cache*, *cache* especial

de tradução de endereços (TLB), barramento, interrupções, dentre outros [1].

Contadores de desempenho de função fixa são implementados com seis registradores (MSRs - *Machine Specific Registers*). Três desses registradores possuem a função de contagem (registradores IA32_FIXED_CTR[1]), sendo cada um reponsável por contabilizar a ocorrênvia de um evento pré-estabelecido pela arquitetura. A utilização dos registradores contadores, chamados de PMC (*Performance Monitor Counters*), é realizada através da configuração de outros três registradores:

Registrador de controle global (IA32_PERF_GLOBAL_CTRL): permite aplicações de monitoramento ativar/desativar a contagem de eventos para todos ou para qualquer combinação de registradores contadores de função fixa.

Registrador de estado global (IA32_PERF_GLOBAL_STATUS): permite aplicações de monitoramento consultar condições de *overflow* de qualquer combinação de registradores contadores de função fixa.

Registrador de controle de *overflow* global (IA32_PERF_GLOBAL_OVF_CTRL): permite aplicações de monitoramento limpar condições de *overflow* de qualquer combinação de registradores contadores de função fixa.

A implementação de contadores de desempenho de propósito geral é realizada através de pares de registradores. O registrador chamado seletor de evento (IA32_PERFEVTSEL MSR) compõe o par com o registrador contador (IA32_PMC MSR), sendo, o primeiro responsável por manter a configuração do evento a ser monitorado, e o segundo por armazenar a contagem de eventos observados [1].

Os eventos suportados pelos contadores de desempenho de propósito geral são divididos em dois grupos: não-arquiteturais e arquiteturais. A configuração do registrador seletor de evento para o monitoramento de eventos não-arquiteturais não é padronizada entre os diversos processadores que dão suporte aos mesmos eventos, enquanto a configuração para eventos arquiteturais é compatível entre diferentes processadores.

Na Figura 1 é apresentada a estrutura do registrador seletor de evento. Esta estrutura está relacionada às facilidades de monitoramento de desempenho arquitetural versão 1. Este registrador permanece compatível com as versões mais recentes que agregam novas funcionalidades e melhorias [1].

Para configurar um registrador seletor de evento é necessário, através do campo *Event Select*, selecionar a unidade lógica responsável por detectar a ocorrência do evento que se quer monitorar; e, no campo UMASK, especificar a condição microarquitetural detectável pela unidade lógica (o evento a ser monitorado). Ambos os campos são de 8 bits. A relação de todos os eventos suportados e respectivas configurações para cada microarquitetura pode ser encontrada no manual do processador [1].

[1] IA32_FIXED_CTR0, IA32_FIXED_CTR1 e IA32_FIXED_CTR2

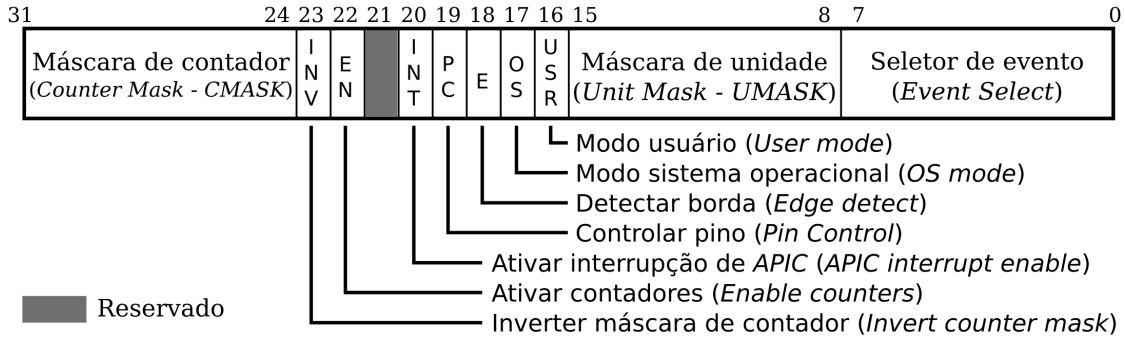


Figura 1: Estrutura do registrador seletor de evento versão 1 (adaptado de [1]).

Em adição à configuração do evento alvo é possível filtrar as ocorrências por nível de privilégio. Eventos gerados pela execução de instruções com privilégios 1, privilégio 2 ou inclusive com privilégio 3 são contabilizados pela ativação do bit USR. O bit OS ativo inclui eventos gerados pela execução de instruções com privilégio 0. A Figura 2 ilustra os níveis de privilégios suportados pela arquitetura. Quanto mais ao centro está o nível, mais alto é o privilégio. O nível de privilégio 0 é o nível mais alto, permite executar qualquer instrução e ter acesso irrestrito à memória e portas de I/O, por isso é utilizado pelo núcleo do sistema operacional. Os níveis de privilégio 1 e 2 são atribuídos à serviços do sistema operacional. O nível de privilégio 3 é atribuído às aplicações em geral. O suporte a níveis de privilégio propõe maior segurança aos sistemas operacionais.

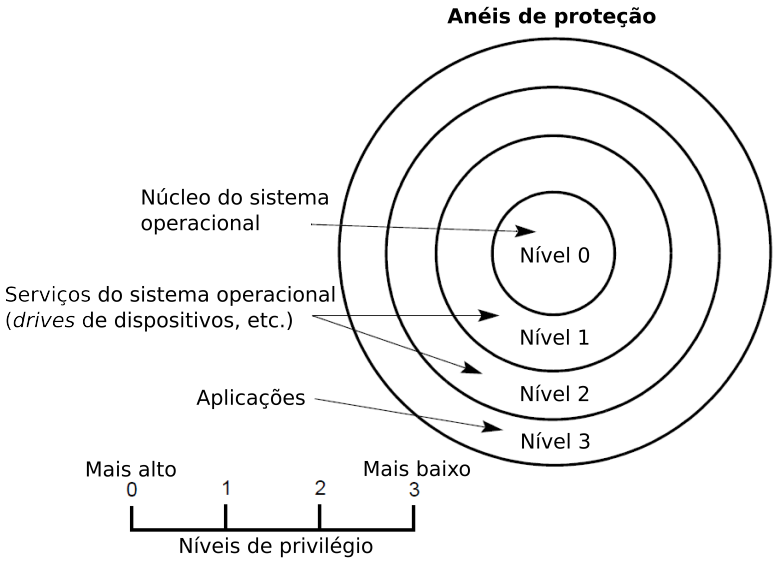


Figura 2: Níveis de privilégio para execução de instruções (adaptado de [1]).

Além de contabilizar os eventos ocorridos, os contadores de desempenho também permitem mensurar o tempo de permanência de determinada condição microarquitetural. A atividade deve ser realizada por *software*, porém o suporte é oferecido pelo bit E. Quando ativo, a unidade lógica detecta as bordas de ocorrências do evento selecionado, permitindo que possa ser mensurado por *software* a fração de tempo ou o tempo médio gasto

em determinada condição.

Os campos INV e CMASK indicam uma condição para o incremento do registrador contador. O valor do campo CMASK é comparado com a quantidade de ocorrências do evento configurado durante um ciclo. O registrador contador será incrementado caso a quantidade de ocorrências for maior ou igual ao valor do campo *counter mask*. Com o bit INV ativo, a condição de incremento muda de "menor ou igual" para "menor que".

O bit PC, quando ativo, permite que pinos PMi indiquem a ocorrência de incremento no registrador contador; ou *overflow*, quando desativo, no mesmo registrador. Associado a cada contador existe um pino de evento (PMi/BPi) que pode ser usado para indicar o estado do contador a um *hardware* externo [27] [28]. O bit INT ativo permite que o processador gere uma interrupção na ocorrência de *overflow* do registrador contador através do APIC (*Advanced Programmable Interrupt Controller*) local.

A ativação/destivação do contador de desempenho de *hardware* é realizada pelo bit EN. Esse bit está presente em apenas um dos registradores seletores de eventos. A configuração desse bit implica na ativação/desativação de outros contadores.

A Figura 3 ilustra um registrador seletor de evento configurado. A combinação dos campos *Event Select* e UMASK, respectivamente com os valores C0H e 00H, seleciona o evento arquitetural *Instruction Retired* [1]. Esse evento ocorre na retirada de instruções do processador. Com os bits USR, OS, EN, INT e EN ativos, os bits E, PC e INV inativos e campo CMASK configurado com valor 01H, cada ocorrência do evento selecionado será contabilizada e uma interrupção será gerada quando ocorrer *overflow* do registrador contador. Os eventos serão contabilizados para todas as instruções executadas, independente do nível de privilégio.

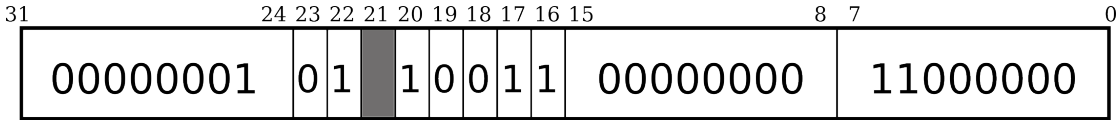


Figura 3: Registrador seletor de evento configurado.

O monitoramento de eventos não-arquiteturais são programados utilizando esse mesmo mecanismo [1]. A Tabela 1 apresenta alguns exemplos de eventos arquiteturais e a Tabela 2 exemplos de eventos não-arquiteturais para a arquitetura Skylake [1]. Também presentes nas Tabelas 1 e 2 estão os respectivos códigos para a configuração dos eventos nos registradores seletores de evento (valor para o campo seletor de evento na coluna *Número do Evento* e valor para o campo máscara de unidade na coluna *Valor UMASK*). Em [1] é possível encontrar a lista completa desses eventos, incluindo os eventos relacionados aos registradores de função fixa.

Os contadores de desempenho de propósito-geral são disponibilizados por núcleo lógico de processamento. Arquiteturas atuais, como a arquitetura Skylake, são implementados quatro contadores de desempenho de propósito-geral para cada núcleo. Caso a

Tabela 1: Exemplos de eventos arquiteturais (aquitetura Skylake).

Número do Evento	**Valor UMASK**	**Mnemonico da máscara do evento**	**Descrição**
C0H	00H	*Instruction Retired*	Instrução retirada
2EH	4FH	*LLC Reference*	Referências de cache de latência mais longas
2EH	41H	*LLC Misses*	Falha de cache de latência mais longa
C4H	00H	*Branch Instruction Retired*	Instruções de desvio em retirada
C5H	00H	*Branch Misses Retired*	Instruções de desvio má previstas em retirada

Tabela 2: Exemplos de eventos não-arquiteturais (arquitetura Skylake).

Número do Evento	**Valor UMASK**	**Mnemônico da máscara do evento**	**Descrição**
C4H	00H	BR_INST_RETIRED.ALL_BRANCHES	Instruções de desvio retiradas.
A3H	04H	CYCLE_ACTIVITY.STALLS_TOTAL	Execuções totais de *stalls*.
24H	D8H	L2_RQSTS.DEMAND_DATA_RD_HIT	Demanda de leitura de dados com acertos na cache L2.
49H	01H	DTLB_STORE_MISSES.MISS_CAUSES_A_WALK	Perdas de armazenamento em todos níveis da TLB que causam *page walk*.
B0H	08H	OFFCORE_REQUESTS.ALL_DATA_RD	Requisições de leitura de dados enviadas para fora do núcleo.

tecnologia *hyperthreading* seja desativada, é possível utilizar oito contadores de desempenho por núcleo físico. Embora seja restrita a quantidade de contadores de desempenho disponíveis nos núcleos do processador, é possível monitorar maior quantidade de eventos do que a quantidade de contadores disponíveis. No entanto, essa atividade gera um custo extra ao sistema, pois ocorre a multiplexação da configuração dos eventos em contadores.

Além dos contadores de desempenho presentes nos núcleos do processador, existem contadores de desempenho fora dos núcleos. Esses são capazes de monitorar eventos relacionados às memórias cache e controladora de memória. Eventos, não-arquiteturais e arquiteturais, compõe, respectivamente, a primeira e a segunda classe de capacidades de monitoramento de desempenho.

2.1.1.1.2 Modos de utilização de PMU

O uso de contadores de desempenho de *hardware* normalmente é realizado por usuários através de ferramentas. Ferramentas como perf [29] e OProfile [30] proveem

abstrações para o acesso aos dados de contadores de desempenho e disponibilizam diferentes modos de utilização de contadores de desempenho de *hardware*. Os principais modos de utilização são:

Counting: Esse modo realiza a contagem de ocorrências do evento selecionado. Possui a capacidade de distinguir se os eventos são decorrentes da atividade do sistema operacional (modo privilegiado) ou das aplicações em execução (modo usuário). Porém, não indica as instruções ou funções geradoras dos eventos observados [9][25]. Quando a quantidade de eventos observados é superior à quantidade de contadores de desempenho disponíveis, a ferramenta realiza a multiplexação dos eventos nos contadores. Logo, o tempo de utilização do contador é dividido entre os eventos, com isso a ocorrência total dos eventos não pode ser observada.

Sampling: Possibilita ao usuário analisar eventos gerados pelo código-fonte de alguma aplicação. Esse recurso é possível porque, além dos eventos, o conteúdo do ponteiro de instrução (IP, *Instruction Pointer*) também é coletado. Uma interrupção deve ser gerada no momento em que ocorre *overflow* no registrador contador. Na ocorrência da interrupção, a ferramenta possui um gatilho que é disparado. Então, o conteúdo do registrador IP e de outros registradores são copiados. Após a coleta dos dados, os endereços de instruções da aplicação são combinados com as informações do ponteiro de instrução com o objetivo de determinar a quantidade de vezes que cada instrução foi amostrada [25]. O *sampling* é uma abordagem estatística. Ferramentas permitem que o período de coleta do *sampling* seja uma quantidade de ocorrências de evento ou uma média de coletas por segundo.

Ainda há duas classificações que podem ser feitas quanto ao modo *sampling*:

Time-based sampling: Quando a latência de execução das instruções influencia na frequência de amostragem. Instruções com longa latência terão maior probabilidade de serem amostradas. Ocorre, por exemplo, quando o modo *sampling* é configurado com o evento *cycles per instruction*.

Frequency-based sampling: A execução das instruções gera pesos de amostragem homogêneos. Ocorre, por exemplo, quando o modo *sampling* é configurado com o evento *instruction retired*.

O modo *sampling* descrito pode ser chamado de *interrupt-based event sampling* pois faz uso de interrupções [31]. Porém, também existe o modo *sampling* realizado através de *polling* [31]. Na literatura o modo *interrupt-based event sampling* é tradicionalmente referenciado simplesmente pelo termo *sampling* [12] [26] [25].

Precise Event-Based Sampling, PEBS: Esse modo difere do *sampling* por possuir suporte do processador. Tem o objetivo de eliminar imprecisões na localização do

código gerador de eventos observados. No entanto, essa funcionalidade é implementada no processador para um subconjunto de eventos não-arquiteturais [25].

Last Branch Record, LBR: Assim como o PEBS, esse modo também necessita do suporte do processador. Coleta os últimos desvios ocorridos, instruções geradoras de desvio e respectivos endereços de instruções-alvo [25].

Eventos não-arquiteturais e arquiteturais suportam os modos de uso *counting* e *interrupt-based event sampling*, a segunda classe tem um conjunto menor de eventos [1].

Estudos que analisam os resultados do modo *sampling* mostram que o IP copiado corresponde à instrução que está no topo da fila de instruções ciclos depois de ter ocorrido a geração do *overflow* [12]. Há relatos de 6 e 30 ciclos de atraso, podendo variar de acordo com o processador. Logo, a instrução indicada pelo IP não é a geradora do *overflow*, e sim uma instrução executada alguns ciclos depois. Essa característica é chamada de **deslize** e não ocorre quando existe o suporte ao modo PEBS [12].

O efeito deslize não é importante para *time-based sampling*, apenas distorce o período de amostragem. Porém, tem importância para *frequency-based sampling*. Instruções de longa latência são desproporcionalmente mais amostradas que as demais, pois essas ficam no topo da fila de instruções por períodos mais longos. Então, o efeito de deslize é interrompido nessas instruções. Esse efeito é chamado de **agregação**, que, por consequência, acarreta no efeito chamado de sombra. O efeito de **sombra** ocorre quando as instruções em sequência de instruções de longa latência possuem pouca amostragem [12].

Outro problema relatado na literatura é a possibilidade das amostragens serem realizadas muito próximas de certas instruções do código-fonte da aplicação. Em trechos com laços de repetição, por exemplo, ocorre acúmulo de ocorrências de *overflow* causadas por determinadas instruções. Esse efeito é indesejado e é chamado de **sincronização**. Uma forma de evitar esse tipo de ocorrência é utilizando períodos de amostragem primos ou aleatórios. A utilização desses tipos de períodos possibilita uma distribuição mais homogênea das amostras, pois haverá maior dispersão da ocorrência de *overflow* entre as instruções do código-fonte da aplicação sob análise. No entanto, a opção de utilizar períodos aleatórios não é encontrada em todas as ferramentas. O modo PEBS é inflexível ao lidar com a sincronização da amostragem com o código da aplicação [26] [12].

2.1.1.2 Camada de sistema operacional

Na camada de sistema operacional ocorre o gerenciamento de recursos do sistema. Como um meio de facilitar essa atividade, contadores de desempenho são implementados em sistemas operacionais. Esses contadores são capazes de fornecer dados presentes nas estruturas que controlam as atividades de gerenciamento, ou mensurar essas atividades em intervalos de tempo. Em sistemas operacionais Linux, o acesso aos contadores de desempenho é disponibilizado através de arquivos virtuais no diretório /proc. Em sistemas

operacionais Windows, o acesso padrão é por meio da ferramenta *Performance Monitor*, também conhecida por *perfmon*.

As capacidades de contadores de desempenho em disponibilizar métricas dependem da plataforma computacional utilizada. Contadores de desempenho de *hardware* possuem a capacidade de mensurar ocorrências de eventos relacionados à arquitetura e conjunto de instruções disponíveis pelo processador utilizado. De forma análoga, a nomenclatura dos contadores não segue um padrão entre diferentes sistemas operacionais. As Tabelas 3 e 4 apresentam métricas de desempenho importantes oferecidas por contadores de desempenho implementados pelo sistema operacional Windows e Linux. É possível visualizar a correspondência entre algumas métricas e a ausência de outras. Porém, nem sempre há uma correspondência entre contadores registrados por sistemas operacionais diferentes.

Conforme aponta a Tabela 3, a métrica que indica a percentagem média de utilização de CPU no Windows é fornecida pelo contador *% Processor Time*. No Linux não existe uma métrica correspondente que forneça exatamente o mesmo dado. Porém, quando os valores fornecidos pelas métricas dos contadores *%user*, *%nice* e *%system* são somados, é possível obter um dado com o mesmo significado que a métrica do contador *% Processor Time* disponiliza. Na Tabela 4, por exemplo, o contador *Disk Bytes/sec* do Windows trabalha com a unidade de medida byte, as métricas dos contadores *rKB/s* e *wKB/s* do Linux estão expressas em kilobyte. Logo, conforme apresenta a Tabela 4, é necessário converter o múltiplo da unidade de medida para obter a relação entre as métricas.

Os contadores de desempenho podem ser classificados de acordo com as métricas que representam. Segundo HP [16], as principais classificações são:

- *Instantaneous counters*: Mostra um valor numérico simples da medição mais recente. O comprimento atual da fila de processos esperando por atendimento do processador (runq-sz) é um exemplo de métrica para essa categoria de contador.

- *Interval counters*: Mostra uma taxa de atividade para um dado intervalo de tempo. A percentagem de processamento em modo *kernel* (*%system*) é um exemplo de métrica para essa categoria de contador.

- *Interval difference counter*: Mostra a atividade ocorrida ao longo de um intervalo de tempo. São necessárias duas coletas para o cálculo da atividade. Os valores são obtidos pela divisão do acréscimo de atividade entre as coletas pelo tempo decorrido. O resultado será uma quantidade de ocorrências por segundo para determinada atividade. Pode ser referenciado como uma média, pois a intensidade de ocorrências do evento monitorado pode variar ao longo do intervalo entre coletas. Como por exemplo, interrupções ocorridas por segundo (*intrs/s*).

- *Elapsed time counters*: Coletado em um intervalo e não pode ser resumido. Como por exemplo, o tempo de atividade do sistema (*uptime*). O instante inicial sempre é

Tabela 3: Exemplos de métricas de CPU da camada de sistema operacional.

Windows	**Linux**	**Descrição**
Performance monitor	**sar**	
% Processor Time	%user + %nice + %system	Percentagem média de utilização de CPU.
% Privilege Time	%system	Percentagem média de utilização de CPU em modo kernel.
% Interrupt Time	%irq + %soft	Percentagem média de utilização de CPU recebendo e servindo interrupções.
Context Switches/sec	cswch/s	Taxa média de trocas de contexto por segundo.
Processor Queue Length	runq-sz	Número de processos enfileirados em estado pronto.
% DPC Time	-	Percentagem média de utilização de CPU recebendo e servindo interrupções DPCs.
% User Time	%user	Percentagem média de utilização de CPU em modo usuário.
-	%usr	Percentagem média de utilização de CPU em modo usuário. Não inclui tempo gasto em processadores virtuais.
-	%nice	Percentagem média de utilização de CPU em modo usuário com prioridade *nice* positiva.
-	%steal	Percentagem média de CPU virtual involuntariamente esperando para ser atendida pelo *Hypervisor*.
-	%irq	Percentagem média de utilização de CPU servindo interrupções de *hardware*
-	%soft	Percentagem média de utilização de CPU servindo interrupções de *software*
-	%iowait	Percentagem média de CPU ociosa esperando por requisições pendentes de I/O de disco.
-	%guest	Percentagem média de utilização de CPU executando CPU virtual.
% Idle Time	%idle	Percentagem média de CPU ociosa.
DPCs Queued/sec	-	Enfileiramento médio de DPCs por segundo.
Interrupts/sec	intr/s	Taxa média de interrupções por segundo.

utilizado no cálculo da métrica.

- *Averaging counters*: Provê um valor médio de uma determinada atividade ao longo de um intervalo de tempo. Como por exemplo, média de tempo de resposta para requisições de disco (*Await*).

2.1.2 Nível de aplicação

Assim como os sistemas operacionais, aplicações também podem implementar contadores de desempenho. Os contadores disponibilizados pelas aplicações variam de acordo com o respectivo serviço oferecido. Contadores comuns de serem implementadas estão relacionados a métricas de eficiência, como por exemplo, tempo de resposta e taxas de

Tabela 4: Exemplos de métricas de I/O da camada de sistema operacional.

Windows	**Linux**		**Descrição**
Performance monitor	**iostat**	**df**	
% Idle Time	-	-	Percentagem de tempo que o disco permanece osioso.
(Disk Bytes/sec) / 1024	(rKB/s) + (wKB/s)	-	Número de Kilobytes escrito/lido por segundo.
Disk Transfers/sec	(r/s) + (w/s)	-	Número de requisições de disco completadas por segundo.
Split IO/Sec	-	-	Número de requisições por segundo que foram divididas em múltiplas requisições.
Free Megabytes	-	Available	Megabytes disponíveis para uso na unidade de armazenamento.
Avg. Disk sec/Transfer	Await	-	Tempo médio para completar uma requisição (tempo de fila + tempo de serviço).
Avg. Disk Queue Length	avgqu-sz	-	Tamanho médio da fila de requisições esperando pelo disco.

transferências. A classificação de contadores de desempenho segundo a obtenção das métricas, descrita na Seção 2.1.1.2, também se aplica aos contadores implementados em aplicações.

Plataformas de tempo de execução, como J2EE [32] e .NET [33] disponibilizam contadores de desempenho e também viabilizam a implementação nas aplicações. A tecnologia Java Management Extension (JMX) [34] possibilita a instrumentação de aplicações e do ambiente de tempo de execução Java. A instrumentação é acessível através da interfaces JMX managed bean (MBean). Interfaces MBean são registradas no servidor MBean da plataforma. A plataforma .NET, através de sua linguagem comum de tempo de execução (CLR), expõe seus próprios contadores de desempenho na ferramenta nativa de monitoramento do Windows [16].

