UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Gabriel Rosa Costa Giacomoni Pes

Implementação de um serviço de monitoramento de contadores de desempenho com baixa sobrecarga

Florianópolis
2022

Gabriel Rosa Costa Giacomoni Pes

Implementação de um serviço de monitoramento de contadores de desempenho com baixa sobrecarga

Trabalho de conclusão de curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do título de Bacharel em Ciências da Computação.
Orientador: Prof. Odorico Machado Mendizabal, Dr.
Coorientadora: Profª. Karina dos Santos Machado, Drª.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Pes, Gabriel
Implementação de um serviço de monitoramento de contadores de desempenho com baixa sobrecarga / Gabriel Pes ; orientador, Odorico Machado Mendizabal, coorientador, Karina dos Santos Machado, 2022.
75 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2022.

Inclui referências.

1. Ciências da Computação. 2. monitoramento de contadores de desempenho. 3. melhoria de performance. 4. ambientes virtualizados. I. Machado Mendizabal, Odorico. II. dos Santos Machado, Karina. III. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. IV. Título.

Gabriel Rosa Costa Giacomoni Pes

Implementação de um serviço de monitoramento de contadores de desempenho com baixa sobrecarga

O presente trabalho em nível de graduação foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Jônata Tyska Carvalho , Dr.
Universidade Federal de Santa Catarina

Prof. Márcio Bastos Castro, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Bacharel em Ciências da Computação.

Coordenação do Programa de
Pós-Graduação

Prof. Odorico Machado Mendizabal, Dr.
Orientador

Profª. Karina dos Santos Machado, Drª.
Coorientadora
Universidade Federal do Rio
Grande – FURG

Este trabalho é dedicado à minha querida avó Teresa, mulher lutadora, forte e que sempre me apoiou nos momentos que eu mais precisei. Também ao meu avô Roberto (*in memorian*), que foi um grande amigo e companheiro, e que infelizmente não pôde vivenciar este momento comigo.

AGRADECIMENTOS

Agradeço à minha família, por me dar condições de estudar, me incentivar e sempre acreditar em mim.

À minha namorada Gabriela, que sempre esteve ao meu lado me incentivando e apoiando nos momentos mais difíceis da minha vida.

Aos meus amigos, Davy, Felipe, Guilherme e Jhonattan, que me proporcionaram diversos momentos de alegria e que compreenderam minha ausência enquanto eu me dedicava à realização deste trabalho.

Aos meus colegas da Falcon, que me fizeram crescer profissionalmente neste período, além de serem grandes amigos, me apoiando nestes últimos momentos da graduação.

Ao meu orientador Odorico e minha coorientadora Karina, que acreditaram e incentivaram para a realização deste trabalho, sempre dispostos a auxiliar de maneira atenciosa para a garantia de um trabalho com qualidade.

Muito Obrigado!

RESUMO

Contadores de desempenho são úteis para prever a utilização de recursos no sistema para detecção de gargalos, verificar a disponibilidade do sistema, além de detectar anomalias presentes no mesmo. Entretanto, em muitos sistemas a utilização de contadores em grande escala geram um impacto no desempenho e no consumo de energia, que muitas das vezes são indesejáveis pois prejudicam a eficiência dos serviços disponíveis ao sistema. Arquiteturas de microsserviços estão cada vez se tornando mais populares em diversos sistemas, dado a facilidade de criação e manutenção de serviços sem a necessidade de dependência dos demais serviços já hospedados no mesmo servidor físico. Máquinas virtuais dividem entre si os recursos do sistema e podem gerar uma ineficiência quando disputam os recursos na qual estão hospedadas. O uso de máquinas virtuais quando em grande escala é frequentemente utilizado em conjunto com serviços de coletas de métricas de desempenho. Portanto, nestes ambientes em que diversas aplicações são executadas em paralelo, o monitoramento com contadores de desempenho nestas aplicações pode se tornar custoso se não administrado com cautela. Este trabalho envolve a criação de uma ferramenta de integração com serviços de coleta de métricas de desempenho, que reduz o impacto que os contadores de desempenho geram em relação ao sistema hospedeiro utilizando técnicas em ciência de dados desenvolvidas em outros trabalhos. Também, realiza a comparação dos diferentes parâmetros que podem ser passados sobre esta ferramenta. Os resultados incluem uma ferramenta adaptável, extensível, configurável e de fácil implementação, que integra o coletor de métricas do usuário, e que informa automaticamente um conjunto reduzido de contadores de desempenho em comparação com os contadores originalmente utilizados.

Palavras-chave: monitoramento de contadores de desempenho; melhoria de performance; ambientes virtualizados.

ABSTRACT