O servidor *Apache HTTP server* implementa o módulo *mod_status* para prover métricas de contadores de desempenho [35]. Com esse módulo ativo, o acesso às métricas de desempenho é realizado através de requisições HTTP. O servidor *Microsoft Internet Information Services* (IIS) disponibiliza o acesso aos contadores através da biblioteca *Microsoft Windows performance data helper* (pdh.dll) [36]. Essa é a plataforma geral de monitoramento do Windows. Logo, os contadores de desempenho podem ser acessados pela ferramenta nativa de monitoramento do Windows ou através de outras que utilizem a biblioteca pdh.dll [16].

O RDBMS da Oracle possui a ferramenta SQL Trace [37]. Essa fornece estatísticas

de desempenho para cláusulas SQL individuais. Também existem tabelas de estatísticas com dados de monitoramento relevantes, estatísticas de sessão, V$SESSTAT; estatísticas de sistema, V$SYSSTAT; V$LATCH; V$BUFFER_POOL_STATISTICS. Essas tabelas, além de estarem a disposição do gerente, também são utilizadas pelo sistema *Oracle SQL statement optimizer*. O *Microsoft SQL server* utiliza dos mesmos meios que o IIS para disponibilizar o acesso aos contadores de desempenho. Quase todos os componentes desse sistema possuem contadores de desempenho que expõem suas atividades [16].

2.2 Cenários e casos de uso de contadores de desempenho

Contadores de desempenho são utilizados para traçar a demanda por serviços e recursos do sistema. Séries temporais de dados históricos de monitoramento podem ser obtidas para intervalos de interesse e mantidas desde a implantação de sistemas. Através desses dados é possível verificar a diminuição ou crescimento na demanda a recursos. Em alguns casos, é possível identificar padrões de uso e acesso a serviços.

A identificação de períodos de escassez de recursos é importante para o planejamento de alocação de recursos adicionais com o objetivo de manter a disponibilidade do sistema, latência dentro de padrões aceitáveis [38] ou níveis de acordo de serviço [39]. Outros requisitos de desempenho também são possíveis de serem alcançados. De forma análoga, o reconhecimento de períodos de baixa atividade possibilita que políticas sejam adotadas com o intuito de poupar recursos, sendo possível reduzir gastos com energia elétrica ou com a contratação de recursos sob demanda [40].

O desenvolvimento de aplicações também pode ser beneficiado pelo uso de contadores de desempenho. Ao analisar os recursos computacionais consumidos por aplicações em desenvolvimento, é possível ajustar o código-fonte da aplicação para regular o consumo de determinado recurso ou fazê-lo da maneira mais adequada. Compiladores, por exemplo, podem gerar o código executável mais adequado para as características da carga de trabalho prevista, levando em consideração a plataforma de execução *profile-directed feedback* [41].

Com objetivo de explorar esses casos de uso para contadores de desempenho, as próximas seções tratam, respectivamente, de: gerenciamento de serviços; gerenciamento de recursos; e otimização de código-fonte. A atividade de alterar o sistema com o objetivo de aumentar a escalabilidade é chamada de *performance tuning*.

2.2.1 Gerenciamento de serviços

O gerenciamento de serviços preocupa-se em manter a qualidade dos serviços prestados segundo requisitos de desempenho pré-estabelecidos. Para isso, desenvolve critérios, modelos ou predições para requisitar a alocação ou desalocação de recursos computacionais. Medidas podem ser tomadas antes da implantação do serviço ou após a implantação,

ou seja, em tempo de execução.

O processo de implantação de sistemas normalmente passa por testes de desempenho *off-line*. Sistemas são submetidos a cargas sintéticas de trabalho simulando diferentes quantidades de usuários para verificar a escalabilidade. As métricas mais importantes verificadas são o tempo de resposta e a vazão [2]. Métricas relacionadas ao sistema operacional e ao *hardware* também são analisadas para: descobrir as correlações com o aumento do tempo de resposta e a queda da vazão; e identificar gargalos de desempenho. Metodologias de análise podem ser focadas na identificação de assinaturas de desempenho [42]. Essas auxiliam na detecção de anomalias de desempenho em novas versões de aplicações a serem colocadas em produção. Grechanik et al. [43] propõem um sistema automático de aprendizado de modelo comportamental de aplicações. Esse modelo é obtido durante o teste de desempenho da aplicação e serve para auxiliar em estratégias de provisionamento de recursos.

A detecção de situações indesejadas de desempenho pode ser realizada *on-line*, com o sistema em produção. Como por exemplo, utilizando limiares de desempenho como gatilhos [40]. Esses limiares podem ser obtidos pela correlação de métricas de utilização de recursos com as métricas relacionadas aos requisitos de desempenho. Em [38] e [44] são apresentados estudos que quantificam o tempo de tolerância de usuários em aplicações Web. Limiares de desempenho, assinaturas de desempenho [42] e modelos comportamentais de aplicações [43], ajudam a detectar situações críticas de utilização de recursos e anomalias de desempenho. E com isso, tomadas de decisão com foco em garantir a disponibilidade do sistema podem ser realizadas em tempo real. Com técnicas de predição de desempenho as decisões podem ser tomadas antes mesmo da manisfestação de situações indesejadas.

Segundo Vazquez et al. [40], a predição de demanda de recursos do sistema é necessária, pois as aplicações tornam-se capazes de antecipar o escalonamento e aliviar parcialmente ou totalmente a sobrecarga dessa atividade. Em [45] é apresentado um estudo de predição de desempenho baseado em rede de filas para sistemas Web. Esse estudo toma como base a identificação de tamanhos limites para as filas.

2.2.2 Gerenciamento de recursos

O gerenciamento de recursos preocupa-se com a disponibilidade, alocação e os requisitos de desempenho dos recursos computacionais demandados pelos serviços. O provisionamento dos recursos computacionais pode ser realizado por terceiros, como por exemplo, através de serviços oferecidos por nuvens computacionais. Nesse caso, o gerenciamento dos recursos computacionais é responsabilidade do provedor.

Em ambientes virtualizados, como os presentes em nuvens computacionais, é necessário conhecer a carga demandada pelas máquinas virtuais que compartilham um mesmo servidor físico. Também é preciso conhecer a capacidade do servidor físico e

de sistemas de armazenamento secundário em hospedar as máquinas virtuais respeitando os requisitos de desempenho. A simples alocação de recursos não garante que os requisitos de desempenho sejam alcançados. Há necessidade de um processo de escolha da máquina física que irá hospedar a máquina virtual adicional ou com recursos estendidos, com base na carga de trabalho das máquinas virtuais já hospedadas e na sua capacidade em atendê-las. A máquina virtual precisa ter acesso aos recursos necessários sem impactar as demais máquinas virtuais que compartilham o mesmo servidor físico.

O processo de escolha da máquina física que irá hospedar a máquina virtual é chamado de *placement*. O processo de *placement* pode ser realizado de modo estático ou dinâmico. No modo estático, o histórico de utilização de recursos das máquinas virtuais é utilizado no mapeamento das máquinas virtuais para as máquinas hospedeiras. O mapeamento pode ser refeito, porém, possui baixa periodicidade e as realocações são realizadas *off-line*. No modo dinâmico, o mapeamento é refeito conforme são observadas mudanças relevantes nas tendências de utilização de recursos. O modo dinâmico é auxiliado por técnicas de *live migration*, que permitem uma máquina virtual em execução trocar de servidor hospedeiro sem ter suas atividades interrompidas [40].

É possível realizar um escalonamento horizontal ou um escalonamento vertical ao detectar uma condição que indique a necessidade de alocar recursos adicionais. No escalonamento horizontal os recursos adicionados são máquinas extras que compartilham a carga de trabalho demandada. Técnicas de balanceamento de carga são utilizadas normalmente para distribuir a carga entre as máquinas disponíveis. O escalonamento vertical permite que os recursos de uma máquina sejam ampliados ou reduzidos, como por exemplo: a quantidade de núcleos de processamento, memória principal, volumes de armazenamento. O escalonamento horizontal e vertical também podem ser utilizados em conjunto [46] [40].

Em ambientes de nuvem computacional nem todas as cargas de trabalho são previsíveis. Especialmente aquelas dependentes da interação com os usuários. Provedores de IaaS e PaaS não possuem conhecimento do tipo de aplicação executando sobre esses recursos, nem mesmo possuem acesso aos dados provenientes da instrumentação do código-fonte dessas aplicações. Logo, indícios da sequencialidade de operações que um usuário poderá realizar numa aplicação de comércio eletrônico, por exemplo, não estão disponíveis. Com isso, torna mais difícil a predição de cargas de trabalho. Mesmo aplicações que não interagem com usuários - como exemplo, aplicações de mineração de dados - podem não apresentar padrões de carga de trabalho previsíveis [39].

Com objetivo de lidar com cargas de trabalho com essas caraterísticas, Patel et al. [39] relatam que provedores de nuvem mantêm perfis de execução de tarefas para todos os trabalhos submetidos à nuvem. Os autores supõem que o conjunto de perfis de execução de tarefas coletados durante anos é representativo. Logo, esses perfis podem ser generalizados para padrões de carga de trabalho de todas as tarefas que serão submetidas à nuvem.

Com base nessa hipótese, é possível criar modelos de cargas de trabalho baseado no conjunto de perfis históricos. E com base nos modelos criados estimar os padrões de carga de trabalho das tarefas atuais. Baseados nessas suposições, Patel et al. [39] utilizaram um conjunto de dados do Google [47] como base para criar modelos de padrões de carga de trabalho. Inicialmente os perfis são agrupados em subconjuntos e, em seguida, são criados modelos de padrões de carga de trabalho. Em sequência, o algoritmo desenvolvido pelos autores é capaz de estimar as demandas de novas tarefas com base no modelo criado.

2.2.3 Otimização de código-fonte

Dados extraídos de contadores de desempenho auxiliam no aprimoramento de desempenho de aplicações. As métricas monitoradas servem como indícios de como os recursos estão sendo utilizados. Essa característica é importante, pois processadores possuem diferentes implementações. Com isso, o aproveitamento eficiente de recursos é sensível a lógica utilizada na implementação de aplicações. Logo, a lógica utilizada na aplicação necessita estar alinhada com a lógica de processamento e os recursos disponíveis. Os aprimoramentos de desempenho em aplicações utilizando dados extraídos de contadores de desempenho podem ser realizados de forma manual pelo programador e também através de compiladores *profile-directed feedback* [9].

O programador pode realizar *profiling* do sistema durante a execução de aplicações. É registrado o comportamento de utilização de recursos demandado ao sistema para estudo *off-line*. Com base nas métricas de desempenho observadas, é possível guiar o aperfeiçoamento de aplicações. Uma análise mais refinada pode ser obtida ao realizar *tracing*. Com a instrumentação do código-fonte da aplicação, é possível realizar a coleta de eventos gerados no sistema por componentes da aplicação. Bibliotecas e código-fonte extra são necessários. Uma alternativa à instrumentação é utilizar ferramentas para a realização de *sampling*. Essa técnica não necessita da inserção de código-fonte na aplicação. Porém, a instrumentação possibilita análises mais detalhadas. Os dados podem ser coletados com auxílio de ferramentas com os sistemas em produção [2].

Em [9] são ilustrados os benefícios de utilização de contadores de desempenho de *hardware*. O exemplo de Sprunt [9] destaca como foi possível melhorar o acesso a memória de um processador para uma dada aplicação. Os processadores avaliados foram o Pentium e o Pentium Pro. Ambos os processadores possuem 8 Kbytes de memória cache de dados L1. O Pentium Pro possui adicionais 256 Kbytes de memória cache L2 unificada. Era esperado que o processador Pentium Pro tivesse melhor desempenho por ter um projeto mais avançado e recursos adicionais de memória cache L2. No entanto, ao executar um algoritmo de números primos em ambos os processadores, o processador Pentium Pro obteve resultado inferior ao resultado de seu antecessor. Esse fato ocorreu pois a política de alocação de cache dos processadores são diferentes. O Pentium Pro aloca uma nova linha de cache para cada perda de armazenamento na cache (*store-*

miss), seu antecessor não. Logo, na ocorrência de um *store-miss*, ocorre no Pentium uma transação no barramento entre processador e a memória principal. No Pentium Pro ocorre quatro transações no barramento (8 bits de largura) entre a memória principal e a memória cache (linha de 32 bits), também ocorrendo a atualização na memória cache (linha suja). Em uma nova atualização de linha de cache o Pentium Pro necessitará realizar antes a atualização do conteúdo da memória com o conteúdo presente na linha de cache suja (*write-back*). A utilização de contadores de desempenho de *hardware* tornou esse problema aparente. Com isso foi possível alterar o algoritmo para minimizar as ocorrências de *white-back* no Pentium Pro.

O processo de compilação de aplicações, quando prevê otimizações, é feito com base em heurísticas ou pode ser guiado pelo perfil de execução da aplicação. Esse último método segue um modelo de dupla-compilação, podendo ser chamado de *profile-guided optimization* (PGO ou POGO), *profile-directed feedback* (PDF) ou *feedback-directed optimization* (FDO) [41]. A metodologia de compilação FDO é baseada em instrumentação ou em *sampling*. E necessita que a carga de trabalho gerada na aplicação seja realista para gerar um conjunto de dados representativo para o processo de compilação. Ainda existe o processo de compilação *Just-in-time*, esse possibilita que o perfil de execução da aplicação seja utilizado para a recompilação dinâmica, *on-line*, de parte do código-fonte. O código-fonte é dinâmicamente recompilado, por partes, conforme surgem mudandas no perfil de execução da aplicação [41] [12]. Em [12] é relatado por Chen et al. um ganho de desempenho de 4.106% para FDO baseado em *sampling* e 5.009% para FDO baseado em instrumentação.

2.3 Avaliação de sistemas e diagnóstico de anomalias

Os valores obtidos pela leitura de contadores de desempenho são utilizados como um indicativo do que está ocorrendo no sistema. Esses valores, quando descrevem o estado de uma operação do sistema são chamados de métricas/medições de desempenho. Métricas de desempenho possibilitam detectar a ocorrência de situações indesejadas ou até mesmo prenunciá-las. A interpretação das métricas é dependente do contexto, porém a ocorrência de anomalias ou a ascendência de métricas a certos limiares possibilita o início de uma investigação de possíveis problemas que causam esses fenômenos [2]. Com objetivo de introduzir essa atividade de investigação, esta seção apresenta anomalias e gargalos de desempenho, os indícios que métricas de desempenho podem fornecer com relação ao estado e a saúde do sistema. O conteúdo aqui apresentado não é exaustivo. Os trabalhos [2] e [16] possuem maior abrangência sobre os assuntos apresentados.

2.3.1 Anomalias e gargalos de desempenho

Anomalias de desempenho são comportamentos inesperados do sistema que podem ser observados através do monitoramento de métricas de desempenho. Com frequência estão relacionados à manifestação de gargalos que, por sua vez, são recursos ou componentes de aplicação que limitam o desempenho do sistema. Gargalos se pronunciam quando ocorre a saturação de utilização ou contenção de acesso a recursos ou componentes de aplicação. Esse fenômeno pode ocorrer em um único elemento do sistema, simultaneamente em mais de um elemento ou também em alternância entre elementos. A identificação e eliminação desses fenômenos possibilita resolver problemas de desempenho e de disponibilidade de sistemas. A Figura 4 apresenta as relações causa-efeito de gargalos de desempenho [2].

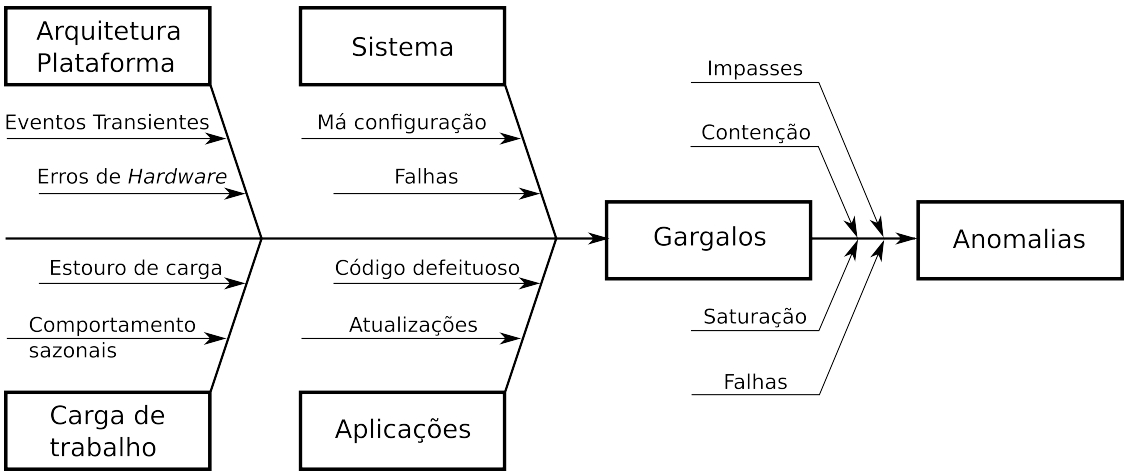


Figura 4: Relações causa-efeito de gargalos de desempenho (adaptado de [2]).

Na Figura 4, os quatro retângulos mais à esquerda representam as principais categorias de causa-raiz de gargalos e anomalias de desempenho. As setas horizontais rotuladas são exemplos de causas primárias. Os dois retângulos mais à direita são os principais efeitos das causas primárias. E as setas inclinadas são exemplos de causas secundárias. As setas horizontais mais grossas explicam, da esquerda para a direita, os efeitos decorrentes das causas primárias e secundárias [2]. Comportamentos sazonais, atualizações, estouros de carga e má configuração do sistema são fatores que contribuem para a saturação do sistema. E, por consequência, podem induzir a ocorrência de falhas. Erros de *hardware* e códigos defeituosos além de contribuírem para a saturação do sistema, também podem ser as fontes geradoras de contenção de recursos ou de impasses [48].

A ocorrência de algumas anomalias está relacionada a cenários específicos, como: intensidade e tipo de cargas de trabalho, configurações do sistema, natureza da infraestrutura computacional, entre outros [2]. A evidência de anomalias pode ser baseada em requisitos de desempenho. O não cumprimento de requisitos de desempenho, como por exemplo, uma certa quantidade mínima de transações por segundo, pode ser considerada

uma anomalia.

Essas anomalias podem ser identificadas através da visualização de gráficos de dados extraídos de contadores de desempenho; e podem ser classificadas de acordo com a disposição dos dados no gráfico e o contexto. Essa *análise manual* exige experiência e pode ser subjetiva quando existe pequenas variações nos valores dos dados sob análise. Por exemplo, o gráfico ilustrado na Figura 5 apresenta uma anomalia coletiva, ou seja, uma região do gráfico mostra pontos que desviam, entre 300 e 400 segundos, um grupos de pontos que se desviam do comportamento de desempenho esperado para a métrica monitorada. Neste exemplo, por alguma razão, a taxa de transferência apresenta uma queda momentânea.

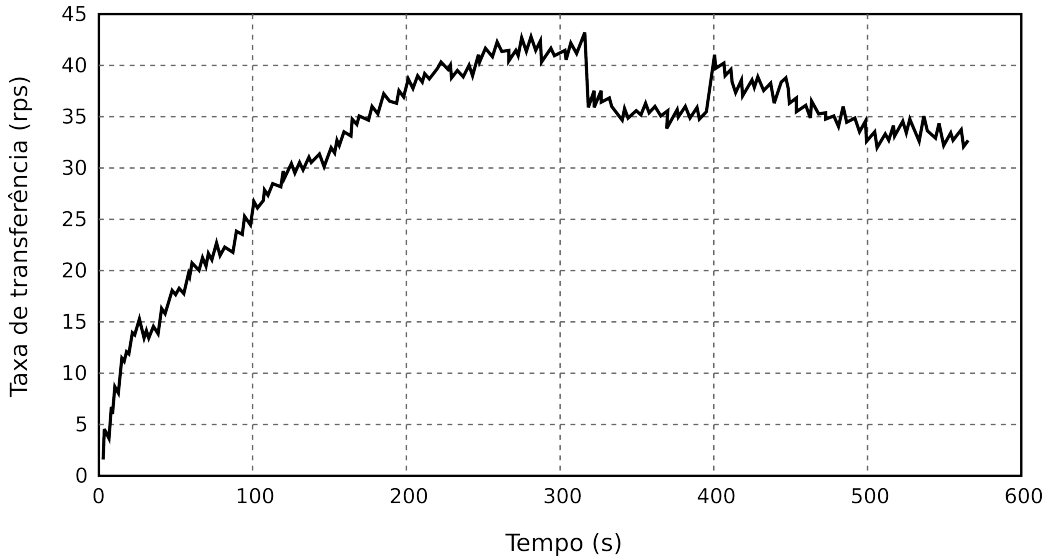


Figura 5: Anomalia de desempenho (adaptado de [2]).

A *análise automática* é uma alternativa à análise manual. A análise automática, segundo Ibidunmoye et al. [2], é realizada por sistemas chamados de sistemas de detecção de anomalias de desempenho e identificação de gargalos. Esses sistemas possuem como objetivo: detectar comportamento de desempenho anômalo; identificar a causa raiz de uma anomalia de desempenho; e indicar o recurso de sistema ou componente responsável por uma violação de objetivo de desempenho. A concretização desses objetivos é alcançada através da análise de dados extraídos por contadores de desempenho submetidos a diferentes metodologias de análise. Dentre essas, segundo os autores, destacam-se: abordagens estatísticas; algoritmos de aprendizado de máquina; e métodos como Transformadas de Fourier, Filtragem Adaptativa e Média de Janela Estendida.

2.3.2 Interpretação de métricas de desempenho

Métricas extraídas de contadores de desempenho revelam o comportamento de utilização de recursos do sistema. Através da interpretação de métricas de desempenho é possível verificar o estado de utilização de recursos que compõem o sistema, como por

exemplo: processador, memória e serviços. Situações de utilização de recursos prejudiciais à saúde do sistema podem ser identificadas pela interpretação de métricas. No entanto, essa atividade exige experiência. Em sequência são apresentados estados de utilização de recursos importantes relatados em [16].

2.3.2.1 Processador

O tamanho da fila de processador é utilizado como um indicativo da capacidade do processador em atender, imediatamente, a carga de trabalho. Em multiprocessadores essa métrica deve ser analisada dividindo o seu valor pelo número de núcleos físicos de processamento [16]. Segundo [16], é necessário observar essa métrica com cautela quando seu valor ultrapassar, consistentemente, a quantidade de 2 processos em espera por núcleo físico. Sempre que o tamanho da fila de processador for maior que 5, indica que o processador não está conseguindo lidar prontamente com toda a carga de trabalho. Se o tamanho da fila for maior que 10, indica que o processador está saturado.

Ainda relacionado ao processador, a utilização total do tempo de processamento por longo período normalmente indica a existência de algum processo em descontrole. Nesse caso, normalmente há a combinação com 3 ou mais processos enfileirados ou bloqueados por prioridade. A atividade em modo usuário ou modo *kernel* pode indicar, respectivamente, se o processo é originário de alguma aplicação executando em nível de usuário ou se é originário do sistema. Nem sempre altas taxas de utilização do processador indicam a presença de um problema que deve ser corrigido. Porém, é preciso ter cautela quando a utilização do processador se mantiver entre 80-90% por longo período. O tempo de processador ocioso abaixo de 10% pode indicar alguma anomalia [16].

Taxas de trocas de contexto altas indicam, com frequência, projeto de aplicações mal desenvolvidos e podem implicar em problemas de escalabilidade. Dispositivos em mau funcionamento podem gerar interrupções em excesso e, consequentemente, contribuir para o processador se tornar um gargalo [16].

A métrica de utilização total de processamento é um indicativo primário de que o processador pode ser o gargalo no sistema. O tamanho da fila de processador é um indicativo secundário, pois é um reflexo do esgotamento da capacidade de processamento. Ao observar limites para as métricas, e relacionar os dados coletados de várias métricas, é possível identificar anomalias, bem como suas causas [16].

2.3.2.2 Memória

Relacionado à memória, é necessário ter cautela quando a leitura de páginas por segundo ultrapassa 50 por disco acessado. O aumento excessivo de leitura de páginas em disco diminui o tempo de resposta. O crescimento de 10% de bytes de páginas de memória virtual relacionadas a um único processo pode indicar vazamento de memória, ou seja, quando o processo aloca memória e não faz a desalocação no momento em que aquela

porção de memória não é mais necessária. Vazamento de memória também pode ocorrer com plataformas de tempo de execução que possuem coletor de lixo ineficiente. A Figura 6 ilustra a ocorrência de vazamento de memória. É possível observar que aproximadamente a cada 50 segundos ocorre um processo de desalocação de memória ineficiente seguido de uma nova alocação. A sequencialidade dessas ocorrências acarreta no esgotamento de memória disponível. Quando a memória principal atinge 80% de utilização indica que sua capacidade é insuficiente. Ao alcançar 90% de utilização pode causar a falha de processos em execução. A escassez de memória está frequentemente relacionada com a sobrecarga da largura de banda do disco. Ou criação de objetos que nunca são derreferenciados no código-fonte de aplicações. Por consequência, taxa de paginação no disco é um indicador de desempenho importante para a memória principal [16].

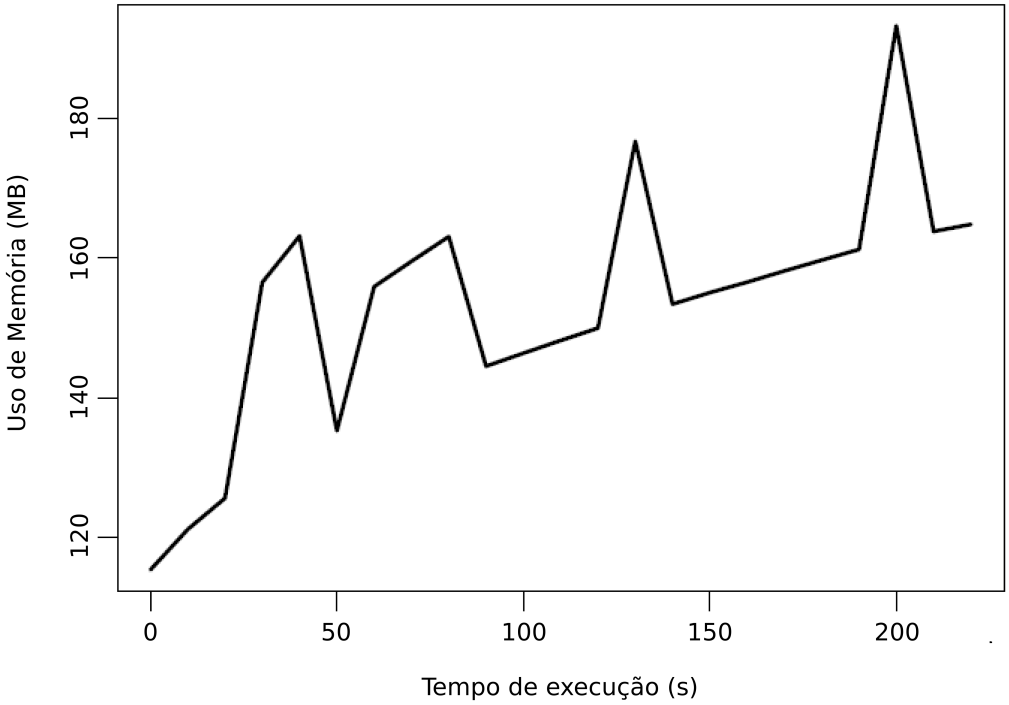


Figura 6: Vazamento de memória (adaptado de [3])

2.3.2.3 Armazenamento

Unidades de disco, além de constituírem o espaço de endereçamento secundário, também possuem *buffer* de memória e processadores capazes de reordenar as requisições ao disco para extrair melhor desempenho do dispositivo. Esses recursos permitem ao disco elevar o seu desempenho em 25-30%. A utilização de 90% do dispositivo deve ser evitada. Acima de 95% de utilização gera ocorrências de erros. Portanto, transferências de disco (bytes lidos ou escritos em disco) por segundo é um indicador primário de que o disco é um gargalo. Porém, o limiar de detecção dessa situação é dependente do dispositivo [16].

2.3.2.4 Comunicação

Contadores de desempenho relacionados à rede podem fornecer indícios relativos à segurança e ao desempenho dos equipamentos físicos em uso. Tentativas de ataques como negação de serviço e intrusão são situações que podem ser detectadas através de mudanças repentinas em métricas extraídas de contadores de desempenho. A ocorrência de picos ou aumentos superiores a 10% no número de conexões TCP e na quantidade de requisições TCP recebidas podem indicar, respectivamente, ataques de negação de serviço e a presença de intrusos no sistema. No aumento da taxa de colisões, não precisando ser superior a 10%, os equipamentos físicos podem estar apresentado problemas, por exemplo, na compressão de *hardware* ou em conectores físicos. Dependendo do contexto, a evidência desses aumentos também pode indicar sobrecarga no sistema [16].

2.3.2.5 Servidores de aplicação

Aplicações específicas, como servidores web e sistemas de gerenciamento de banco de dados implementam os próprios contadores de desempenho [16]. No entanto, o gerenciamento desses serviços também se beneficia de contadores de desempenho provenientes da plataforma em que estão inseridos. Informações de utilização de recursos disponíveis, como processador, memória e disco, auxiliam atividades de *performance tuning*. Desse modo, é possível calibrar o consumo de recursos das aplicações de acordo com o *hardware* disponível, objetivando obter o melhor desempenho possível.

2.3.2.6 Sistemas gerenciadores de banco de dados

Em sistema de banco de dados, o monitoramento individual do tempo de processador para núcleos ou processadores é importante para verificar o balanceamento de carga entre esses recursos. Quando o tamanho da fila de processos exigir atenção e a utilização do processador não estiver alta, pode ser necessário reduzir o número máximo de *threads* de trabalho. Em caso de carga de trabalho elevada é possível reduzir a quantidade de trocas de contexto especificando o processador que cada *thread* irá utilizar. Esse procedimento melhora o desempenho pois diminui o número de vezes que o processador recarrega o conteúdo da memória *cache*. Quando o tempo de utilização do disco supera os 55%, tanto para leitura quanto para escrita, é um indicador de gargalo [16].

2.4 Análises estatísticas

Em estatística, o termo análise multivariada de dados refere-se aos métodos estatísticos capazes de analisar simultaneamente múltiplas medidas sobre cada objeto de investigação [49]. Em contraste com a análise multivariada existem outros tipos de análises que se diferenciam quanto à quantidade de variáveis consideradas: a análise univariada, que considera uma única variável em análises de distribuição; e a análise

bivariada, que considera duas variáveis, como, por exemplo, em análises de correlação e de regressão simples. Muitas das análises multivariadas são extensões de análises univariadas e de análises bivariadas [49]. Nas seções seguintes são apresentadas algumas análises estatísticas.

2.4.1 Coeficiente de correlação linear de Pearson

O coeficiente de correlação linear de Pearson é um tipo de análise bivariada. O coeficiente de Pearson, obtido pela Equação 1, exprime o grau de correlação entre duas variáveis quatitativas, e identifica a direção da correlação. O coeficiente Pearson é representado por r quando o conjunto de dados das variáveis envolvidas na análise representa uma mostra. Quando representado por ρ, o conjunto de dados utilizado representa a população. O grau de correlação, r ou ρ, admite valores entre -1 e 1, inclusive. O sinal do valor do coeficiente indica a direção da correlação, positiva ou negativa. O módudo do valor de coeficiente indica o grau de correlação: quanto mais próximo de 1, mais forte é a correlação; e quanto mais próximo de 0, mais fraca é a correlação [50].

$$r(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left[\sum_{i=1}^{n}(x_i - \bar{x})^2\right]\left[\sum_{i=1}^{n}(y_i - \bar{y})^2\right]}} \tag{1}$$

2.4.2 Regressão múltipla

A técnica de regressão múltipla tem como objetivo estimar o relacionamento entre uma variável dependente e diversas variáveis independentes. Uma variável dependente é uma variável que está sendo prevista ou explicada por um conjunto de variáveis. As variáveis pertencentes ao conjunto de variáveis que prevêem ou explicam uma variável dependente são chamadas de independentes. Uma vez encontrada a equação que estabelece o relacionamento entre a variável dependente e as variáveis independentes é possível predizer o valor da variável dependente baseado em valores conhecidos das variáveis independentes [49]. A equação de regressão é descrita pela Equação (2).

$$Y' = A + B_1X_1 + B_2X_2 + ... + B_nX_n \tag{2}$$

A variável Y' representa o valor predito para a variável dependente Y. Para obter essa predição é necessário: definir um coeficiente linear, representado na equação pela constante A; e definir os coeficientes de regressão, representados na equação pelas constantes B_i. Cada um dos coeficientes de regressão está associado a uma das variáveis independentes, representadas na equação pelas variáveis X_i. O método de regressão múltipla, assim como o método de regressão bivariada, define valores para os coeficientes de regressão (B) de modo que os valores estimados (Y') se aproximem dos valores observados para a variável dependente (Y).

Para isso, uma solução adotada é encontrar os coeficientes de regressão que gerem o menor valor para diferença quadrada entre Y' e Y. Essa solução é chamada de solução dos mínimos quadrados [49]. Nessa solução é minimizada a soma de quadrados dos erros (SSE, do inglês *sum of squared errors*). Considerando n como a quantidade de amostras, SSE é definido pela Equação (3).

$$SSE = \sum_{i=1}^{n} (Y_i - Y_i^{'})^2 \tag{3}$$

Uma forma de avaliar a previsão da variável dependente é através do coeficiente de determinação R^2. Esse coeficiente é encontrado a partir da relação entre a soma de quadrados da regressão (SSE) e a soma total de quadrados (TSS, do inglês *total sum of squares*), conforme a Equação (4). O valor do coeficiente de determinação é um número real e pode variar entre 0 e 1. Um coeficiente de determinação com valor igual a 0 representa que a equação de regressão encontrada não é capaz de prover previsões melhores do que quando comparada com o emprego da média da variável dependente como predição. Se o valor do coeficiente de determinação for 1, indica que a equação de regressão consegue prever perfeitamente valores para a variável dependente. Logo, o coeficiente de determinação R^2 representa o quanto as variáveis independentes conseguem explicar a variância da variável dependente em torno de sua média [49].

$$R^2 = \frac{\sum_{i=1}^{n} (Y_i - Y_i^{'})^2}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2} \tag{4}$$

O valor do coeficiente de determinação representa o efeito combinado de todas as variáveis independentes usadas para prever a variável dependente. No entanto, a capacidade de uma variável independente melhorar o poder de explicação da equação de regressão não está relacionada somente com sua correlação com a variável dependente. A capacidade de explicação também está relacionada com a correlação existente com as demais variáveis independentes já incluídas na equação. Essa associação entre variáveis independentes, medida como a correlação, é chamada de colinearidade. Essa associação é chamada de multicolinearidade quando ocorre entre três ou mais variáveis. A presença de (multi)colinearidade faz com que exista uma variância única explicada por cada variável independente e um percentual de previsão compartilhado [49].

Uma questão importante que deve ser analisada ao empregar a técnica de regressão é a abordagem que será utilizada para especificar o modelo de regressão. Em [49] são descritas três abordagens para a seleção das possíveis variáveis independentes:

- Especificação confirmatória: Nessa abordagem o pesquisador especifica por completo o conjunto de variáveis independentes que devem ser consideradas no método de regressão.

- Métodos de busca sequencial: Abordagens desse tipo trabalham com um conjunto de variáveis. Essas variáveis são acrescentadas e/ou eliminadas do modelo seguindo um critério seletivo passo a passo até que um critério geral seja alcançado. Métodos desse tipo possuem como objetivo maximizar a previsão com o menor número de variáveis independentes. São exemplos desses métodos, estimação *stepwise*, adição *forward* e eliminação *backward*.

- Abordagem combinatória: Essa abordagem prevê que todas as possíveis combinações de variáveis independentes sejam consideradas. O procedimento mais conhecido é a regressão em todos os possíveis subconjuntos.

Algumas considerações importantes podem ser feitas quanto às abordagens descritas. Na abordagem de especificação confirmatória, o pesquisador é o encarregado de assegurar que o conjunto de variáveis selecionadas atinja a melhor previsão. Esse cuidado é dispensado em métodos de busca sequencial, porém, o pesquisador deve ter ciência de que as variáveis excluídas da equação de regressão podem ter correlações quase iguais às correlações que as variáveis selecionadas possuem com a variável dependente. Isso ocorre quando existe multicolinearidade entre as variáveis independentes. Na abordagem combinatória o pesquisador deve lembrar que nenhuma outra questão, além das possíveis combinações possíveis, foi considerada para encontrar o melhor modelo. Por isso, a suposta melhor equação encontrada pode acabar sendo substituída por outra ao considerar questões como a multicolinearidade, interpretação dos resultados e observações atípicas e influentes [49].

2.4.3 Análise de agrupamento

Agrupamento é uma técnica multivariada capaz de dividir uma amostra de dados em conjuntos. Essa divisão é baseada nos atributos descritivos dos dados. Técnicas de agrupamento utilizam estratégias baseadas em medidas de distância para agrupar dados por similaridade. As técnicas de agrupamento buscam obter agrupamentos com elevada homogeneidade interna (similaridade entre elementos de um mesmo grupo maximizada) e elevada heterogeneidade externa (similaridade entre elementos de diferentes grupos minimizada) [49] [51].

O propósito de utilização da análise de agrupamento pode ser para realizar [49]:

- Descrição taxonômica: Classificação de objetos baseada na experiência, servindo como uma análise exploratória ou para fins confirmatórios.

- Simplificação de dados: Concede uma perspectiva simplificada das observações, podendo utilizar-se das agregações para análise posterior.

- Identificação de relação: Meio de revelar relações ou similaridades entre as observações.

Diferentes estratégias podem ser utilizadas para realizar a tarefa de agrupamento, e dividem-se nas seguintes categorias [51]:

- Estratégias hierárquicas: os dados são organizados em grupos hierárquicos. Há duas abordagens: *top-down* (divisiva); e *bottom-up* (aglomerativa). A abordagem *top-down* inicialmente insere todos os dados em um único grupo e em sequência divide o grupo em grupos menores e esses em outros, assim sucessivamente até que cada dado seja isolado em um grupo. A abordagem *bottom-up* inicia com cada dado em um grupo separado e sucessivamente aglomera os grupos em grupos maiores, até que um único grupo possa ser formado. O cluster hierárquico é apresentado com os dados nas folhas de uma hierarquia que representa a sequência de divisões ou aglomerações, dependendo da abordagem. Os métodos AGNES [49] e DIANA [49] são implementações da abordagem aglomerativa e divisiva, respectivamente.

- Estratégias por partição: grupos são formados segundo a disposição dos dados no espaço em que estão inseridos. Para formar os grupos é necessário a realização de partições segundo um critério de particionamento. Objetivo de maximização de similaridade intragrupo. O algoritmo k-médias é um exemplo dessa categoria [49].

- Estratégias baseadas em densidade: dados são escolhidos para formar grupos unitários ou com poucos elementos. Iterativamente são inseridos dados aos grupos por critério de vizinhança até que determinado limiar seja atingido. O algoritmo DBSCAN [49] é um exemplo dessa categoria.

2.5 Descoberta de conhecimento em banco de dados

Descoberta de conhecimento em banco de dados ou KDD (do inglês *Knowledge Discovery in Data Base*) é um processo que permite extrair ou minerar conhecimento de um grande volume de dados. Nesse contexto, dado é definido como um valor documentado; e informação, como um significado atribuído a um conjunto de dados. O conhecimento é descoberto quando um agente é capaz de tomar decisões baseadas em informações de seu domínio [51].

O KDD possui várias utilizações, tanto na indústria [52] como na academia [53]. Esse processo permite abstrair um grande volume de dados em modelos e/ou em estruturas compactas, tornando os dados interpretáveis. O processo de KDD promove a substituição de métodos manuais, tornando a análise de dados mais eficiente, menos custosa e livre da subjetividade de especialistas.

Segundo Fayyad *et al.* [4] o processo de KDD é dividido em etapas. A primeira etapa descrita pelos autores compreende na definição do objetivo do KDD, levantamento de conhecimentos prévios e entendimento da área da aplicação. As demais etapas podem ser sintetizadas, conforme ilustra a Figura 7, nas etapas básicas:

- Seleção: Seleção dos dados de interesse. Subconjunto dos dados disponíveis sobre os quais se deseja descobrir algo.

- Pré-processamento: Preparação dos dados selecionados na etapa de seleção. São tratadas questões como: organização em um repositório único, instâncias repetidas, valores discrepantes, presença de ruído; inconsistência; formatação inadequada; e dados faltantes.

- Transformação: Abrange a normalização e a conversão de dados.

- Mineração de dados: Seleção e execução de métodos de mineração de dados capazes de descobrir padrões nos dados preparados.

- Interpretação/Avaliação: Visualização da representação do modelo de conhecimento extraído para realização da interpretação e validação.

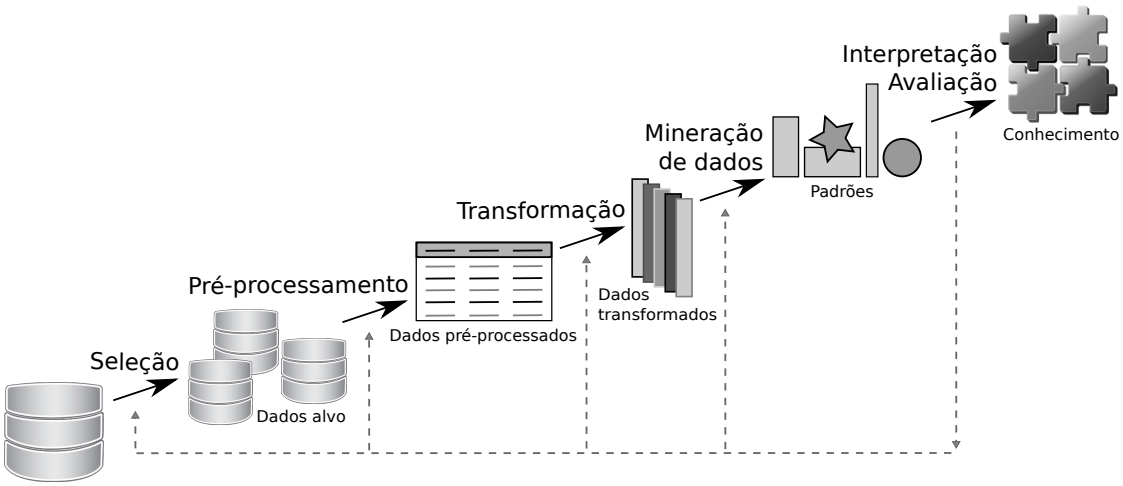


Figura 7: Etapas do processo de KDD (adaptado de [4]).

As setas tracejadas presentes na Figura 7 representam possível retorno da etapa de interpretação/validação para etapa(s) anterior(es) com o objetivo de modificar a estratégia adotada no processo para obtenção de resultados mais satisfatórios ou que complementem algum conhecimento obtido. Decorrente dessa atividade cíclica o processo é dito iterativo, e interativo, pois necessita da interação com o pesquisador.

A etapa de mineração de dados é a etapa que encontra padrões para o processo de KDD. A mineração de dados pode ser realizada através de métodos estatísticos, como os apresentados nas Seções 2.4.2 e 2.4.3, ou através de algoritmos de aprendizado de máquina. Diversas tarefas podem ser realizadas nessa etapa, entretanto, com uma visão abrangente, é possível citar três grandes tarefas que podem assumir mais de uma variação e dar origem a subtarefas [51]:

- Predição: Essa tarefa mapeia, através de um modelo descoberto, cada um dos dados de entrada aos seus respectivos rótulos. Essa análise pode ser dividida em duas subtarefas, análise preditiva categórica (classificação) e análise preditiva numérica (regressão). Os rótulos para a subtarefa de classificação devem assumir valores categóricos pertencentes a um conjunto discreto e finito. Para a subtarefa de regressão os rótulos devem ser numéricos e pertencentes a um conjunto de valores contínuos.

- Agrupamento: Fornece um modelo de agrupamento ou perfis para grupos de dados. Essa tarefa tem como objetivo organizar os dados em grupos através de critérios que consideram a similaridade entre os dados. Para a formação dos grupos os dados são avaliados de forma que os dados mais similares pertençam ao mesmo grupo e que dados com baixa similaridade pertençam a grupos diferentes. A similaridade intra grupo deve ser maximizada enquanto que a similaridade extra grupo seja mínima.

- Associação: Eventos ou transações pertencentes a um determinado contexto são explorados a procura de ocorrências frequentes e simultâneas entre elementos. Ocorre a procura por elementos frequentemente envolvidos nos mesmos eventos ou que apresentam algum tipo de correlação em seus comportamentos em tais eventos.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados da seguinte forma. Na Subseção 3.1 são levantados trabalhos que tratam da utilização de contadores de desempenho. Na Subseção 3.2 são apresentados trabalhos que expõem efeitos indesejados ao utilizar contadores de desempenho. Na Subseção 3.3 são abordados trabalhos que tratam de sobrecargas que afetam aplicações em decorrência do uso de ambientes virtualizados.

3.1 Emprego de contadores de desempenho

Com cada vez mais aplicações hospedadas em nuvens computacionais, torna-se mais importante o provisionamento de recursos computacionais de forma eficiente. Enquanto algumas abordagens levam em conta características de aplicações para identificar quais aplicações devem estar co-alocadas, outras analisam a carga de aplicação em tempo de execução através da leitura de dados de monitores de desempenho.

Em [54], Jin et al. analisam o padrão de acesso à memória cache em aplicativos HPC (do inglês, *high performance computing*) executados em nuvens. Para isso é utilizada uma análise de distância de reutilização [55] melhorada juntamente com um algoritmo de compactação cíclica acelerado para identificar a intensidade de interferência na memória cache da aplicação. Os autores observam que aplicativos com localidade fraca e perfis de conjuntos de trabalho de cache grande pode ser co-localizado com aplicativos amigáveis em cache, a fim de minimizar a interferência entre aplicativos.

Mars et al. [56] apresentam uma metodologia para previsão de degradação de desempenho em ambientes virtualizados. A abordagem é adequada para aplicativos em que a sobrecarga é causada pela contenção de recursos compartilhados no subsistema de memória. A metodologia é dividida em duas etapas. Na primeira etapa é realizada uma caracterização da sensibilidade da aplicação em sofrer diferentes pressões de compartilhamento de memória. Em segunda etapa, a aplicação é avaliada quanto a contenção de recursos de memória, em termos de pressão realizada ao subsistema. A previsão é realizada através de curva de sensibilidade e de pontuações atribuídas às aplicações. Os autores investigaram o isolamento de desempenho de tarefas sensíveis à latência.

As abordagens mencionadas anteriormente dependem de abordagens estáticas baseadas no perfil anterior de aplicativos. Em cenários mais gerais, em que uma caracterização completa de aplicativos é inviável, a interferência de desempenho deve ser detectada no tempo de execução. Nesse sentido, monitoramento em tempo de execução é amplamente adotado como um meio de observar o desempenho dos sistemas [40] [2].

Rameshan et al. [57] melhoraram a alocação de recursos, aprendendo continuamente o comportamento favorável e desfavorável da co-execução. São utilizados métodos de aprendizado de máquina, como classificador binário, para realizar a modelagem de cargas de trabalho que violam condições de desempenho estabelecidas. Essas informações são usadas para prever e evitar decisões de agendamento ineficientes.

A abordagem proposta por Zhang e Figueiredo [58] também auxilia na programação de decisões com base no monitoramento de desempenho de MVs centradas em aplicativos. Os instantâneos de desempenho são obtidos no tempo de execução e os dados coletados são processados para extrair e classificar informações relevantes sobre a utilização de recursos. Sua abordagem de classificação é baseada em algoritmos de seleção em destaque, na Análise de Componentes Principais e no classificador *k-Nearest Neighbor*.

Nossa abordagem visa reduzir a sobrecarga de monitoramento através da seleção de contadores de desempenho. Em [58], os autores propõem uma abordagem semelhante, mas limitam os dados da amostra a um pequeno conjunto de contadores de desempenho escolhidos antecipadamente por um especialista. Nossa abordagem não limita o número de contadores e nenhum conhecimento prévio sobre o conjunto de contadores de desempenho sob monitoramento é necessário.

Em [15], os autores também avaliam a correlação entre os contadores de desempenho, no entanto, eles se concentram em testes de regressão. No trabalho deles, eles aplicam a mesma carga de trabalho em diferentes versões de um aplicativo e detectam alterações de desempenho de uma versão para outra. Os autores declaram que a análise desse procedimento, quando realizado de forma manual, está sujeita a experiência do profissional responsável. Destacam também que a interpretação dos dados coletados por contadores de desempenho está sujeita a subjetividade. Em análise automática proposta pelos autores, não há seleção de contadores considerados relevantes, todos os contadores são submetidos ao processo de análise.

HP [16] discute a relevância de alguns contadores de desempenho para o monitoramento de sistemas. Nesse mesmo trabalho, são citados alguns limiares para valores obtidos por contadores de desempenho que exigem atenção dos administradores de sistemas. No entanto, as informações oferecidas são empíricas, não é apresentado nenhum experimento ou metodologia de estudo científico.

Segundo Patel et al. [39], a predição de cargas de trabalho é uma tarefa indispensável para um gerenciamento de recursos eficiente, e provisionamento excessivo ou insuficiente são indesejáveis. Vazquez et al. [40] afirmam que, com predições corretas, aplicações são

capazes de antecipar o escalonamento e aliviar parcialmente ou totalmente a sobrecarga dessa atividade.

3.2 Efeitos causados pela utilização de contadores de desempenho

Segundo Bitzes e Nowak [25] a utilização de contadores de desempenho de *hardware* gera uma sobrecarga ao sistema. O estudo dos autores é baseado na utilização de contadores de desempenho implementados em PMU de processadores de propósito geral. As sobrecargas observadas variam conforme o modo de utilização dos contadores de desempenho. Os autores também observaram que, dependendo do modo de utilização dos contadores, o tipo de carga de trabalho gerada e a escolha dos eventos monitorados podem influenciar drasticamente nas taxas de sobrecarga gerada ao sistema. Esse efeito toma maiores proporções quando há a multiplexação de eventos nos contadores. A sobrecarga é mais significativa para os modos *interrupt-based sampling* pois uma maior quantidade de dados necessita ser transferida para a ferramenta de monitoramento. As observações de sobrecarga de pior caso, apresentadas pelos autores, não ficam abaixo de 5% de sobrecarga, podendo chegar até em 45%, porém, em geral, não passam de 18%. Sobrecargas elevadas são justificadas pelo aumento de operações de I/O no processo de monitoramento em períodos mais curtos. Em geral, é observado um aumento de sobrecarga conforme maior for a quantidade de núcleos submetidos ao monitoramento [25].

Em [25] é disponibilizado um estudo da acurácia para métodos de realização de *sampling*. Diferentes arquiteturas são comparadas para diferentes tipos de períodos de amostragem. Os autores recomendam a utilização de períodos primos, pois, além de poder apresentar melhores resultados, reduz o risco de ocorrer sincronização com a carga de trabalho submetida ao processador. Embora as ferramentas disponíveis não suportem coletas em períodos aleatórios, esses também agregam melhor acurácia aos períodos primos e não primos.

Em [21], Bara et al. relatam que a ativação de contadores de desempenho de *hardware* acarreta em maior consumo de energia pelo processador. O estudo realizado é baseado em implementação de contadores de desempenho em processador OpenRISC de núcleo OR1200 open-source. Os experimentos dos autores mostraram que a utilização dos contadores de desempenho gerou um consumo extra de energia de 1% a 5%. Esse aumento possui aproximadamente 61% de correlação com as atividades dos contadores. É importante destacar que o processador utilizado nesse último experimento possui 8 contadores de desempenho, cada contador monitorou um único evento durante os experimentos. A realização de multiplexação de eventos para o monitoramento de diferentes eventos por um mesmo contador não foi considerada. A multiplexação, conforme apresentam os experimentos de Bitzes e Nowak [25], é um fator importante para a elevação da sobrecarga gerada pela utilização dos contadores de desempenho. Com isso, é possível considerar a

possibilidade de observação de consumo de energia mais elevado do que apresentado por Bara et al. em [21]. Porém, considerando que uma sobrecarga é gerada pela utilização de contadores de desempenho em processadores de propósito geral, é esperado um maior consumo de energia também para esses processadores.

3.3 Sobrecarga em ambientes virtualizados

A contenção de recursos compartilhados é uma das principais causas de degradação de desempenho em sistemas computacionais. Em infraestruturas virtualizadas, como nuvens computacionais, a contenção é ainda mais perceptível. Nesses ambientes, aplicações sofrem interferência devido a co-alocação dessas em um mesmo servidor hospedeiro [59]. A interferência pode ocorrer em qualquer nível, seja em nível de CPU, memória, *buffer* de I/O ou em nível de outros componentes do sistema. Por isso, para usufruir dos benefícios trazidos pela utilização de ambientes virtualizados de maneira eficiente, deve haver um balanceamento entre desempenho e utilização de recursos para minimizar a sobrecarga gerada pela interferência de desempenho [57].

A degradação de desempenho pode ser causada por padrão específico utilização de recursos demandado pela aplicação. Por exemplo, algumas aplicações são caracterizadas por amplo consumo de memória cache, as quais podem criar gargalo de desempenho no último nível de cache compartilhada (*Shared Last Level Cache* - SLLC) [60]. Aplicações co-agendadas podem induzir interferências no acesso à cache, as quais resultam em maior ocorrência de eventos de *cache miss*. Nessa situação, a quantidade de *cache misses* é excessiva, e por consequência acarreta em degradação de desempenho por ter que buscar dados e instruções em níveis inferiores da hierarquia de memória, os quais possuem menor desempenho quando comparados com o acesso à memória cache.

Máquinas virtuais compartilhando os mesmos núcleos de processamento estão sujeitas a interferência devido ao SSLC. Como investigado em [54], contenções de cache corrompe a proteção inerente e o isolamento promovido pela virtualização. Assim, ocorre interferência direta no desempenho de aplicações independentes executando em um mesmo servidor físico. Esse cenário é normalmente encontrado em ambientes que hospedam aplicações HPC em máquinas virtuais. As características desse tipo de aplicação acarretam em consumo intensivo de memória cache. Alocar essas máquinas virtuais em núcleos distintos pode acarretar em contenções mais intensivas para as demais máquinas virtuais co-alocadas.

Alves et al. [61] demonstram que a interferência de desempenho também pode ser influenciada pela similaridade das cargas de aplicações. Quando aplicações co-alocadas em um mesmo servidor apresentam um alto nível de similaridade de carga demandada para um dado recurso compartilhado, as aplicações competem uniformemente pelo recurso o qual, por sua vez, influência o nível de interferência sofrido pelas aplicações co-alocadas.

Neste mesmo trabalho, os autores identificam que a interferência sofrida não é determinada totalmente pelo compartilhamento de um único recurso, como o SSLC, mas sim, pelo conjunto de recursos compartilhados.

Em adição, as camadas adicionais de *software* responsáveis pela emulação e virtualização de dispositivos, e o nível de concorrência dentre os sistemas convidados resultam em um custo computacional [62]. O escalonamento realizado pelo monitor de máquinas virtuais também é responsável pela queda de desempenho em infraestruturas virtualizadas. Usualmente, a política de escalonamento destina maior prioridade para a atividade de particionamento de recursos do processador dentre as máquinas virtuais ativas em vez de escalonar operações de I/O. Por essa razão, processos que realizam operações intensivas de I/O possuem baixo desempenho comparado com processos que desempenham operações intensivas à CPU [63].

Como observado, existem muitas fontes potenciais de contenção de recursos. A perda de desempenho depende das características das aplicações, escalonamento e política de provisionamento de recursos, ou peculiaridades da tecnologia de virtualização. Assim, monitores de desempenho são comumente usados para investigar questões de performance em cenários imprevisíveis. Por coletar informação sobre o uso de recursos em tempo de execução, monitores são ferramentas essenciais em centro de dados e infraestruturas de nuvem. No entanto, a coleta de dados realizada por monitores introduzem uma sobrecarga ao sistema não negligenciável. Sistemas de grande escala coletam com frequência milhões de contadores de desempenho durante monitoramento [15]. No Capítulo 5 é realizada uma avaliação do impacto na sobrecarga causado por contadores de desempenho em ambientes virtualizados.

4 MÉTODO PARA A SELEÇÃO DE CONTADORES

Neste capítulo é descrita a metodologia proposta para a seleção de contadores de desempenho para o monitoramento de sistemas. O método proposto tem como objetivo indicar o conjunto de contadores de desempenho necessários para a observação de mudanças no estado de utilização de recursos de um determinado sistema. Por isso, são eliminados contadores desnecessários para o monitoramento do sistema, e também aqueles considerados redundantes. Dessa forma, uma menor quantidade de dados precisa ser processada e armazenada.

Os benefícios com a utilização do método incluem: melhor desempenho das aplicações devido a liberação de recursos computacionais antes contidos pela atividade de monitoramento; a eliminação da subjetividade inerente ao processo manual de seleção de contadores desempenho; e a eliminação da necessidade de um especialista para realizar a atividade de seleção dos contadores de desempenho para o monitoramento de sistemas.

O método desenvolvido para a seleção de contadores de desempenho aplica o processo de KDD para análise de dados experimentais. A Figura 8 ilustra o método proposto na forma de um fluxograma, destacando a técnica, os processos envolvidos, dados de entrada e saída. O método, com característica típica de processos de KDD, é iterativo.

A aplicação do método inicia no *processo de experimentação* e, em sequência, realiza o processo de KDD. Apenas na última etapa do processo do KDD, o fluxo de execução retorna para o *processo de experimentação*. Essa iteração ocorre para que os diferentes cenários de utilização de recursos conhecidos do sistema sejam reproduzidos pelo *processo de experimentação.* Assim, permitindo que as informações obtidas no KDD sejam confirmadas ou detalhadas, sobre as relações entre contadores de desempenho. Após realizar uma iteração por cenário conhecido, o método encaminha-se para o *processo de seleção de contadores de desempenho representativos*, término do método proposto.

A seguir são discutidos cada um dos processos ilustrados no fluxograma presente na Figura 8. O fluxo de execução dos processos caracteriza a aplicação do método de seleção de contadores de desempenho proposto para o monitoramento de sistemas.

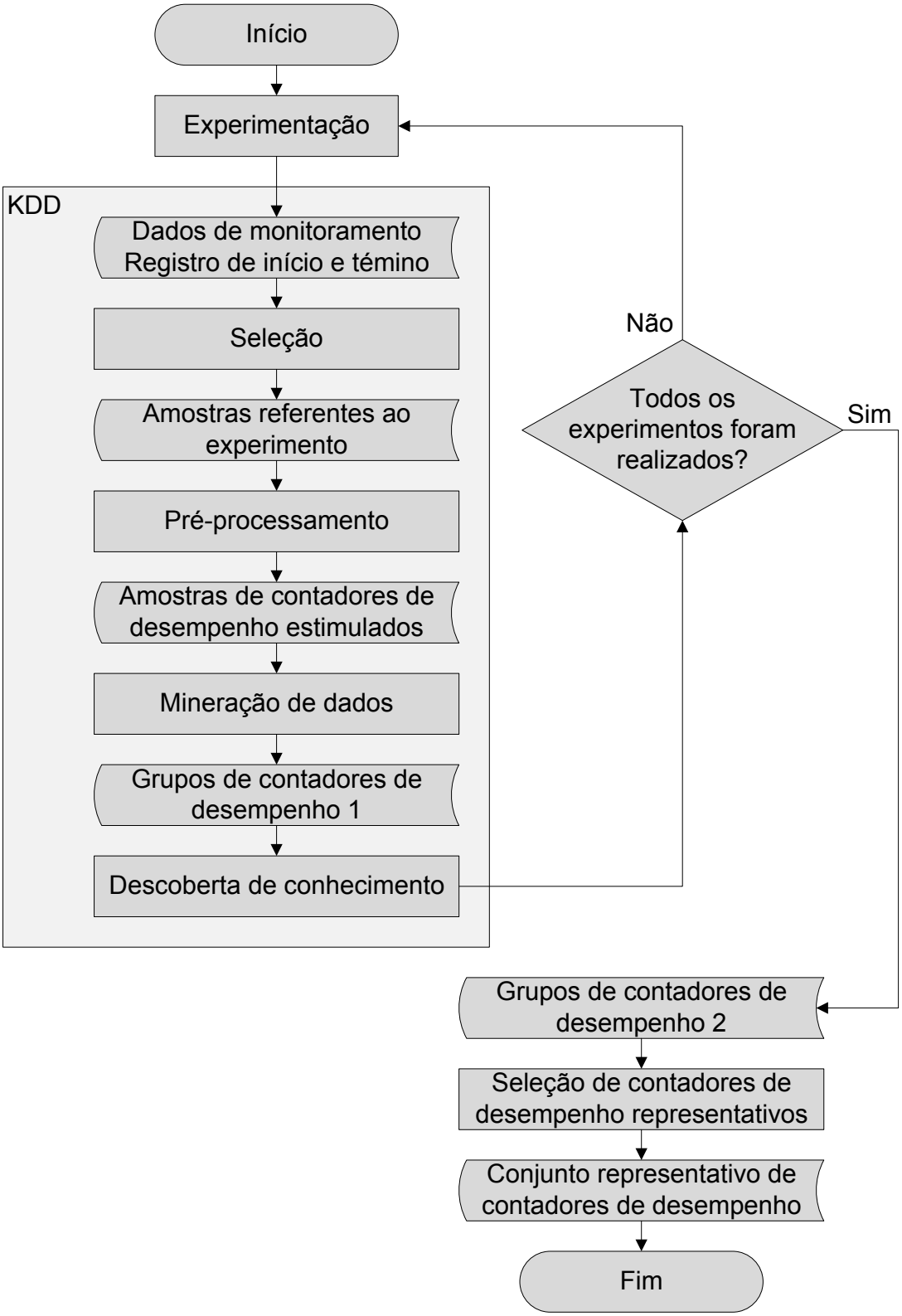


Figura 8: Representação em fluxograma da metodologia proposta para a seleção de contadores de desempenho para o monitoramento de sistemas.

4.1 Experimentação

O *processo experimentação* compreende a coleta de dados de monitoramento de utilização de recursos do sistema em forma de registro. Durante o monitoramento, ocorre a reprodução de uma carga de trabalho sobre o sistema em monitoramento. O sistema utilizado no *processo de experimentação* deve reproduzir as características do sistema em produção que se quer monitorar em termos de *hardware*, sistema operacional e aplicação.

A cada iteração da metodologia – que se inicia no *processo experimentação* e termina no *processo descoberta de conhecimento* – é analisado o registro de monitoramento referente a uma dada carga de trabalho. A carga de trabalho em análise deve ser diferente das cargas submetidas ao *processo experimentação* em iterações anteriores, caso a referente análise não ocorra na primeira iteração. Durante a primeira iteração pode ser utilizada alguma das cargas conhecidas de utilização de recursos submetidas ao sistema.

As cargas de trabalho devem se diferenciar quanto ao perfil de demanda gerado ao sistema. Ou seja, devem considerar diferentes estímulos aos componentes do sistema, ou impor diferentes comportamentos. Em adição, a intensidade das cargas de trabalho devem variar ao longo do experimento – *i.e.*, em quantidade de requisições por segundo. Esse último aspecto garante que os estímulos gerados ao sistema não reflitam na coleta de valores fixos pelos contadores de desempenho que caracterizam a carga. Logo, não é recomendável a utilização de *benchmarks* como cargas de trabalho, quando esses possuem o objetivo único de testar a vazão máxima suportada pelo sistema.

A utilização de cargas sintéticas com intensidade variável ao longo de sua execução, e que estimulem componentes utilizados pelo sistema em questão, beneficia o resultado obtido pela metodologia proposta. Cargas de trabalho como essas possuem a capacidade de gerar estímulos que produzem maior dispersão nos valores coletados por alguns contadores de desempenho. Assim, possibilita melhor observação das relações existentes entre os contadores de desempenho estimulados pelo sistema em estudo.

Quanto ao intervalo de amostragem utilizado no monitoramento, ou seja, o intervalo de tempo entre coletas de dados fornecidos por contadores de desempenho, é importante que seja capaz de possibilitar a captura de transições de comportamento das cargas de trabalho. Intervalos extensos transformam as transições em médias de utilização de recursos. E, por consequência, a aferição de contadores relacionados pode ficar prejudicada.

A Tabela 5 ilustra o conteúdo gerado pelo *processo experimentação*. Cada linha da tabela representa uma amostra de dados fornecidos por contadores de desempenho. A coluna registro de tempo indica o momento que ocorreu a coleta da amostra. As demais colunas indicam a origem dos dados coletados, ou seja, os contadores de desempenho. Os índices n e m representam, respectivamente, a quantidade de amostras coletadas e a quantidade de contadores de desempenho utilizados durante o monitoramento do sistema. É possível que formatações diferentes sejam obtidas em decorrência da ferramenta de

monitoramento utilizada. A apresentação do método proposto considera que todos os dados estejam dispostos conforme apresentado na Tabela 5.

Tabela 5: Disposição de dados de monitoramento do sistema.

Registro de tempo	Contador 1	Contador 2	...	Contador m
tempo 1	amostra 1.1	amostra 1.2	...	amostra 1.m
tempo 2	amostra 2.1	amostra 2.2	...	amostra 2.m
...	...	...	...	...
tempo n	amostra n.1	amostra n.2	...	amostra n.m

4.2 Seleção

Após o *processo experimentação*, o registro de amostras de utilização de recursos, representado pela Tabela 5, é submetido ao *processo seleção*. O *processo seleção* seleciona as amostras de interesse. Somente são avaliadas pelo método as amostras pertencentes ao intervalo de tempo em que o sistema foi estimulado pela carga de trabalho. Além disso, amostras referentes ao início e ao término do experimento são descartadas, pois não representam o comportamento habitual do sistema em estudo.

Os registros de tempo de início e término do experimento são necessários para a eliminação das amostras que não são analisadas pelo método. Essas informações são fornecidas pelo *benchmark*, pelo *script* gerador de requisições executado, ou pelas informações do agendador de tarefas, conforme for a abordagem empregada no *processo experimentação*.

Benchmarks produzirem dados de saída durante a sua execução. Como por exemplo: o registro de tempo inicial; estatísticas de desempenho acompanhadas de registro de tempo; e estatísticas finais acompanhada do registro de tempo do término da execução. Quando a informação de início e término da execução não é disponibilizada pelo *benchmark*, é possível criar um *script* contendo a execução do *benchmark* intercalada por instruções que realizam os registros de tempo. Também é possível agendar a execução do *benchmark* em determinado tempo através de ferramenta disponível no sistema operacional. Dessa forma o registro de tempo inicial é definido previamente, e o registro de tempo final pode ser obtido pelo próprio *benchmark* ou através de parâmetro de execução para o tempo de duração da carga de trabalho.

A definição da quantidade de amostras descartadas do início e do término do registro de amostras de utilização de recursos é dependente do intervalo de monitoramento adotado e do sistema em análise. A quantidade de amostras descartadas no início deve abranger o tempo de aquecimento (*warmup*) do sistema, e as amostras descartadas no final devem compreender o tempo de finalização da carga executada, assim como as amostras referentes a momentos em que a carga de trabalho não esteve em execução.

A Tabela 6 exemplifica as amostras eliminadas devido ao momento de inicialização da carga de trabalho e ao tempo de aquecimento. Essas amostras foram coletadas no *tempo 1*, *tempo 2* e *tempo 3*. Nesse caso é possível que a carga de trabalho tenha iniciado no *tempo 2* e o tempo de aquecimento se estendido próximo ao *tempo 3*. A Tabela 6 também exemplifica as amostras eliminadas devido ao momento de finalização da carga de trabalho e ao tempo de desalocação de recursos. Em analogia, é possível que a carga de trabalho tenha terminado no *tempo n-2* e o tempo de encerramento do processo se estendido próximo ao *tempo n-1*. Logo, os registros de tempos entre o *tempo 4* e o *tempo n-3* seriam considerados na análise.

Tabela 6: Registro de amostras de utilização de recursos selecionados.

Registro de tempo	Contador 1	Contador 2	...	Contador m
~~tempo 1~~	~~amostra 1.1~~	~~amostra 1.2~~	~~...~~	~~amostra 1.m~~
~~tempo 2~~	~~amostra 2.1~~	~~amostra 2.2~~	~~...~~	~~amostra 2.m~~
~~tempo 3~~	~~amostra 3.1~~	~~amostra 3.2~~	~~...~~	~~amostra 3.m~~
tempo 4	amostra 4.1	amostra 4.2	...	amostra 4.m
...	...	...	...	...
tempo n-3	amostra n-3.1	amostra n-3.2	...	amostra n-3.m
~~tempo n-2~~	~~amostra n-2.1~~	~~amostra n-2.2~~	~~...~~	~~amostra n-2.m~~
~~tempo n-1~~	~~amostra n-1.1~~	~~amostra n-1.2~~	~~...~~	~~amostra n-1.m~~
~~tempo n~~	~~amostra n.1~~	~~amostra n.2~~	~~...~~	~~amostra n.m~~

4.3 Pré-processamento

Esse processo é responsável pela eliminação dos registros referentes aos contadores de desempenho que não foram estimulados pela carga de trabalho submetida ao sistema durante o *processo experimentação*. Esses atributos não são utilizados pelos processos subsequentes pertencentes à mesma iteração. Dados faltantes podem indicar que o contador de desempenho está relacionado à componente não utilizado pelo sistema. Em adição, medição de uso de contadores com variância igual à zero representa a ausência de estímulo ou comportamento estacionário dos contadores de desempenho.

Supondo que o *Contador 1* e o *Contador 2*, presentes na Tabela 6, apresentam dados faltantes nas amostras coletadas ou variância igual a zero ($s^2 = 0$, segundo a Equação 5), então são removidos do conjunto de dados, conforme ilustrado pela Tabela 7.

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1} \tag{5}$$

O cálculo da variância dado pela Equação 5, é aplicado para cada um dos contadores de desempenho presentes no registro de amostras de utilização de recursos representado pela Tabela 6. A variável x_i é utilizada para armazenar cada um dos valores obtidos pelas amostras do referido contador de desempenho. A representação de cada amostra na

equação é feita pelo índice i, que varia de 1 a n. O índice igual a 1 representa a primeira amostra não eliminada, e o valor n assume a última amostra não eliminada.

Tabela 7: Registro de amostras de utilização de recursos pré-processados.

~~Registro de tempo~~	~~Contador 1~~	~~Contador 2~~	...	Contador m
~~tempo 4~~	~~amostra 4.1~~	~~amostra 4.2~~	...	amostra 4.m
~~tempo 5~~	~~amostra 5.1~~	~~amostra 5.2~~	...	amostra 5.m
...	...	...	...	...
~~tempo n-3~~	~~amostra n-3.1~~	~~amostra n-3.2~~	...	amostra n-3.m

Observa-se na Tabela 7 que há uma menor quantidade de amostras quando comparada com a Tabela 5. Essa redução é devido à eliminação ocorrida no processo seleção. No pré-processamento observa-se a eliminação de colunas. A primeira coluna removida é referente aos dados de registro de tempo das amostras. Esses dados não são mais utilizados para a aplicação do método. Observa-se também que colunas referentes a dados obtidos de contadores de desempenho foram eliminadas. A eliminação de colunas ocorre quando os valores de uma mesma coluna são dados faltantes ou possuem variância igual a zero. A identificação dos contadores de desempenho removidos e o motivo das remoções são informações que podem ser utilizadas no *processo descoberta de conhecimento*.

4.4 Mineração de dados

O processo de mineração de dados adotado pela metodologia utiliza um método de agrupamento. Segundo Han et al. [64] agrupamento consiste no processo de agrupar os dados em grupos, nos quais os objetos de um grupo possuem alta similaridade entre si, mas quando comparados com objetos de outros grupos, possuem uma alta dissimilaridade. A maioria dos algoritmos de agrupamento baseados em memória considera como dados de entrada uma matriz com dados específicos.

A metodologia proposta utiliza como dados de entrada a matriz de dissimilaridade descrita na Subseção 4.4.1. A especificação do método de agrupamento está presente na Subseção 4.4.2, e o método para obtenção dos padrões está descrito na Subseção 4.4.3.

4.4.1 Passo 1: Matriz de dissimilaridade

O registro de amostras de utilização de recursos pré-processado é utilizado para construção de uma matriz de dissimilaridade. A matriz de dissimilaridade armazena uma coleção de proximidades para todos os n pares de objetos sob análise. Na metodologia desenvolvida, os contadores de desempenho são os objetos.

O *processo mineração de dados* implementado utiliza o coeficiente de correlação linear de Pearson (r) [50] – descrito pela Equação 1 – como base para o cálculo da medida de dissimilaridade. O coeficiente de Pearson é uma medida da correlação linear entre dois

objetos. O coeficiente pode assumir valores dentro do intervalo $[-1, 1]$ no domínio dos números reais. Os valores extremos indicam uma correlação perfeita: -1 para correlação negativa; e 1 para correlação positiva. O valor 0 indica a não existência de correlação. O quanto mais distante de 0 for o valor do coeficiente, mais forte é a correlação: correlação negativa entre 0 e -1; ou positiva, entre 0 e 1.

Após o cálculo do coeficiente de correlação linear de Pearson, r, para os pares de contadores de desempenho, representados por x e y, os coeficientes $r(x, y)$ são transformados em um coeficiente de distância $d(x, y)$. A metodologia proposta emprega a Equação 6 para realizar a transformação. Como resultado da transformação as correlações mais fortes (positiva ou negativa) assumem ou se aproximam do valor 0 para o coeficiente de dissimilaridade e, as correlações fracas, do valor 1. Dessa forma, é preenchida a matriz de dissimilaridade, conforme apresenta a matriz M representada pela Equação 7, e torna-se possível avaliar as correlações sem considerar o tipo de correlação linear envolvida, ou seja, se é positiva ou negativa.

$$d(x, y) = \begin{cases} 1 - r(x, y) & para \quad r(x, y) \geqslant 0 \\ r(x, y) + 1 & para \quad r(x, y) < 0 \end{cases} \tag{6}$$

Cada interseção de uma linha com uma coluna da Matriz M (Equação 7) possui um valor de dissimilaridade que determina a dissimilaridade calculada entre o contador representado pela linha e o contador representado pela coluna. Os contadores considerados são os contadores restantes do *processo pré-processamento*. As amostras de utilização de recursos obtida por cada contador são utilizadas para o cálculo da dissimilaridade. A Equação 7 exemplifica a matriz de dissimilaridade ***M***, onde $d(j_l, j_c)$ é a dissimilaridade calculada entre os contadores j_l e j_c, antes referenciados pela Equação 1 por x e y.

$$M = \begin{bmatrix} d(j_2, j_1) & & & \\ d(j_3, j_1) & d(j_3, j_2) & & \\ \quad ... & \quad ... & ... & \\ d(j_n, j_1) & d(j_n, j_2) & ... & d(j_n, j_{n-1}) \end{bmatrix} \tag{7}$$

É importante destacar que, para fins de visualização, os contadores recebem uma nova identificação. O contador antes identificado como *Contador 3* agora é identificado por j_1, *Contador 5* por j_2, e assim por diante. A identificação utilizada anteriormente retorna a ser utilizada ao decorrer do capítulo.

4.4.2 Passo 2: Agrupamento de contadores similares

O *processo de mineração de dados* aplica um algoritmo de agrupamento hierárquico. Esse tipo de método de agrupamento cria uma decomposição hierárquica de um conjunto de objetos de dados. As abordagens de algoritmos de agrupamento hierárquicos podem

ser classificadas como agrupamento aglomerativo ou agrupamento divisivo. O processo emprega um algoritmo hierárquico aglomerativo chamado de *complete linkage* [65].

Nesse algoritmo, cada objeto da matriz de dissimilaridade compõe um grupo. A matriz de dissimilaridade é usada para encontrar a menor distância entre dois grupos. A distância entre dois grupos é dada pela distância dos objetos mais distantes. Os dois grupos com menor distância encontrados são agrupados. O processo é repetido iterativamente analisando a próxima menor distância, até formar apenas um grupo. O processo resulta em um dendrograma, ilustrado pela Figura 9. A metodologia proposta utiliza como objetos os contadores de desempenho presentes na matriz de dissimilaridade encontrada conforme descrito na subseção anterior (Matriz M, Equação 7).

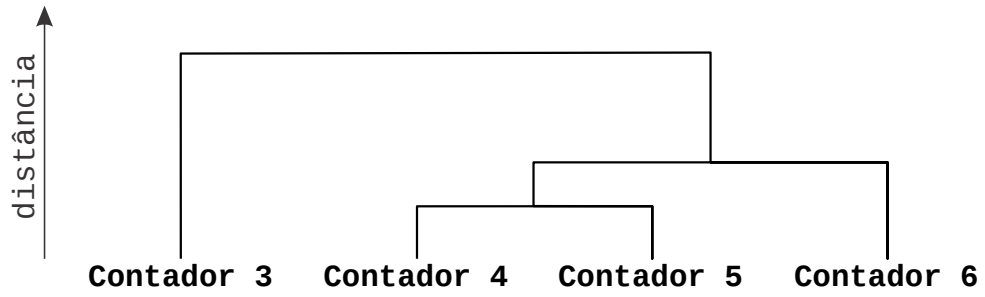


Figura 9: Dendrograma gerado por agrupamento hierárquico.

A Figura 9 apresenta as possibilidades de agrupamento através da hierarquia traçada pela dissimilaridade existente entre os grupos. Os segmentos horizontais que ligam dois grupos são traçados observando o eixo dissimilaridade. A dissimilaridade entre dois grupos agrupados é registrada pela posição do segmento horizontal no eixo dissimilaridade. É possível observar que os dois primeiros grupos agrupados foram o grupo contendo o objeto *Contador 4* com o grupo contendo o objeto *Contador 5*. O próximo grupo agrupado foi o grupo contendo os objetos *Contador 4* e *Contador 5* com o grupo contendo o objeto *Contador 6*. Finalmente o grupo contendo os objetos *Contador 4*, *Contador 5* e *Contador 6* foi agrupado com o grupo contendo o objeto *Contador 3*.

4.4.3 Passo 3: Definição de grupos de contadores

O agrupamento hierárquico define uma hierarquia de agrupamento, porém não define uma quantidade de grupos que deve utilizada para análise. Com o objetivo de definir essa quantidade, é possível especificar uma distância que o dendrograma deve ser cortado, conforme ilustra a Figura 10. Os contadores conectados pelos ramos abaixo do ponto de corte compõem um grupo. Por exemplo, *contador 4* e *contador 5* compõem o mesmo grupo e os contadores *contador 3* e *contador 6* formam separadamente outros dois grupos diferentes.

Considerando que os valores da matriz de dissimilaridades mais próximos de 0 correspondem a uma correlação forte entre os pares de contadores, e valores próximos de 1

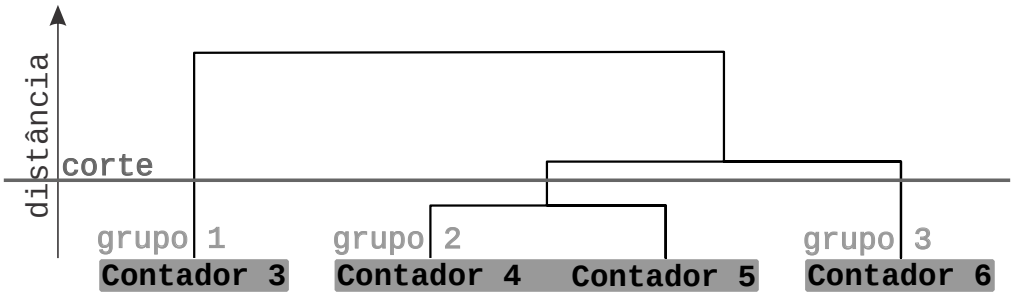


Figura 10: Corte aplicado a um dendrograma.

indicam que os pares de contadores têm baixa similaridade, então, para formar grupos de contadores com correlação forte entre os possíveis pares, o corte aplicado a um dendrograma deve ser próximo do valor 0 de distância. A utilização do método do algoritmo *complete linkage* garante que todos os contadores de desempenho pertencentes ao mesmo grupo possuam entre si uma dissimilaridade inferior a especificada pelo corte.

O ponto de corte pode ser obtido por muitos métodos de corte de árvore [66] [67]. Calinski-harabasz [68] é um exemplo de método usado com sucesso por Shang et. al. [15] para executar o agrupamento hierárquico de contadores de desempenho. No entanto, com a utilização desse método automático, ou de outros, como o Silhouette [69], não há garantia de que o corte esteja próximo ao valor 0 de distância. Por esse motivo, a metodologia proposta utiliza um corte de árvore estático. O ponto de corte sugerido é em $distancia = 0.1$ (isto é, $r = \pm 0,9$). De acordo com Taylor e Richard [70], o coeficiente r descreve uma correlação muito forte quando $r \geq 0.9$ ou $r \leq -0.9$.

4.5 Descoberta de conhecimento

O *processo descoberta de conhecimento* tem como objetivo verificar os padrões de agrupamento de contadores de desempenho. Esses padrões são evidenciados ao decorrer das iterações da metodologia proposta. Os grupos de contadores de desempenho encontrados em cada iteração são comparados com os demais. Essa comparação visa identificar os contadores de desempenho que permanecem similares em diferentes iterações, ou seja, continuam fazendo parte de um mesmo grupo. Também visa evidenciar aquelas similaridades entre contadores que ocorrem somente em cenários específicos de utilização de recursos do sistema, conforme o cenário no processo de experimentação.

O *processo descoberta de conhecimento* trabalha com dois conjuntos, A e B. O conjunto B contém os grupos de contadores encontrados em iteração atual (*processo mineração de dados*), $B = \{B_1, B_2, ..., B_m\}$. O conjunto A contém grupos de contadores proveniente de iteração anterior (*processo descoberta de conhecimento*), $A = \{A_1, A_2, ..., A_n\}$. Cada elemento B_j e A_i é um grupo de contadores. Os índices j e i identificam os grupos e variam de 1 até m e n, respectivamente. Os valores de m e n também indicam a quantidade de grupos contida nos conjuntos. O *processo descoberta*

de conhecimento na primeira iteração retorna o conjunto A como $A = \{B\}$.

Nas iterações seguintes, para toda relação $A_i \cap B_j \neq \{\emptyset\}$ verdadeira é criado um novo grupo $A_{n+1} = \{A_i \cap B_j\}$ para o mesmo par de índices, e o conjunto A é atualizado para $A = \{A \cup A_{n+1}\}$. Em sequência, o conteúdo desse par de grupos é atualizado para: $A_i = \{A_i - A_{n+1}\}$; e $B_j = \{B_j - A_{n+1}\}$. Antes de testar outra relação $A_i \cap B_j$, o valor de n é atualizado para $n \leftarrow n + 1$. Ao final, após verificar todas as relações $A_i \cap B_j$ considerando o valor inicial de n na iteração atual, para cada conjunto $B_j \neq \{\emptyset\}$ é criado um novo conjunto $A_{n+1} = \{B_j\}$ considerando o mesmo índice e, em sequência, atualizado o conjunto A para $A = \{A \cup A_{n+1}\}$ e o valor de n para $n \leftarrow n+1$. O processo descrito, para uma iteração, é ilustrado pela Figura 11.

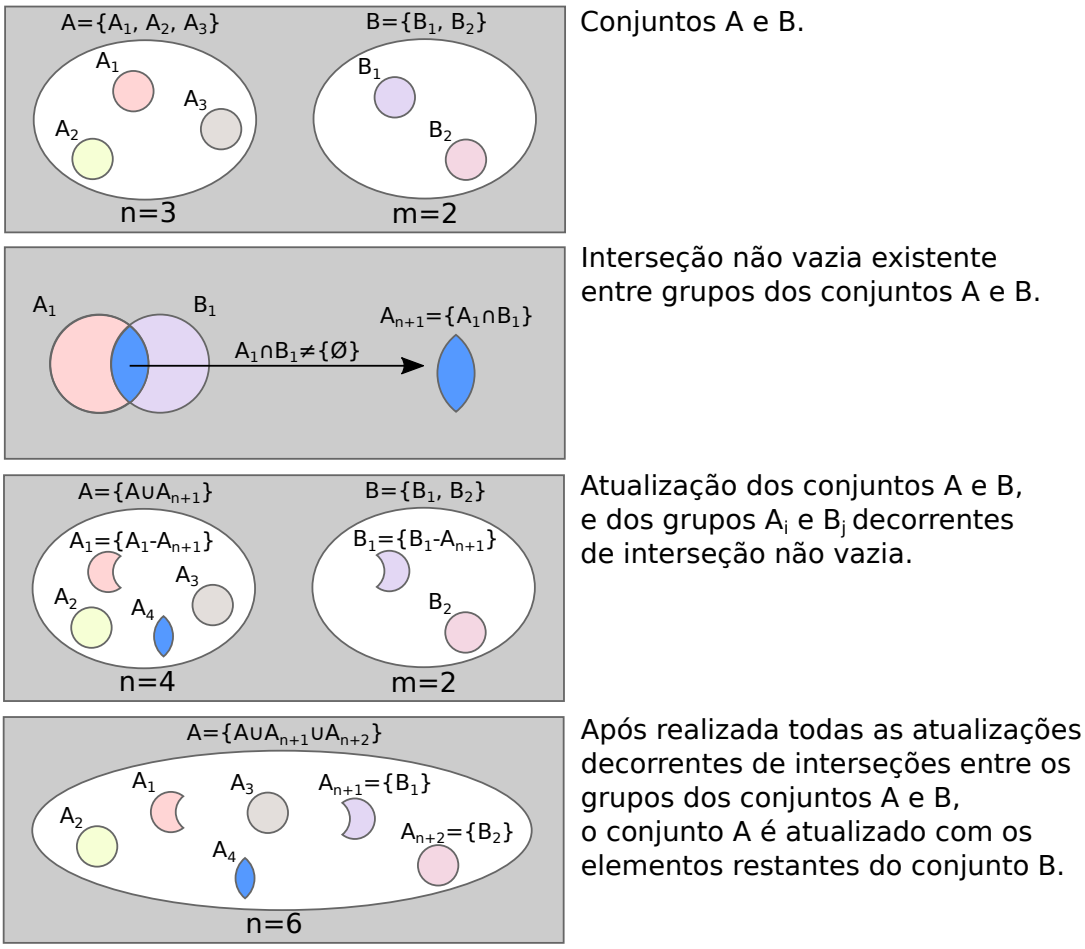


Figura 11: Verificação de padrões durante uma iteração no processo de descoberta de conhecimento.

O conjunto A retornado pela última iteração contém grupos em que os contadores de desempenho de cada grupo também pertencem a um mesmo grupo nos diferentes conjuntos B tratados. Dessa forma, são agrupados os contadores de desempenho que mantêm correlação independente do cenário de utilização de recursos. O conjunto A obtido na última iteração é utilizado no *processo seleção de contadores de desempenho representativos*.

4.6 Seleção de contadores de desempenho representativos

A partir dos conjuntos de contadores de desempenho obtidos no processo anterior é formando um conjunto representativo de contadores de desempenho. Esse conjunto representativo é o conjunto de contadores encontrado pela metodologia para a realização do monitoramento do sistema sob análise. A obtenção desse conjunto é realizada através da seleção de um único contador de cada um dos conjuntos obtidos no *processo descoberta de conhecimento*, referente à última iteração.

A seleção realizada leva em consideração a dissimilaridade entre os contadores de desempenho. Aquele contador que melhor representa os demais é escolhido. Essa escolha é baseada no somatório de dissimilaridade de cada contador com os demais pertencentes ao mesmo conjunto. O contador que possui o menor somatório é o contador selecionado para representar o conjunto de origem. Por exemplo, considerando os contadores *c1*, *c2* e *c3* pertencentes a um mesmo conjunto, e as respectivas dissimilaridades $d(c1, c2) = 0.01$, $d(c1, c3) = 0.05$ e $d(c2, c3) = 0.1$, o contador com menor somatório será o contador *c1*, com somatório igual a 0.06.

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo evidencia os benefícios de utilização do método proposto no Capítulo 4. É dado destaque à contenção de recursos como efeito gerado pela atividade de monitoramento do sistema, mostrando que esse efeito pode ser reduzido através da seleção de contadores de desempenho. A avaliação do método exige implementação e utilização de artefatos computacionais, como por exemplo: ambiente de experimentação; cargas de trabalho para estimular o sistema em estudo; e ferramenta de monitoramento.Este capítulo descreve os meios utilizados e implementados para avaliação do método.

A avaliação do método para seleção de contadores de desempenho de sistemas baseia-se em dois principais objetivos. Um objetivo é analisar a sobrecarga gerada pelo uso desse conjunto de contadores para o monitoramento do sistema, de forma que, seja possível caracterizar a contenção de recursos decorrente da atividade de monitoramento. Outro objetivo é verificar a representatividade do conjunto de contadores formado. Ou seja, se o conjunto de contadores encontrado é capaz de caracterizar os diferentes estados de utilização do sistema. Logo, perante mudanças no perfil de carga de trabalho o conjunto de contadores deve representar os mesmos grupos de origem. Os resultados da avaliação experimental realizada neste capítulo permitem dissertar sobre: a eliminação da subjetividade no processo de seleção de contadores de desempenho; redução de sobrecarga gerada pelo monitoramento de ambientes virtualizados; preservação de dados que descrevem o estado do sistema; e, principalmente, sobre a efetividade e vantagens de utilização do método proposto para a seleção de contadores de desempenho para o monitoramento de sistemas.

Este capítulo está organizado da seguinte forma: a Seção 5.1 descreve os *bechmarks* utilizados para a geração de carga durante a realização de experimentos; a Seção 5.2 apresenta o ambiente de experimentação utilizado; a Seção 5.3 expõe uma avaliação de sobrecarga gerada pelo monitoramento do sistema; a Seção 5.4 apresenta a redução de sobrecarga obtida pelo uso do conjunto de contadores de desempenho encontrado pela aplicação do método; e, por fim, a Seção 5.5 traz uma avaliação do conjunto de contadores encontrado.

5.1 Geração de carga

Devido a propósitos funcionais distintos ou otimizações específicas as aplicações geram diferentes padrões de carga aos componentes do sistema, a sequencialidade de operações realizada por uma aplicação pode gerar correlação entre métricas de utilização de recursos particular a essa aplicação. Portanto, estimar a contenção de recursos gerada pela atividade de monitoramento do sistema, baseado somente em carga mista de utilização de recursos, limitaria a caracterização da sobrecarga gerada ao sistema.

Com o objetivo de encontrar correlações e entender de que forma o monitoramento afeta o desempenho do sistema, são utilizados *microbenchmarks* na realização de experimentos. Dessa forma, além das cargas representarem padrões presentes em aplicações, é possível gerar cargas controladas, reprodutíveis e destinadas a componentes específicos do sistema. Em contra partida, são utilizados *macrobenchmarks* para simular os possíveis comportamentos de um sistema ao qual se quer monitorar. Em sequência, são descritos os *benchmarks* utilizados.

5.1.1 *Microbenchmarks*

Benchmarks são utilizados tradicionalmente como geradores de cargas intensivas com o objetivo de descobrir a vazão máxima que o sistema consegue extrair de determinado componente ou configuração [71]. *Microbenchmarks* utilizam código-fonte pequeno que realiza tarefa específica, com foco em estimular determinado recurso [72]. Essa característica é importante e utilizada em trabalho anterior para mensurar a sobrecarga gerada pela utilização de contadores de desempenho implementados em *hardware* [26]. Com objetivos similares, utilizamos cargas intensivas de *microbenchmarks* para investigar a contenção de recursos, em ambientes virtualizados, causada pelo uso de contadores de desempenho implementados pelo sistema operacional.

No entanto, cargas intensivas podem elevar métricas aos valores máximos, com pouca ou nenhuma variação nos valores coletados pelos contadores. Dessa forma, a observação de correlações entre métricas fica prejudicada. Portanto, foram implementados *microbenchmarks* com a capacidade de reduzir e intensificar a carga gerada durante a execução. Esse comportamento é configurável e trabalha com quantidades de *threads* e tempo ocioso, que podem variar em tempo de execução. Os *microbenchmarks* desenvolvidos geram cargas de trabalho com as seguintes características:

- Carga intensiva à CPU: Implementa o cálculo do número π. Cada *thread* calcula o número π com n casas decimais. O cálculo é repetido até a finalização do experimento. Tanto n, quanto o tempo de execução são parâmetros do *benchmark*.

- Carga intensiva à memória principal: Implementa operações de leitura e escrita em um vetor pré-alocado de números inteiros com tamanho de m MB. O tempo de execução e m são parâmetros do *benchmark*.

- Carga intensiva à memória secundária: Implementa operações de leitura e escrita aleatórias em memória secundária. Para cada *thread* são criados *n* arquivos com tamanho de *m* MB. O tamanho dos blocos, percentagem de leituras e de escritas, e tempo de execução são parâmetros do *benchmark,* juntamente com *n* e *m*.

Os *benchmarks* implementados também possuem a capacidade de gerar arquivos de registro com a vazão média de operações realizadas durante a geração da carga. Esses dados são disponibilizados em série temporal, com intervalo de amostragem ajustável.

5.1.2 *Macrobenchmarks*

Macrobenchmarks são utilizados para medir o desempenho do sistema como um todo. São tradicionalmente aplicados para comparar configurações de sistemas para uma determinada classe de aplicação. Com isso, *macrobenchmarks* auxiliam na definição de necessidades de *hardware* e melhores configurações de aplicações para alcançar o desempenho desejado ao sistema [73].

O YCSB (*Yahoo! Cloud Serving Benchmark*) é um *framework* consolidado para a realização de testes de sistemas com bancos de dados NoSQL e que oferecem serviços em nuvem [74] [5] [75]. O YCSB oferece compatibilidade com inúmeros bancos de dados [76] através de interfaces de ligação (Accumulo [77], Cassandra [78], Couchbase [79], DynamoDB [80], ElasticSearch [81], HBase [82], HyperTable [83], Infinispan [84], JDBC [85], MongoDB [86], OrientDB [87], Redis [88] e Tarantool [89]).

O YCSB é utilizado para avaliação da escalabilidade, elasticidade e disponibilidade de banco de dados NoSQL. É possível avaliar decisões de arquitetura que impactam nas relações entre: desempenho de leitura e desempenho de escrita; latência e durabilidade; replicações síncronas e assíncronas; e particionamentos de dados [5].

A utilização do YCSB [90] é dividida em duas fases, fase de carregamento (*load phase*) e fase transacional (*transaction phase*). Ambas as fases seguem o modelo cliente-servidor. A fase de carregamento está relacionada à preparação do sistema que será testado, momento em que é realizado o carregamento da base de dados. A fase transacional está relacionada à execução do teste, momento em que é realizada a geração de carga de trabalho no sistema em análise.

O sistema a ser testado deve oferecer como serviço o acesso a uma base de dados com uma tabela. O primeiro campo da tabela é destinado à chave primária e os demais campos aos atributos. A chave primária é do tipo cadeia de caracteres com tamanho máximo de 255 caracteres. Os demais campos são do tipo texto. Por padrão, a base de dados, a tabela, a chave primária e os atributos possuem respectivamente como nomes: *ycsb*; *usertable*; *YCSB_KEY*; *field*0, *field*1, *field*2, conforme a quantidade de atributos.

A configuração do cliente YCSB para carregar a tabela e gerar cargas de trabalho permite definir a quantidade e tamanho dos atributos. Por padrão são 10 atributos com

tamanho de 100 bytes cada, totalizando 1.000 bytes para cada registro sem considerar o tamanho da chave primária.

O tamanho da tabela não é definido um padrão, sendo este definido pela quantidade de registros inseridos. Esse tamanho pode sofrer alterações durante a execução de cargas de trabalho. Por isso, deve ser dada atenção ao tamanho atual da tabela para não haver problemas com execuções consecutivas de cargas de trabalho que realizam novas inserções.

O carregamento da base de dados e a geração de carga de trabalho é dependente da configuração de dois conjuntos de propriedades, propriedades de carga de trabalho e propriedades de tempo de execução:

- Propriedades de carga de trabalho: as principais propriedades definem os tipos de operações, as probabilidades de ocorrência de cada tipo de operação, o tipo de distribuição de probabilidade para a seleção de registros acessados pelas operações, a quantidade de registros da base de dados e a especificação da quantidade e tamanho dos atributos. As Tabelas 8 e 9 apresentam, respectivamente, os tipos de operações e de distribuições de probabilidade disponíveis no YCSB.

- Propriedades de tempo de execução: abrangem as configurações de comunicação com o banco de dados (interface de ligação, nome do servidor, usuário, senha, etc.), e a definição de propriedades que influenciam na carga de trabalho (quantidade de *threads* e a vazão imposta a cada *thread*).

Tabela 8: Tipos de operações disponíveis ao cliente YCSB [5].

Operação	**Descrição**
Inserção (*insert*)	Insere um novo registro na base de dados.
Leitura (*read*)	Lê um campo aleatório ou todos os campos do registro selecionado.
Atualização (*update*)	Atualiza um dos campos do registro selecionado.
Busca (*scan*)	Busca em ordem uma quantidade aleatória de registros. A busca inicia a partir de uma chave escolhida aleatoriamente.

Tabela 9: Tipos de distribuições para seleção de registros disponíveis ao cliente YCSB [5].

Distribuição	**Descrição**
Uniforme (*Uniform*)	Todos os registros da base de dados possuem a mesma chance de ser escolhido.
Mais recente (*Latest*)	Quanto mais recente é o registro, maior é a chance do registro ser escolhido.
Zipfian (*Zipfian*)	Alguns registros são considerados mais populares que os demais e por isso possuem maior chance de serem escolhidos.
Multinomial (*Multinomial*)	Probabilidades de leitura, atualização, busca e inserção podem ser especificadas para cada registro.

O YCSB disponibiliza um *template* para a definição das propriedades da carga de trabalho. Em adição, o YCSB fornece a configuração de seis diferentes perfis de carga. Os perfis disponibilizados foram propostos como cargas representativas, semelhantes às presentes em cenários reais, e assim são reconhecidos em diversos estudos [74] [5] [75]. A Tabela 10 descreve cada um desses perfis de carga de trabalho.

Tabela 10: Descrição do perfil das cargas de trabalho pré-configuradas disponíveis para utilização no cliente YCSB [5][6].

Nome – descrição	**Operações**	**Seleção de registro**	**Exemplo de Aplicação**
A – Atualização intensiva	50% de leitura 50% de atualização	Zipfian	Armazenamento de sessão, registro de ações recentes em uma sessão do usuário.
B – Leitura intensiva	95% de leitura 5% de atualização	Zipfian	Marcação de fotos. A adição de uma etiqueta é uma atualização, mas a maioria das operações são de leitura de etiquetas.
C – Apenas leitura	100% de leitura	Zipfian	Cache de perfil de usuário, onde os perfis são construídos em outro lugar (e.g., Hadoop).
D – Leitura mais recente	95% de leitura 5% de inserção	Latest	Atualização de status do usuário. As pessoas querem ler as últimas atualizações.
E – Faixas curtas	95% de busca 5% de inserção	Zipfian e Uniforme	Comunicação entre *threads*. Cada procura é por postagens de uma dada *thread* (assumido ser agrupadas por identificador de *thread*).
F – Leitura-modificação	50% de leitura 50% de leitura e atualização	Zipfian	Banco de dados do usuário, em que os registros do usuário são lidos e modificados pelo usuário ou para registrar a atividade do usuário.

Ao término da execução de uma carga de trabalho, o cliente YCSB fornece estatísticas gerais, e estatísticas específicas para cada tipo de operação envolvida:

- Estatísticas gerais: fornecem o tempo de execução do *benchmark* e a vazão média de requisições por segundo.

- Estatísticas específicas: fornecem, para cada tipo de operação, a quantidade de requisições enviadas, quantidade de requisições concluídas com sucesso, quantidade de requisições não concluídas, latência média, latência mínima, latência máxima e a série temporal de latência média.

5.2 Ambiente de experimentação

Para a realização dos experimentos foi utilizado um ambiente virtualizado. Tanto no servidor hospedeiro quando nas máquinas virtuais foi utilizado o sistema operacional Windows Server 2012 R2. E como *hypervisor* foi utilizado o VMware Workstation 12 Player. Foram utilizadas até duas máquinas virtuais em execução concomitante, com 1 CPU e 2 GB de RAM cada. O servidor hospedeiro utilizado possui processador Intel Xeon E3-1240V5 3.5GHz, 16 GB de memória principal (DDR4 2133MHz em canal duplo) e disco rígido de 7.200 rpm com 64MB de cache e interface Sata III. Os contadores de desempenho utilizados foram os disponíveis por padrão na ferramenta *Performance Monitor*, nativa do sistema operacional.

A versão do YCSB utilizada foi a 0.12.0. O banco de dados utilizado foi o MySQL Server 5.7 através da interface de ligação JDBC. No MySQL foi criada uma base de dados *ycsb* com a tabela *usertable* padrão configurada com a *engine InnoDB*. A tabela foi carregada com 2.000.000 registros, totalizando um tamanho de aproximadamente 2.5 GB para a base de dados. O parâmetro de configuração do banco de dados *buffer_pool_size* foi configurado com 512MB, *buffer_pool_chunk_size* com 128MB e *buffer_pool_instances* com 1. As requisições ao servidor MySQL em ambiente virtualizado foram geradas em uma máquina física. Essa foi configurada com o sistema operacional Windows Server 2008, processador AMD FX-6100 e 8 GB de memória principal (DDR3 1333MHz).

5.3 Avaliação de sobrecarga

A avaliação da sobrecarga é baseada na vazão de operações concluídas pelo *benchmark*. O cálculo da sobrecarga, descrito pela Equação 8, considera: a vazão média produzida durante um experimento com monitoramento (VCM); e a vazão média produzida durante outro experimento sem monitoramento (VSM). Ambas as vazões devem ser originadas de experimentos com a mesma carga de trabalho submetida ao sistema com as mesmas configurações. O resultado obtido indica a percentagem média de operações que deixaram de ser realizadas devido a atividade de monitoramento do sistema.

$$Sobrecarga(\%) = 100 - \left(\frac{VCM \times 100}{VSM}\right) \tag{8}$$

A sobrecarga é avaliada em diferentes cenários de utilização de recursos, com o objetivo de investigar a contenção de recursos gerada pelo monitoramento do sistema. Também são utilizados diferentes perfis de monitoramento, com o propósito de apurar a liberação de recursos ao reduzir a atividade de monitoramento. Os perfis de monitoramento se diferenciam pela quantidade de contadores de desempenho utilizados.

5.3.1 Experimentos com *microbenchmark*

A Figura 12 ilustra, para dois experimentos, a série temporal de *instruções/s* realizadas para cada um dos experimentos. As instruções representam operações de leitura em disco geradas de forma intensiva pelo *microbenchmark* desenvolvido. O cenário dos experimentos compreende na execução de uma única máquina virtual, não havendo outra máquina virtual em execução concomitante no mesmo servidor hospedeiro. Durante o experimento com a vazão representada pela linha tracejada não foi realizado o monitoramento do sistema. Em contra partida, durante o experimento com a vazão representada pela linha sólida, o monitoramento do sistema foi realizado. O monitoramento coletou dados para todos os contadores de desempenho disponíveis em intervalos de 1 segundo.

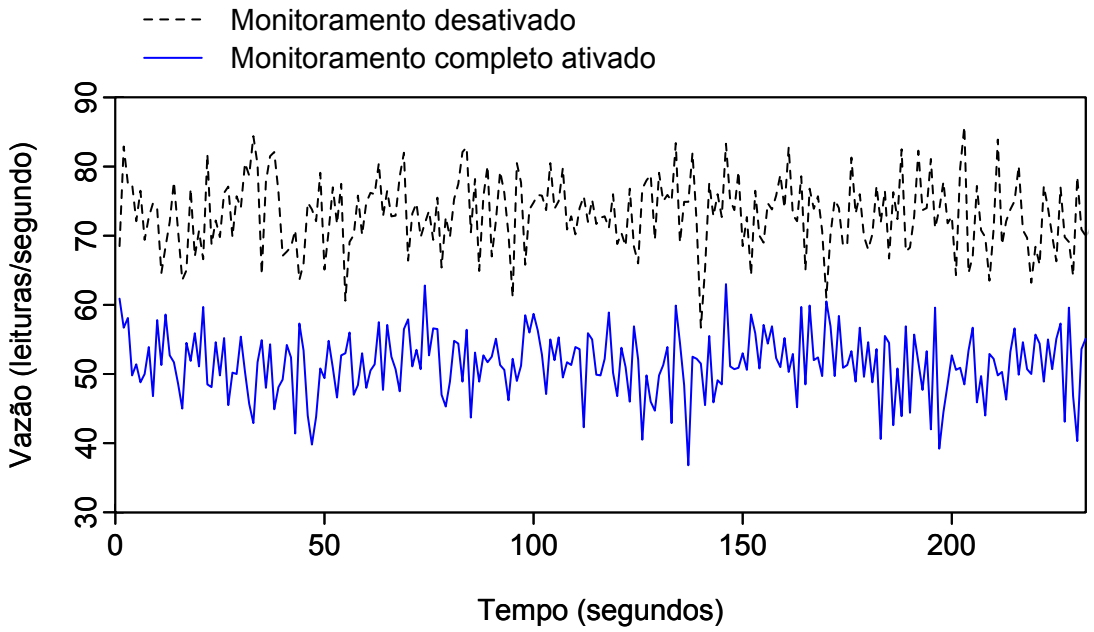


Figura 12: Queda da vazão de operações de leitura intensiva em disco devido a atividade de monitoramento completo do sistema (carga gerada em uma única máquina virtual).

Como é possível observar na Figura 12, a quantidade de operações de leitura realizada é consistentemente menor quando o monitoramento do sistema está ativado. A vazão média com o monitoramento desativado é de 73.17 *operações/s*. Enquanto que a vazão média com o monitoramento ativado é de 51.46 *operações/s*. Essa sobrecarga de monitoramento é de 29.67%. Isso significa que o desempenho da aplicação é aproximadamente 30% menor devido a influência do monitoramento realizado. Essa sobrecarga é referente a utilização de todos os contadores de desempenho disponíveis no sistema.

A Figura 13 retrata a sobrecarga de monitoramento para a execução de aplicações em múltiplas máquinas virtuais. Nos experimentos representados, a infraestrutura foi configurada com: uma única máquina virtual; e duas máquinas virtuais em execução no mesmo servidor hospedeiro. O gráfico de colunas indica os padrões de aplicações, os quais são representados pelas cargas intensivas à memória secundária, processador

e memória primária. O *benchmark* que gera cargas intensivas à memória secundária foi configurado para realizar operações intensivas de: apenas leitura – Disco (Leitura); apenas escrita – Disco (Escrita); e operações mistas com 50% de leitura e 50% de escrita – Disco (Leitura/Escrita). As colunas agrupadas com o rótulo "1 MV" e "2 MV" mostram a sobrecarga de monitoramento das execuções com uma única máquina virtual hospedada e duas máquinas virtuais hospedadas, respectivamente.

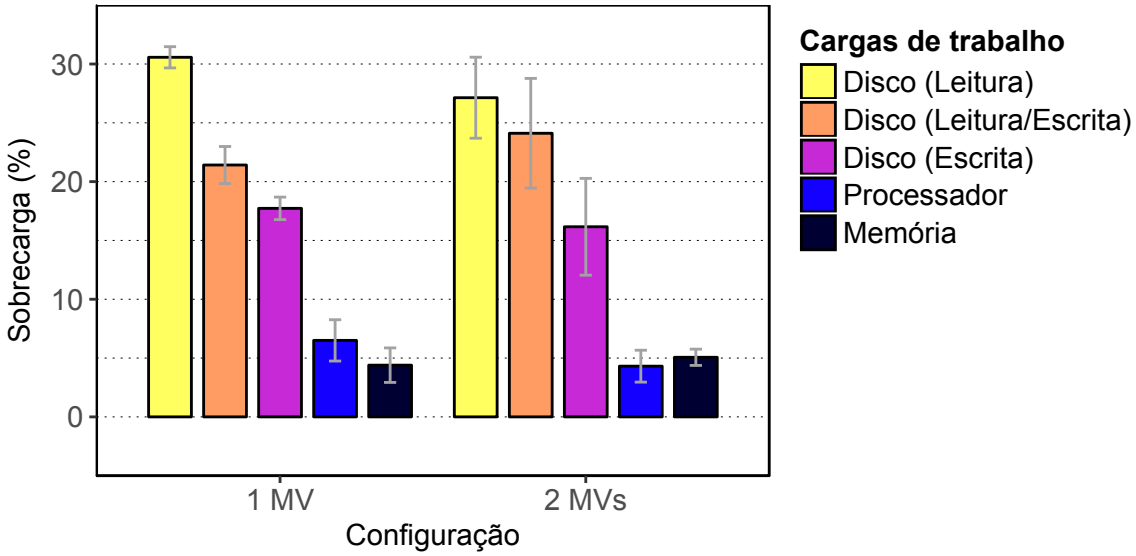
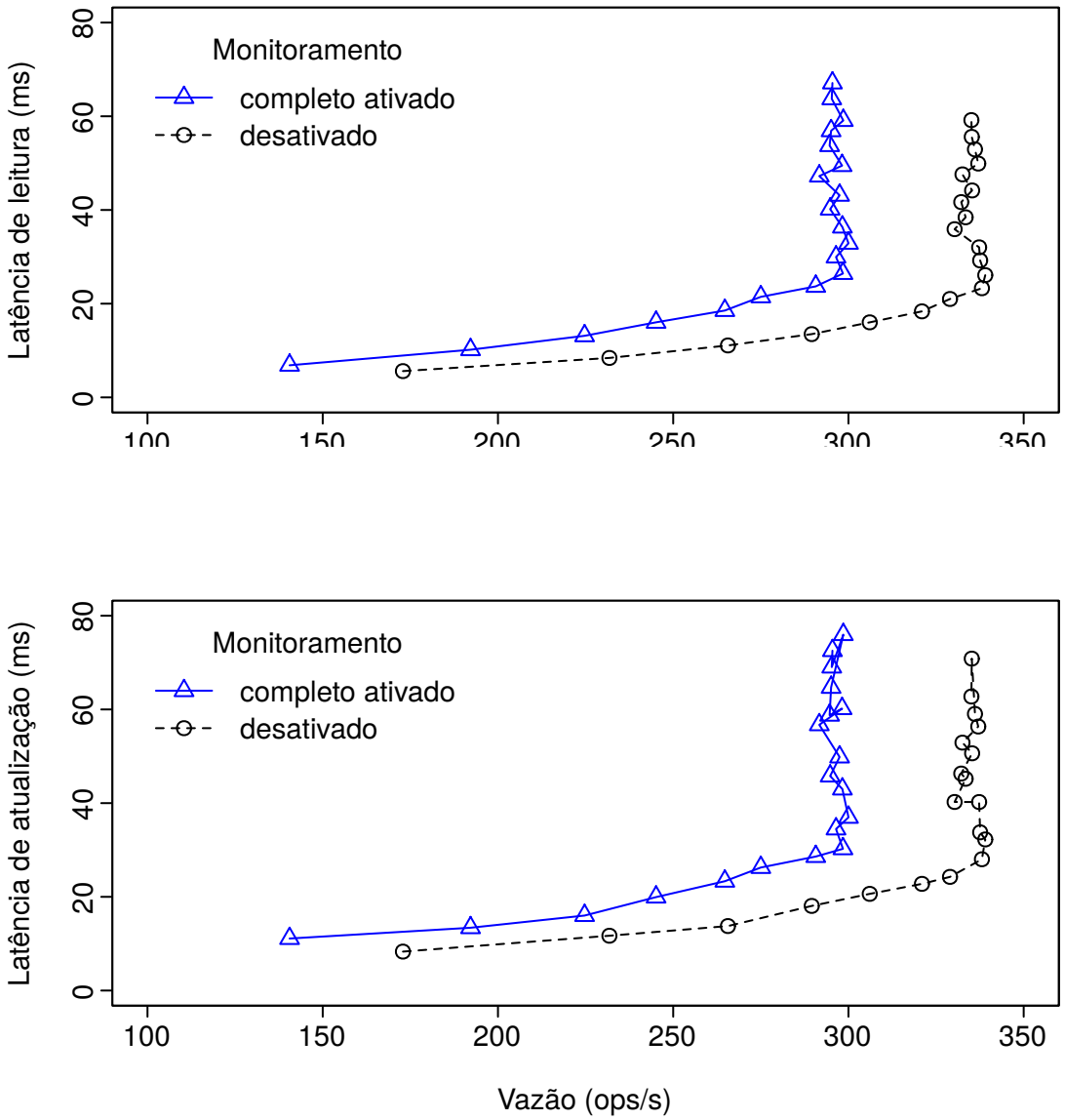


Figura 13: Sobrecarga de monitoramento durante o uso do conjunto completo de contadores de desempenho e de acordo com a quantidade de máquinas virtuais – experimentos com *microbenchmarks*.

De acordo com a ilustração da Figura 13, tanto com uma ou com duas máquinas virtuais em execução, a sobrecarga de monitoramento para cargas intensivas ao processador e à memória principal ficaram entre 4% e 7%. Esses *benchmarks* são menos afetados pelo fato da atividade de monitoramento ser majoritariamente intensiva ao disco. O impacto do monitoramento se torna mais notável quando a aplicação também compete pelo uso do disco, o qual é o caso de carga de trabalho intensiva à memória secundária. Como observado, para operações de apenas leitura a sobrecarga de monitoramento fica em torno de 30%. A sobrecarga média de monitoramento para qualquer um dos tipos de carga à memória secundária não fica abaixo 15% de sobrecarga.

5.3.2 Experimentos com *macrobenchmark*

A Figura 14 contrasta o desempenho de dois tipos de experimentos em relação à atividade de monitoramento do sistema. Os experimentos em questão utilizam a *carga de trabalho b* do *macrobenchmark* YCSB. Cada experimento foi realizado em uma única máquina virtual em momentos diferentes. As Figuras 14a e 14b apresentam gráficos de vazão em relação à latência. No gráfico representado pela Figura 14a estão relacionadas às operações de leitura, no gráfico representado pela Figura 14b, operações de atualização.



(b) Vazão em relação a latência para operações de atualização.

Figura 14: Vazão de operações e respectiva latência gerada em uma única máquina virtual pela *carga de trabalho b*.

As operações de leitura e atualização fazem parte da mesma carga de trabalho. Para cada tipo de operação de um mesmo experimento há 20 pontos no gráfico. Cada ponto representa uma quantidade de *threads* criada na máquina cliente para a geração de carga ao servidor através de requisições destinadas a esse. A quantidade de *threads* inicia em 1 *thread* e é incrementada sequencialmente por 1 unidade, até alcançar 20 *threads*.

Através da Figura 14 é possível evidenciar a existência de sobrecarga devido a atividade de monitoramento. Essa sobrecarga também gera contenção de recursos, afetando o desempenho com acréscimo na latência e menor vazão. Para a *carga de trabalho b* do *benchmark* YCSB essa sobrecarga pode ocasionar a queda de aproximadamente 17% na vazão máxima alcançável.

Embora a Figura 12 e a Figura 14 apresentem resulados para experimentos com cargas que realizam operações intensivas de leitura, a carga de trabalho gerada pelo *microbenchmark* é mais afetada. Isso ocorre pois o *macrobenchmark* YCSB se beneficia de memória cache e da localização dos dados de leitura, referenciados pela distribuição Zipfian.

A Figura 15 apresenta as sobrecargas observadas para cada uma das cargas de trabalho pré-definidas pelo *benchmark* YCSB ao utilizar o monitoramento completo do sistema. As sobrecargas estão descritas por tipo de operação realizada.

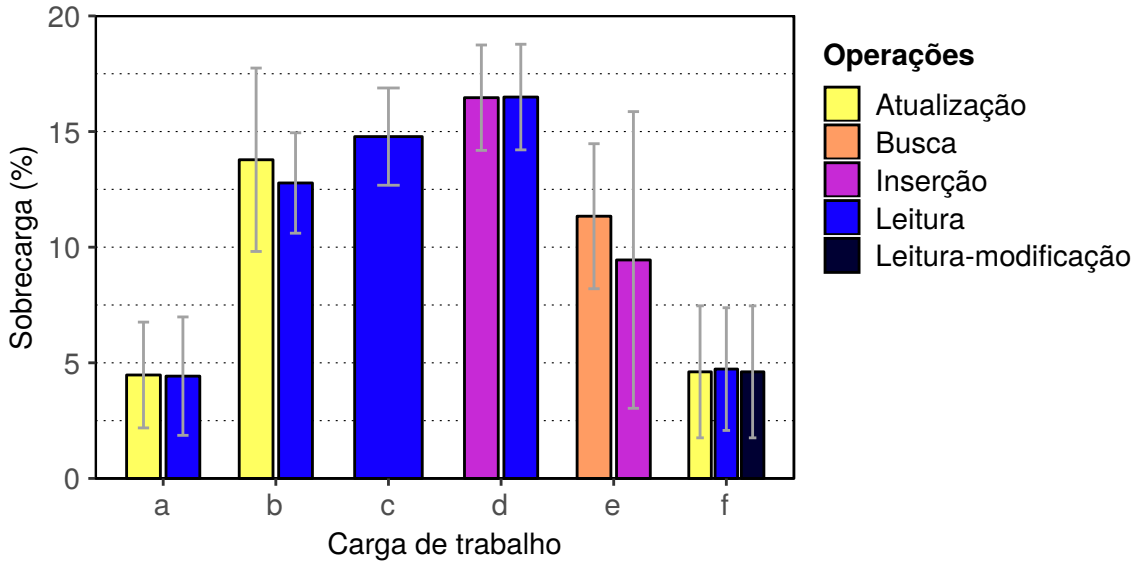


Figura 15: Sobrecarga de monitoramento durante o uso do conjunto completo de contadores de desempenho em uma única máquina virtual – experimentos com *macrobenchmark*.

As cargas de trabalho menos afetadas pela sobrecarga, conforme mostra a Figura 15, foram aquelas com menor taxa de leitura em disco, sendo essas as cargas de trabalho *a* e *f*. Tendo essas uma sobrecarga média próxima de 5%. Em contra partida, as cargas de trabalho mais afetadas pela contenção de recursos foram as que possuem maior atividade de leitura em disco. Novamente, a competição pela realização de operações em disco entre os *benchmarks* e o monitoramento do sistema trouxe perdas de desempenho. As sobrecargas médias para essas cargas de trabalho – *b*, *c*, *d* e *e* – ficaram entre 9% e 17%. Os valores máximos de sobrecarga alcançaram aproximadamente 20%.

5.4 Redução de sobrecarga

Após reduzir a quantidade de contadores de desempenho através da técnica proposta, foram reproduzidos os mesmos cenários de teste descritos na Seção 5.3. As subseções seguintes contrastam as sobrecargas encontradas entre a utilização do conjunto completo de contadores de desempenho e o conjunto reduzido de contadores encontrado através do emprego da técnica.

5.4.1 Experimentos com *microbenchmark*

A Figura 16 ilustra o resultado da vazão de experimentos com carga de trabalho intensiva de operações de leitura em disco. Os cenários utilizados possuem uma única máquina virtual hospedada. O gráfico representado pela Figura 16b apresenta, através da linha sólida, a série temporal para a vazão de operações de *leitura/s* de experimento utilizando o conjunto representativo de contadores de desempenho. Esse conjunto foi encontrado através da técnica proposta para o monitoramento do sistema. Para contrastar com esse novo resultado, a Figura 16a reproduz o resultado apresentado na Figura 12.

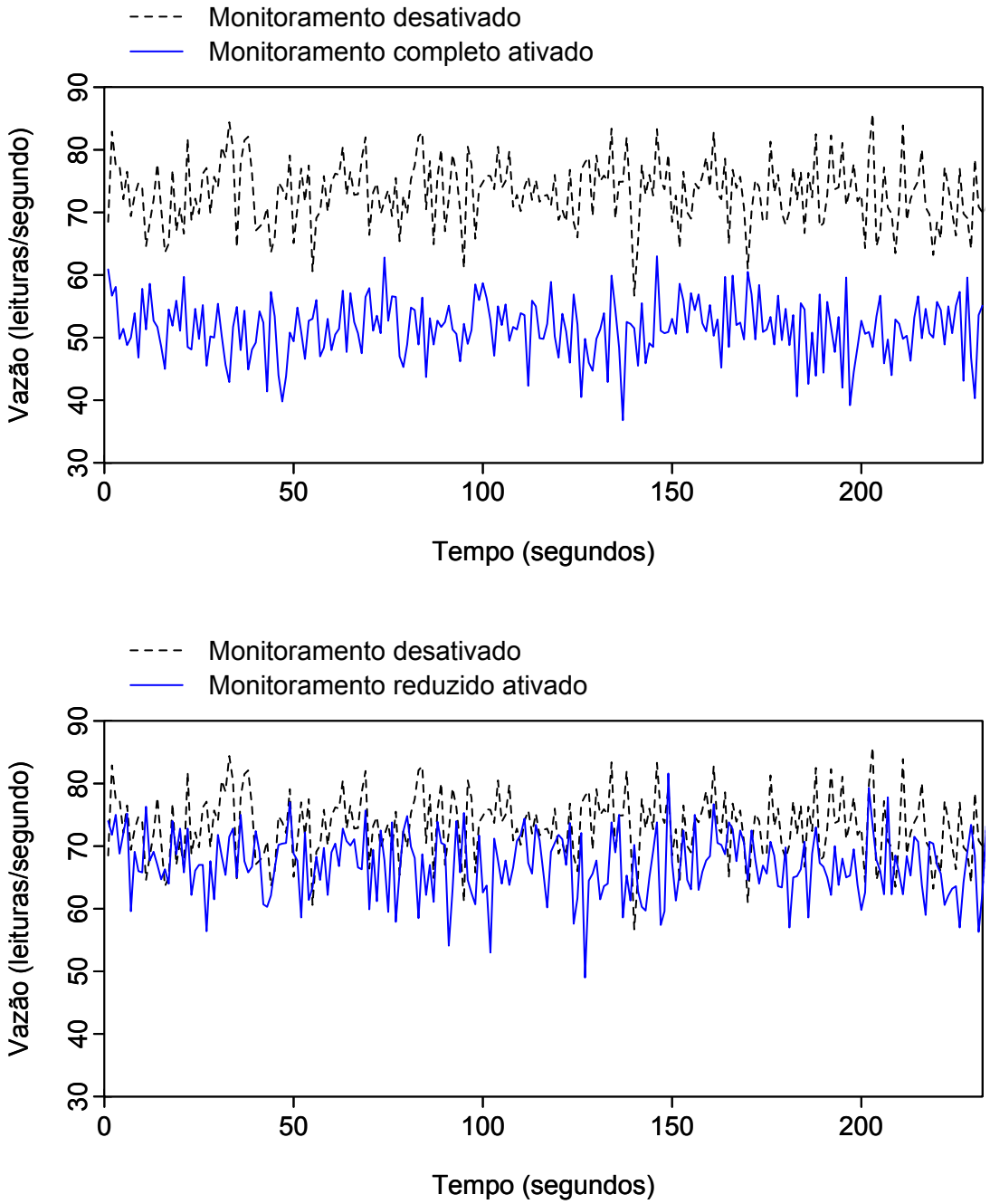
Como é possível observar no gráfico, representado pela Figura 16b, a sobrecarga causada pelo monitoramento é quase imperceptível. A vazão média quando o monitoramento está desativado é de 73.17 *operações/s*. Quando o monitoramento está ativado a vazão média alcançada é de 67.08 *operações/s*. Este resultado representa uma importante redução na sobrecarga de monitoramento, desde que o desempenho da aplicação era de 51.46 *operações/s* antes da seleção de contadores. A sobrecarga com o monitoramento reduzido ativado é de 8.32% contra os 29.67% de sobrecarga com o monitoramento completo ativado. Essa redução de sobrecarga representa uma redução de 71.95% de sobrecarga devido a atividade de monitoramento do sistema.

O gráfico de colunas ilustrado pela Figura 17 compara a sobrecarga causada pelo uso de todos os contadores de desempenho com a sobrecarga causada pelo uso do conjunto reduzido de contadores. As colunas com borda sólida representam a sobrecarga causada pelo conjunto completo de contadores. E as colunas com borda tracejada representam a sobrecarga causada pelo conjunto reduzido de contadores. É possível perceber que o uso do conjunto de contadores encontrado pela técnica proposta reduz consistentemente a sobrecarga de monitoramento para todos os padrões de cargas de trabalho. Testes de hipótese unilateral [91] indicam com 95% de confiança que ao utilizar o conjunto reduzido de contadores de desempenho a sobrecarga gerada é menor quando comparada com a gerada pela utilização do conjunto completo de contadores.

Cargas intensivas ao processador e à memória primária apresentam sobrecarga desprezível quando geradas junto a utilização do monitoramento reduzido. A maior sobrecarga observada pelo monitoramento otimizado é perceptível para carga intensiva de leitura em disco, alcançando 10% de sobrecarga. No entanto, esta sobrecarga corresponde a apenas um terço da sobrecarga em cenário equivalente sob monitoramento completo.

5.4.1.1 Experimentos com macrobenchmark

A Figura 18 contrasta a vazão em relação à latência para operações de leitura realizadas pelo sistema quando não submetido a atividade monitoramento - estimulado pela *carga de trabalho b* - com: a vazão em relação a latência de operações realizadas quando o sistema está submetido ao monitoramento completo, Figura 18a; e a vazão de operações realizadas quando o sistema está submetido ao monitoramento reduzido, Figura 18b, de-



(b) Queda da vazão devido a atividade de monitoramento reduzido.

Figura 16: Contraste entre perfis de monitoramento em relação a queda da vazão de operações de leitura intensiva em disco gerada em uma única máquina virtual.

terminado pela aplicação da técnica proposta. Para ambos os gráficos representados pelas figuras, o desempenho do sistema sob monitoramento está registrado pela linha contínua, e o registro de desempenho do sistema sem monitoramento está registrado pela linha tracejada. O mesmo contraste de desempenho, no entanto para operações de atualização também realizadas pela *carga de trabalho b*, está representado pela Figura 19.

Através dos resultados experimentais apresentados pelas Figuras 18 e 19 evidencia-se diferenças no desempenho do sistema quando submetidos aos diferentes perfis de mo-

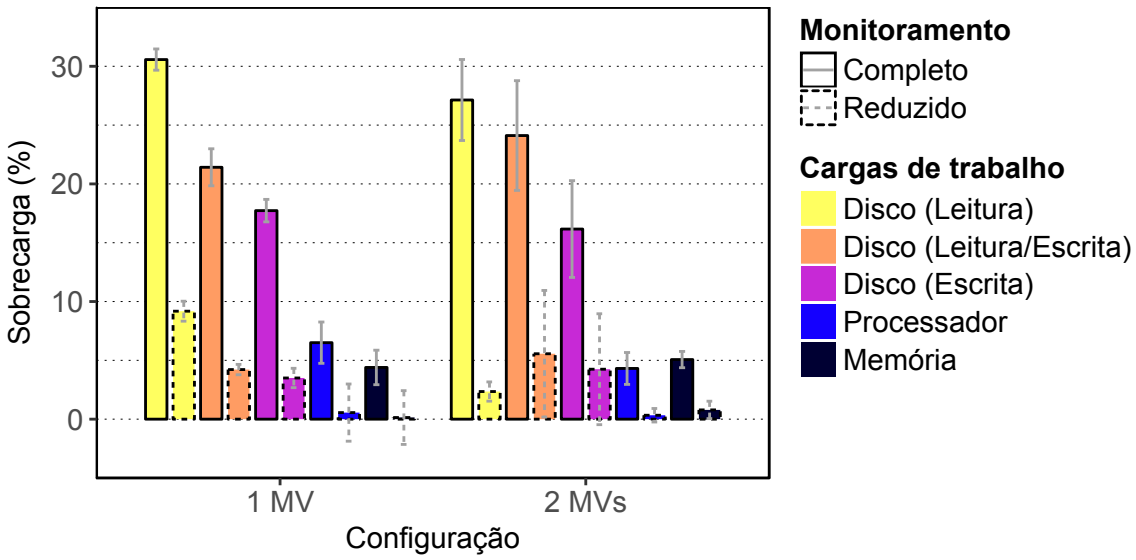
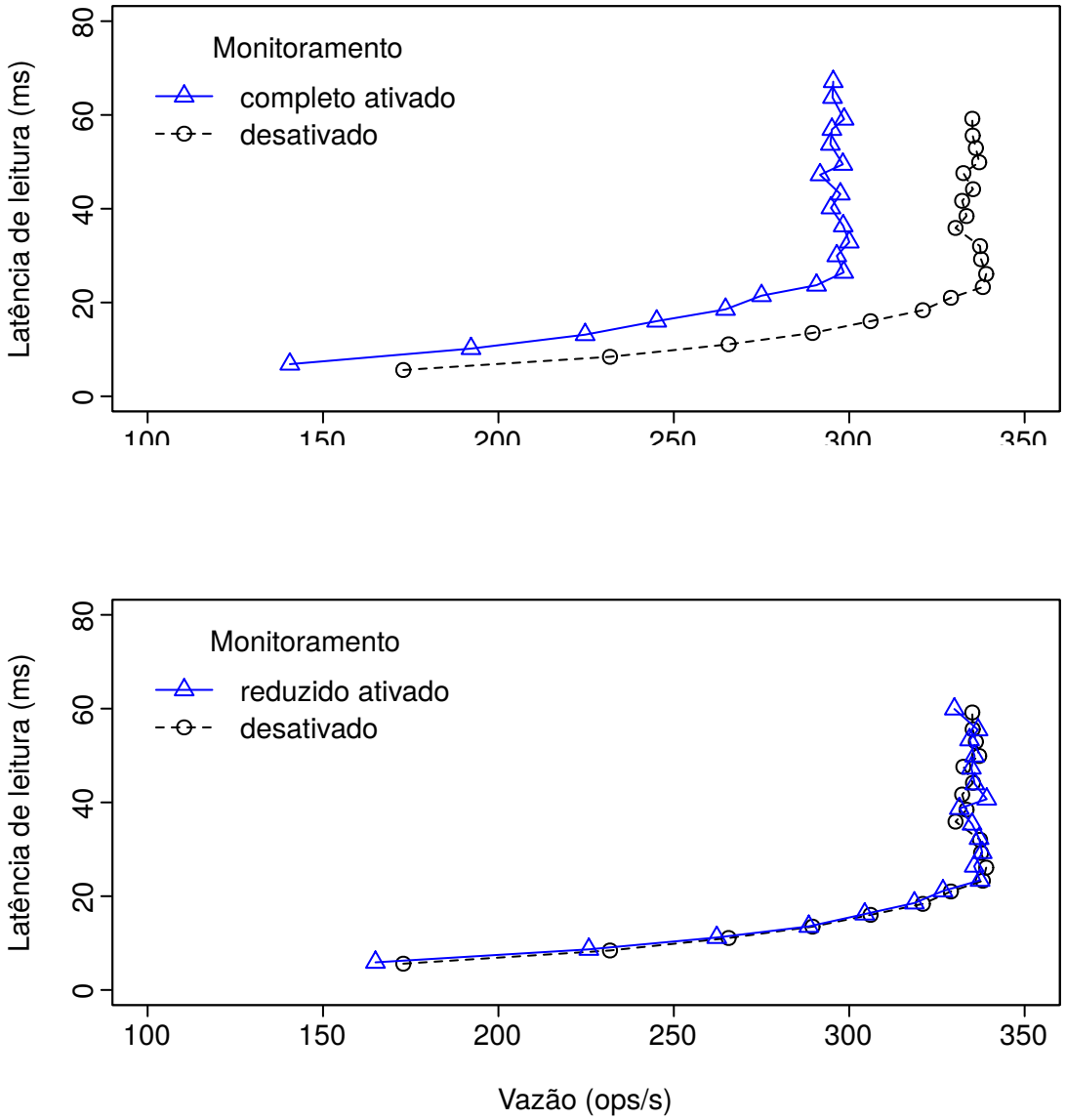


Figura 17: Contraste entre sobrecargas de monitoramento geradas pelo uso de diferentes conjuntos de contadores de desempenho e de acordo com a quantidade de máquinas virtuais.

nitoramento aplicados. Quando o sistema está submetido ao monitoramento completo, tanto a vazão quanto a latência de operações são consideravelmente piores comparado com a vazão e latência do sistema não submetido a atividade de monitoramento. Testes de hipótese unilateral [91] indicam com 95% de confiança que ao utilizar o conjunto reduzido de contadores de desempenho a sobrecarga gerada é menor quando comparada com a gerada pela utilização do conjunto completo de contadores. Considerando o primeiro ponto de cada uma das linhas dos gráficos representados pelas Figuras 18a e 19a, é observado: uma queda de 18.7% na vazão, um aumento de 22.29% na latência para operações de leitura, e um aumento de 33.58% na latência para operações de atualização, decorrentes da atividade de monitoramento.

Em contraste com essa interferência, é possível observar nas Figuras 18b e 19b que tanto a vazão quanto a latência de operações obtida no sistema com monitoramento reduzido são estreitamente próximas aos valores observados quando o sistema não está submetido a atividade de monitoramento. Nesse caso, para o primeiro ponto de cada uma das linhas, observa-se uma redução na interferência de desempenho em: 75.5% para vazão; 77.21% para latência em operações de leitura; e 96.36% para latência em operações de atualização.

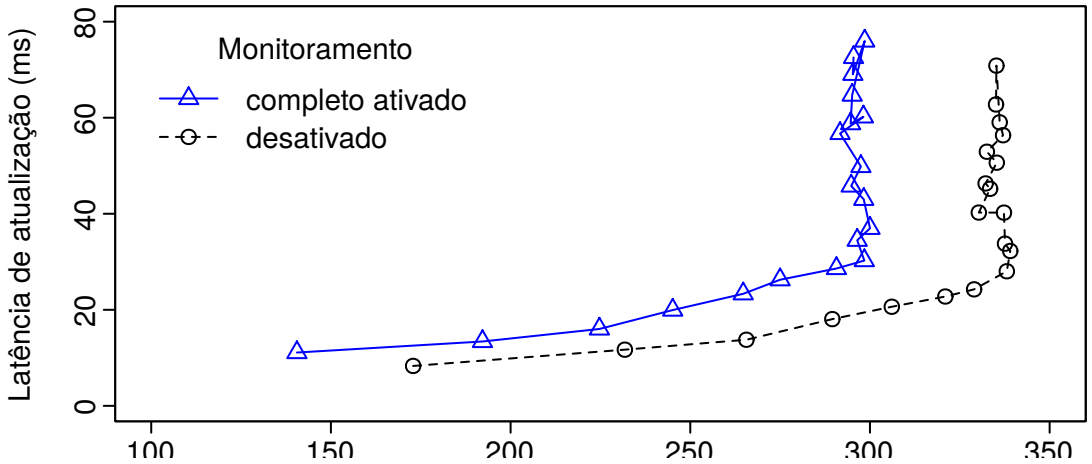
A Figura 20 compara as sobrecargas observadas em ambientes com diferentes perfis de monitoramento – completo e reduzido. As comparações são realizadas para cada uma das cargas de trabalho – *a*, *b*, *c*, *d* e *e*. Em cada comparação é detalhado o tipo de operação envolvida na carga de trabalho – atualização, busca, inserção, leitura e/ou leitura-modificação. Conforme ilustrado pelo gráfico de colunas, presente na Figura 20, as sobrecarga mais elevadas devido à atividade de monitoramento completo se aproxima-

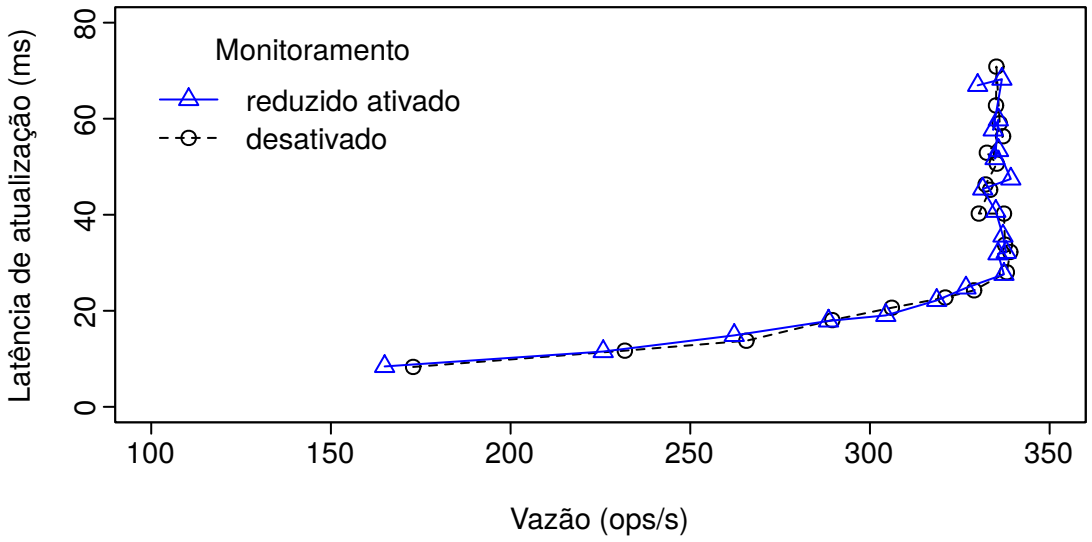


(b) Queda da vazão e aumento da latência devido a atividade de monitoramento reduzido.

Figura 18: Contraste de perfis de monitoramento em relação a queda da vazão e aumento da latência para operações de leitura gerada em uma única máquina virtual pela *carga de trabalho b*.

ram de 20% de sobrecarga. As cargas de trabalho com as sobrecargas mais elevadas – *b*, *c*, *d*, – são as que realizam maior quantidade de operações de leitura. A percentagem de operações de leitura para essas cargas é de 95% ou mais. As sobrecargas obtidas pela aplicação do monitoramento com o conjunto de contadores de desempenho encontrado pela técnica proposta não ultrapassaram, em grande maioria, 2.5% de sobrecarga. Os piores casos de sobrecarga ficaram abaixo de 5% de sobrecarga. Em média as sobrecargas ficam em torno de 0.77%.





(b) Queda da vazão e aumento da latência devido a atividade de monitoramento reduzido.

Figura 19: Contraste de perfis de monitoramento em relação a queda da vazão e aumento da latência para operações de atualização gerada em uma única máquina virtual pela *carga de trabalho b*.

5.5 Avaliação do conjunto de contadores

Os conjuntos de contadores de desempenho utilizados para cada um dos experimentos realizados com *microbenchmarks*, com o monitoramento reduzido, foram obtidos separadamente pela aplicação de uma única iteração do método apresentado no Capítulo 4 para a seleção de contadores de desempenho para o monitoramento de sistemas. Especificamente para este tipo de experimento, com *microbenchmark*, não houve iterações consecutivas para refinamento dos agrupamentos encontrados. Dessa forma as aplicações realizam os mesmos tipos de instruções, ou seja, não há dependência de um usuário para definir o tipo de comportamento imposto em relação a utilização de recursos. A Tabela

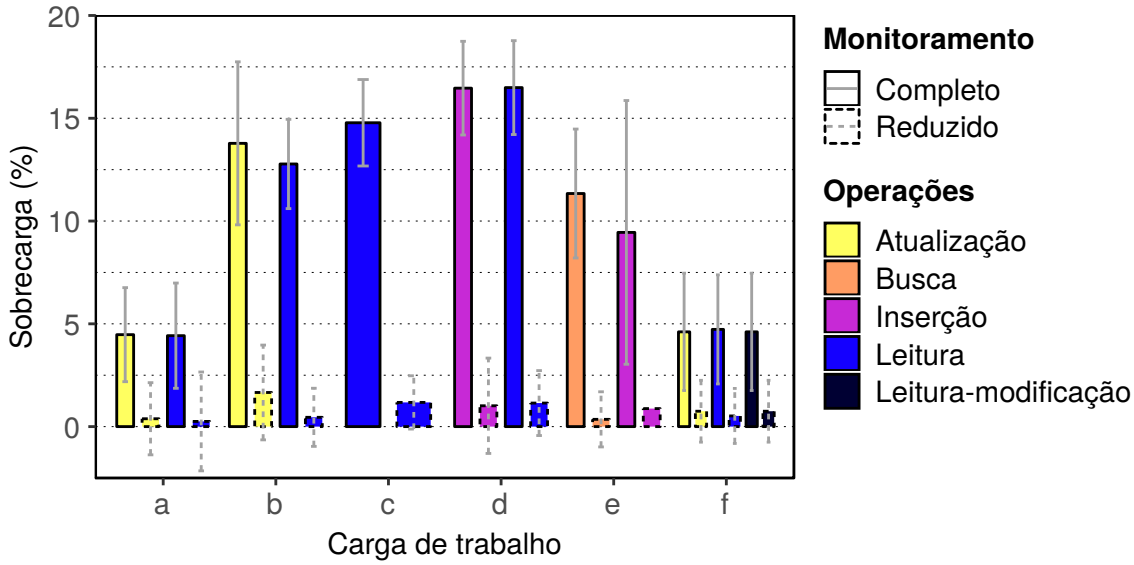


Figura 20: Contraste entre sobrecargas de monitoramento geradas pelo uso de conjunto completo e reduzido de contadores de desempenho – experimentos com *macrobenchmark*.

11 apresenta a quantidade de contadores de desempenho selecionados para cada um dos *benchmarks* e a respectiva percentagem que essa quantidade representa com relação a quantidade total de contadores disponíveis por padrão no sistema operacional.

Tabela 11: Resumo da aplicação do método proposto em registros de monitoramento do sistema durante execução de cargas de trabalho geradas por *microbenchmaks*.

Análise	**Iteração**	**Carga de trabalho**	**Grupos formados**	**Percentagem**
A0	I1	Disco (Leitura)	356	3.91%
A1	I1	Disco (Leitura/Escrita)	360	3.95%
A2	I1	Disco (Escrita)	354	3.88%
A3	I1	Processador	348	3.82%
A4	I1	Memória	352	3.86%

Os experimentos realizados com *macrobechmarks*, com o monitoramento reduzido, utilizaram um único conjunto de contadores de desempenho representativo em comum. Esse conjunto de contadores foi obtido aplicando o método de forma iterativa sobre os registros de monitoramento do sistema obtidos a partir do monitoramento de cada uma das cargas pré-definidas pelo *benchmark* YCSB. A aplicação do método se faz necessária desse modo pois o comportamento de utilização de recursos é dependente da interação que os usuários possuem com o sistema. Logo, o conjunto de contadores de desempenho utilizado deve satisfazer os diferentes estados de utilização de recursos demandados ao sistema. A Tabela 12 apresenta em sequência a quantidade de contadores de desempenho encontrados após cada iteração do método proposto, juntamente com a percentagem que a quantidade de contadores selecionados representa do total de contadores de desempenho

disponíveis por padrão no sistema operacional.

Tabela 12: Resumo da aplicação do método proposto em registros de monitoramento do sistema durante execução de cargas de trabalho geradas por *macrobenchmaks*.

Análise	Iteração	Carga de trabalho	Grupos formados	Percentagem
A5	I1	Carga de trabalho a	423	4.64%
A5	I2	Carga de trabalho b	627	6.88%
A5	I3	Carga de trabalho c	733	8.05%
A5	I4	Carga de trabalho d	944	10.36%
A5	I5	Carga de trabalho e	1124	12.34%
A5	I6	Carga de trabalho f	1160	12.73%

Como um meio de tornar o conjunto representativo de contadores de desempenho mais robusto a mudanças de demanda, foram adicionados às iterações os registros de monitoramento do sistema quando submetido a cargas geradas pelos *microbenchmarks*. Dessa forma as cargas geradas pelos *microbenchmarks* são capazes de estimular os componentes do sistema em mais níveis de intensidade do que quando comparado com as cargas de trabalho utilizadas pelo *macrobenchmark*. A Tabela 12 apresenta os resultados das iterações como apresentado na Tabela 11.

Tabela 13: Resumo da aplicação do método proposto em registros de monitoramento do sistema durante execução de cargas de trabalho geradas por *macrobenchmaks* e *microbenchmaks*.

Análise	Iteração	Carga de trabalho	Grupos formados	Percentagem
A5	I7	Disco (Leitura)	1291	14.17%
A5	I8	Disco (Leitura/Escrita)	1333	14.63%
A5	I9	Disco (Escrita)	1366	14.99%
A5	I10	Processador	1418	15.57%
A5	I11	Memória	1437	15.77%

Após o término da iteração *I11* da análise *A5*, descrita pela Tabela 12 e pela Tabela 13, foram obtidos os grupos de contadores de desempenho que originam os contadores de desempenho representativos.

5.6 Avaliação qualitativa

Como meio de avaliar o conjunto de contadores de desempenho representativo encontrado pelo método proposto, foi realizado um levantamento de contadores de desempenho considerados importantes para o monitoramento de sistemas. Em [16] são apresentados contadores de desempenho e os respectivos problemas de desempenho que esses contadores podem indicar. As informações apresentadas no relatório técnico indicado são baseadas na experiência de profissionais e no conhecimento do funcionamento de sistemas. Ao

total, são indicados e descritos 28 contadores de desempenho disponibilizados pelo sistema operacional. Em nenhum momento o relatório técnico indica os demais contadores de desempenho não relatados como desnecessários ou pouco importantes.

Os contadores de desempenho indicados pelo relatório técnico são: *% Processor Time Counter*; *% Privileged Time Counter*; *% Interrupt Time Counter*; *Processor Queue Length Counter*; *Context Switches Counter*; *System Up Time Counter*; *Available Bytes Counter*; *Working Set Counter*; *Pages/sec Counter*; *Page Reads/sec Counter*; *Pool Non-paged Bytes Counter*; *Paged Pool Bytes Counter*; *Paged Pool Failures Counter*; *Cache Bytes Counter*; *System Cache Resident Bytes Counter*; *Committed Bytes Counter*; *Avg. Disk secs/transfer Counter*; *% Idle Time Counter*; *Disk Transfers/sec Counter*; *Avg. Disk Queue Length Counter*; *Split IO/sec Counter*; *Free Megabytes Counter*; *Bytes Total/sec Counter*; *Server Bytes Total/sec*; *Datagrams/sec Counter*; *Connections Established Counter*; *Segments Received/sec Counter*; *% Interrupt Time Counter*.

Após a realização de nossos experimentos, esses contadores de desempenho foram verificados no conjunto de contadores de desempenho representativo e no conjunto de contadores de desempenho representados, ou seja, aqueles que não foram selecionados mas estão representados por algum contador de desempenho representativo.

Os únicos contadores de desempenho não selecionados pela técnica proposta foram: *Committed Bytes Counter*; *Paged Pool Failures Counter*; e *Server Bytes Total/sec*. Todos os demais contadores de desempenho foram contemplados pela aplicação do método proposto para a seleção de contadores de desempenho para o monitoramento do sistema.

6 CONSIDERAÇÕES FINAIS

A adoção de ambientes de nuvem computacional para a implantação de serviços segue em expansão. Os benefícios oferecidos por essa abordagem são importantes para as decisões de negócio. A tecnologia de virtualização de servidores provê flexibilidade para a alocação de recursos sob demanda. Com isso, possui papel fundamental no gerenciamento de recursos utilizados e manutenção de requisitos de desempenho. Em adição, proporciona aos provedores de serviços em nuvem ofertar infraestruturas ou plataformas virtualizadas com as características que o cliente necessita, tanto em termos de *software* quanto de *hardware*. Ainda assim, a alocação de máquinas virtuais exige atenção devido a possíveis interferências de desempenho, seja pela existência de aplicações co-alocadas ou co-agendadas em recursos de uso comum. Como medida preventiva, é necessário realizar um balanceamento entre o desempenho e a utilização de recursos durante a virtualização de sistemas.

Monitores de desempenho, através das informações coletadas de contadores de desempenho, são largamente utilizados com o objetivo de assistir o desempenho de sistemas. Essas ferramentas são essenciais para o suporte ao balanceamento de carga, realocação de recursos, migração de máquinas virtuais, geração de alertas através do disparo de gatilhos e tomada de decisões. No entanto, o monitoramento por si só pode degradar o desempenho de sistemas, especialmente em ambientes virtualizados. A utilização intensiva do monitoramento acarreta na contenção de recursos computacionais, ou seja, restringe a utilização desses pelas demais aplicações. Nesse caso, ocorre co-alocação do processo de monitoramento com as aplicações em execução. Todavia, a atividade de monitoramento não pode ser negligenciada. A identificação de anomalias de desempenho no sistema pode indicar situações comprometedoras à segurança e à manutenção da disponibilidade do serviço monitorado.

Esse trabalho analisa a sobrecarga de desempenho causada pelo monitoramento de infraestrutura virtualizada. Diferentes padrões de aplicações foram avaliados e resultados experimentais revelaram contenção de recursos significante introduzida pela atividade de monitoramento. Aplicações que realizam operações intensivas de I/O são as mais afetadas, especialmente aquelas com maior proporção de operações de leitura. Através

de experimentos realizados com *microbenchmarks* caracterizou-se a contenção de recursos ocasionados pela atividade de monitoramento. Com características similares as de aplicações de HPC, em termos de operações intensivas e repetitivas, experimentos com os *microbenchmarks* apresentaram as maiores taxas de sobrecarga devido a interferência de desempenho. A contenção de recursos provocada em cenários realistas foi demonstrada com a utilização de *macrobenchmark* capaz de reproduzir diversos comportamentos de usuários pela geração de requisições a sistema gerenciador de banco de dados.

Nesse trabalho propomos uma abordagem capaz de reduzir a sobrecarga de monitoramento. A redução da sobrecarga está baseada na redução da atividade de monitoramento, obtida pela redução de dados coletados e armazenados pelo sistema. Nossa abordagem usa estratégias de mineração de dados para selecionar apenas os contadores de desempenho mais relevantes para diferentes cargas de trabalho geradas por aplicações. Nós aplicamos uma análise de agrupamento hierárquico para encontrar um conjunto relevante de contadores de desempenho. Com a utilização do conjunto de contadores de desempenho selecionado, a sobrecarga de monitoramento é reduzida drasticamente. Em alguns casos, em experimentos com *microbenchmarks*, a sobrecarga pode ser dois terços menor do que a observada com o conjunto completo de contadores de desempenho. Já para para experimentos com *macrobenchmarks*, a redução de sobrecarga é ainda maior e pode alcançar cinco sextos.

A abordagem para seleção de contadores de desempenho proposta mostrou-se capaz de selecionar contadores de desempenho considerados importantes em documentação técnica sobre o tema. Quando não selecionado algum desses contadores, um contador representativo se faz presente no conjunto de contadores utilizados pela nossa técnica para o monitoramento do sistema. Essa característica indica a capacidade de selecionar contadores de desempenho importantes para a análise do estado do sistema de forma automática. Esse recurso, além de auxiliar na seleção de contadores e na liberação de recursos contidos pela atividade de monitoramento, também viabiliza o início de trabalhos que tenham como objetivo a análise e interpretação automática dos dados coletados.

Trabalhos com essa iniciativa são importantes para análise e interpretação dos valores obtidos pelo contadores de desempenho em tempo de execução. Possibilita, por exemplo, a geração de alertas de situações indesejadas e o disparo de medidas preventivas no gerenciamento de recursos alocados. Podendo essas, anteciparem a solução de problemas de desempenho iminentes. Com o suporte da nossa técnica proposta, além de reduzir a sobrecarga gerada pela atividade de monitoramento, essa se torna menos intrusiva aos valores observados nos dados coletados e que devem descrever os diferentes estados de utilização de recursos do sistema em produção, além de gerar uma menor quantidade de dados a ser analisada. É desejável que a interferência de desempenho gerada pela atividade de monitoramento seja mínima. Porém, informações importantes sobre mudanças de estado do sistema devem ser mantidas e analisadas.

6.1 Trabalhos futuros

Como trabalho futuro pretende-se explorar o tema tratado nessa dissertação sob diferentes perspectivas. Quanto a avaliação de sobrecarga, esta pode ser avaliada em cenários onde os dados são enviados através da rede, tanto para o sistema monitorado quanto para o servidor de monitoramento. Dessa forma, seria possível visualizar os principais gargalos de sistemas de monitoramento remoto e revelar os principais requisitos de dimensionamento desses sistemas, além de revelar os ganhos obtidos pela redução do conjunto de contadores de desempenho utilizados para o monitoramento de sistemas nesses cenários. Em relação a técnica proposta, essa pode ser aprimorada ou novas abordagens podem ser empregadas para o desenvolvimento de métodos para a seleção de contadores de desempenho. Existem muitas técnicas de descoberta de conhecimento em banco de dados e várias podem ser aplicadas para a solução de um mesmo problema.

Além daquilo explorado, é possível avançar o estudo sobre contadores de desempenho de outras maneiras. Uma delas é através do correlacionamento de contadores de desempenho implementados em diferentes camadas do sistema, como por exemplo, contadores implementados em *hardware* com contadores implementados no sistema operacional. Dessa forma, seria possível selecionar aqueles que acarretam em menor sobrecarga ao sistema. Outra possibilidade é a revelação de equações envolvendo contadores de desempenho de forma a descrever o comportamento do sistema, contribuindo para uma redução ainda maior dos contadores necessários ao monitoramento. Essas equações também podem auxiliar na antecipação de situações indesejadas de utilização de recursos através da análise de dados de monitoramento em tempo real. Com o mesmo propósito podem ser desenvolvidos métodos de aprendizado de máquina, por exemplo.

REFERÊNCIAS

[1] Intel. *Intel 64 and ia-32 architectures software developerś manual, volume 3b: System programming guide, part 2*, 2016.

[2] IBIDUNMOYE, O.; HERNANDEZ-RODRIGUEZ, F.; ELMROTH, E. Performance anomaly detection and bottleneck identification. *ACM Computing Surveys (CSUR)*, v. 48, n. 1, p. 4, 2015.

[3] SYER, M. D.; JIANG, Z. M.; NAGAPPAN, M.; HASSAN, A. E.; NASSER, M.; FLORA, P. Leveraging performance counters and execution logs to diagnose memory-related performance issues. In: . 29th IEEE International Conference on Software Maintenance, 2013. p. 110–119.

[4] FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. *AI magazine*, v. 17, n. 3, p. 37, 1996.

[5] COOPER, B. F.; SILBERSTEIN, A.; TAM, E.; RAMAKRISHNAN, R.; SEARS, R. Benchmarking cloud serving systems with ycsb. In: . 1st ACM symposium on Cloud computing, 2010. p. 143–154.

[6] Core workloads. https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads.

[7] STRAWN, G.; STRAWN, C. Moore's law at fifty. *IT Professional*, v. 17, n. 6, p. 69–72, Nov 2015.

[8] MUSTAFA, S.; NAZIR, B.; HAYAT, A.; UR REHMAN KHAN, A.; MADANI, S. A. Resource management in cloud computing: Taxonomy, prospects, and challenges. *Computers & Electrical Engineering*, v. 47, p. 186 – 203, 2015.

[9] SPRUNT, B. The basics of performance-monitoring hardware. *IEEE MICRO*, v. 22, n. 4, p. 64–71, 2002.

[10] ERANIAN, S. What can performance counters do for memory subsystem analysis? In: . ACM SIGPLAN Workshop on Memory Systems Performance and Correctness, 2008. ACM. p. 26–30.

[11] OGASAWARA, T. Stall reducing method, device and program for pipeline of processor with simultaneous multithreading function, Feb. 13 2014. US Patent App. 13/965,562.

[12] CHEN, D.; VACHHARAJANI, N.; HUNDT, R.; LIAO, S.-W.; RAMASAMY, V.; YUAN, P.; CHEN, W.; ZHENG, W. Taming hardware event samples for fdo compilation. In: . ACM international symposium on Code generation and optimization, 2010. p. 42–52.

[13] JACOBI, E. B.; POPIOLEK, P. F.; HAX, V. A.; CARVALHO, J. T.; DUARTE FILHO, N. L.; MENDIZABAL, O. M. O modelo de computação em nuvem e sua aplicabilidade. *Revista Jr de Iniciação Científica em Ciências Exatas e Engenharia*, v. 2, n. 1, p. 1–10, 2012.

[14] GHOSH, R.; LONGO, F.; NAIK, V. K.; TRIVEDI, K. S. Modeling and performance analysis of large scale IaaS clouds. *Future Generation Computer Systems*, v. 29, n. 5, p. 1216 – 1234, 2013. Special section: Hybrid Cloud Computing.

[15] SHANG, W.; HASSAN, A. E.; NASSER, M.; FLORA, P. Automated detection of performance regressions using regression models on clustered performance counters. In: . 6th ACM/SPEC International Conference on Performance Engineering, 2015. p. 15–26.

[16] HP. Hp performance engineering best practices series. Technical report, 2010.

[17] POPIOLEK, P. F.; DOS SANTOS MACHADO, K.; MENDIZABAL, O. M. Redução de sobrecarga gerada pelo uso de contadores de desempenho em ambientes virtualizados. *XVIII Escola Regional de Alto Desempenho, 2018*.

[18] POPIOLEK, P. F.; MACHADO, K. S.; MENDIZABAL, O. M. Reducing monitoring overhead in virtualized environments through feature selection. In: . XXXVI Simpósio Brasileiro de Redes de Computadores (SBRC), 2018.

[19] POPIOLEK, P. F.; DOS SANTOS MACHADO, K.; MENDIZABAL, O. M. Identificação de custos computacionais causados por contadores de desempenho. *IBERCHIP 2017*.

[20] PRODANOV, C. C.; DE FREITAS, E. C. *Metodologia do trabalho científico: Métodos e técnicas da pesquisa e do trabalho acadêmico-2ª edição*. Editora Feevale, 2013.

[21] BARA, L.; BONCALO, O.; MARCU, M. Hardware support for performance measurements and energy estimation of OpenRISC processor. In: . Applied Computational Intelligence and Informatics, 2015. p. 399–404.

[22] TANENBAUM, A. S.; WOODHULL, A. S. *Sistemas operacionais: Projetos e implementação*. Bookman Editora, 2009.

[23] HENNESSY, J. L.; PATTERSON, D. A. *Computer architecture: a quantitative approach*. Elsevier, 2011.

[24] JIANG, Z. M.; HASSAN, A. E.; HAMANN, G.; FLORA, P. Automated performance analysis of load tests. In: . IEEE International Conference on Software Maintenance, 2009. p. 125–134.

[25] BITZES, G.; NOWAK, A. The overhead of profiling using PMU hardware counters. *CERN openlab report*, 2014.

[26] NOWAK, A.; YASIN, A.; MENDELSON, A.; ZWAENEPOEL, W. Establishing a base of trust with performance counters for enterprise workloads. In: . USENIX Annual Technical Conference, 2015. p. 541–548.

[27] Performance counters on intel(r) processors. http://qcd.phys.cmu.edu/.

[28] Model specific registers and functions. http://datasheets.chipdb.org/Intel/x86.

[29] Perfmon2. http://perfmon2.sourceforge.net/.

[30] Oprofile. http://oprofile.sourceforge.net/news/.

[31] DEMME, J.; SETHUMADHAVAN, S. Rapid identification of architectural bottlenecks via precise event counting. In: . v. 39 of *ACM SIGARCH Computer Architecture News, 2011*. p. 353–364.

[32] Java 2 platform, enterprise edition (j2ee) overview. https://www.oracle.com/technetwork/java/javaee/appmodel-135059.html.

[33] .net core guide. https://docs.microsoft.com/en-us/dotnet/core/index.

[34] Java management extensions (jmx) technology. https://www.oracle.com/technetwork/articles/java/javamanagement-140525.html.

[35] Apache module mod_status. https://httpd.apache.org/docs/2.4/mod/mod_status.html.

[36] Performance counters functions. https://docs.microsoft.com/en-us/windows/desktop/perfctrs/performance-counters-functions.

[37] Oracle9i database performance tuning guide and reference. https://docs.oracle.com/cd/B10501_01/server.920/a96533/sqltrace.htm.

[38] NAH, F. F.-H. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, v. 23, n. 3, p. 153–163, 2004.

[39] PATEL, J.; JINDAL, V.; YEN, I.-L.; BASTANI, F.; XU, J.; GARRAGHAN, P. Workload estimation for improving resource management decisions in the cloud. In: . IEEE Twelfth International Symposium on Autonomous Decentralized Systems, 2015. p. 25–32.

[40] VAZQUEZ, C.; KRISHNAN, R.; JOHN, E. Time series forecasting of cloud data center workloads for dynamic resource provisioning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, v. 6, n. 3, p. 87–110, 2015.

[41] Profile-guided optimization. https://en.wikipedia.org/wiki/.

[42] FOO, K. C.; JIANG, Z. M.; ADAMS, B.; HASSAN, A. E.; ZOU, Y.; FLORA, P. Mining performance regression testing repositories for automated performance analysis. In: . 10th International Conference on Quality Software (QSIC), 2010. p. 32–41.

[43] GRECHANIK, M.; LUO, Q.; POSHYVANYK, D.; PORTER, A. Enhancing rules for cloud resource provisioning via learned software performance models. In: . 7th ACM/SPEC on International Conference on Performance Engineering, 2016. p. 209–214.

[44] RYAN, G.; VALVERDE, M. Waiting online: a review and research agenda. *Internet Research*, v. 13, n. 3, p. 195–205, 2003.

[45] BOGÁRDI-MÉSZÖLY, Á.; RÖVID, A.; LEVENDOVSZKY, T. Performance prediction of web-based software systems. In: *Computational Intelligence in Engineering*. Springer, 2010. p. 323–336.

[46] KOUNEV, S. Performance modeling and evaluation of distributed component-based systems using queueing petri nets. *IEEE Transactions on Software Engineering*, v. 32, n. 7, p. 486–502, 2006.

[47] REISS, C.; WILKES, J.; HELLERSTEIN, J. L. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, p. 1–14, 2011.

[48] TANENBAUM, A. S. *Sistemas operacionais modernos*. 2010.

[49] HAIR, J. F.; BLACK, W. C.; BABIN, B. J.; ANDERSON, R. E.; TATHAM, R. L. *Análise multivariada de dados*. Bookman Editora, 2009.

[50] FIGUEIREDO FILHO, D. B.; SILVA JÚNIOR, J. A. D. Desvendando os mistérios do coeficiente de correlação de pearson (r). 2009.

[51] BOSCARIOLLI, L. A. S. S. M. P. C. *Introdução a mineração de dados com aplicações em r*. 2016.

[52] PIATETSKY-SHAPIRO, G. The data-mining industry coming of age. *IEEE Intelligent Systems and their Applications*, v. 14, n. 6, p. 32–34, 1999.

[53] CAMILO, C. O.; SILVA, J. C. D. Mineração de dados: Conceitos, tarefas, métodos e ferramentas. *Universidade Federal de Goiás (UFC)*, p. 1–29, 2009.

[54] JIN, H.; QIN, H.; WU, S.; GUO, X. Ccap: a cache contention-aware virtual machine placement approach for hpc cloud. *International Journal of Parallel Programming*, v. 43, n. 3, p. 403–420, 2015.

[55] FAUZIA, N.; ELANGO, V.; RAVISHANKAR, M.; RAMANUJAM, J.; RASTELLO, F.; ROUNTEV, A.; POUCHET, L.-N.; SADAYAPPAN, P. Beyond reuse distance analysis: Dynamic analysis for characterization of data locality potential. *ACM Transactions on Architecture and Code Optimization (TACO)*, v. 10, n. 4, p. 53, 2013.

[56] MARS, J.; TANG, L.; HUNDT, R.; SKADRON, K.; SOFFA, M. L. Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations. In: . 44th IEEE/ACM International Symposium on Microarchitecture, 2011. p. 248–259.

[57] RAMESHAN, N. On the role of performance interference in consolidated environments. In: . IEEE/USENIX International Conference on Autonomic Computing, 2016.

[58] ZHANG, J.; FIGUEIREDO, R. J. Application classification through monitoring and learning of resource consumption patterns. In: . 20th International Parallel and Distributed Processing Symposium, 2006. p. 10–pp.

[59] POPIOLEK, P. F.; MENDIZABAL, O. M. Monitoring and analysis of performance impact in virtualized environments. *Jornal of Applied Computing Research*, v. 2, n. 2, p. 75–82, 2012.

[60] ALAM, S. R.; BARRETT, R. F.; KUEHN, J. A.; ROTH, P. C.; VETTER, J. S. Characterization of scientific workloads on systems with multi-core processors. In: .

[61] ALVES, M. M.; DRUMMOND, L. M. D. A. A quantitative model for predicting cross-application interference in virtual environments. *arXiv preprint arXiv:1610.04309*, 2016.

[62] MCDOUGALL, R.; ANDERSON, J. Virtualization performance: perspectives and challenges ahead. *ACM SIGOPS Operating Systems Review*, v. 44, n. 4, p. 40–56, 2010.

[63] PU, X.; LIU, L.; MEI, Y.; SIVATHANU, S.; KOH, Y.; PU, C. Understanding performance interference of i/o workload in virtualized cloud environments. In: . IEEE 3rd International Conference on Cloud Computing, 2010. p. 51–58.

[64] HAN, J.; KAMBER, M.; PEI, J. *Data mining: Concepts and techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[65] DA SILVA, L. A.; PERES, S. M.; BOSCARIOLI, C. *Introdução à mineração de dados: com aplicações em r*. Elsevier Brasil, 2017.

[66] DESGRAUPES, B. Clustering indices. *University of Paris Ouest-Lab ModalX*, v. 1, p. 34, 2013.

[67] CHARRAD, M.; GHAZZALI, N.; BOITEAU, V.; NIKNAFS, A. *Determining the best number of clusters in a data set*. https://cran.r-project.org/web/packages/NbClust/NbClust.pdf, 3,0. ed., 2015.

[68] CALIŃSKI, T.; HARABASZ, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, v. 3, n. 1, p. 1–27, 1974.

[69] ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, v. 20, p. 53–65, 1987.

[70] TAYLOR, R. Interpretation of the correlation coefficient: a basic review. *Journal of diagnostic medical sonography*, v. 6, n. 1, p. 35–39, 1990.

[71] Benchmark. https://pt.wikipedia.org/wiki/Benchmark_(computa%C3%A7%C3%A3o).

[72] Microbenchmark. https://en.wiktionary.org/wiki/microbenchmark.

[73] DE ROSE, C. A.; NAVAUX, P. Fundamentos de processamento de alto desempenho. *Anais: 2a Escola Regional de Alto Desempenho*, p. 3–29, 2002.

[74] CURINO, C.; JONES, E.; ZHANG, Y.; MADDEN, S. Schism: a workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment*, v. 3, n. 1-2, p. 48–57, 2010.

[75] TUDORICA, B. G.; BUCUR, C. A comparison between several nosql databases with comments and notes. In: . 10th Roedunet International Conference (RoEduNet), 2011. p. 1–5.

[76] Getting started. https://github.com/brianfrankcooper/YCSB/wiki/Getting-Started.

[77] Apache accumulo. https://accumulo.apache.org/.

[78] Apache cassandra. http://cassandra.apache.org/.

[79] Couchbase. https://www.couchbase.com/.

[80] Amazon dynamodb. aws.amazon.com/dynamodb.

[81] Elasticsearch. https://www.elastic.co/products/elasticsearch.

[82] Apache hbase. https://hbase.apache.org/.

[83] Hypertable. http://www.hypertable.org/.

[84] Infinispan. http://infinispan.org/.

[85] Java se technologies - database. http://www.oracle.com/technetwork/java/javase/jdbc/index.html.

[86] Mongodb atlas. https://www.mongodb.com/.

[87] Orientdb. https://orientdb.com/.

[88] Redis. https://redis.io/.

[89] Tarantool. https://tarantool.io/.

[90] Running a workload. https://github.com/brianfrankcooper/YCSB/wiki/Running-a-Workload.

[91] SCHMIDT, A. V.; BOITO, F. Z.; PILLA, L. L. Fundamentos de estatística para análise de desempenho. *Anais: XVII Escola Regional de Alto Desempenho*, p. 3–24, 2017.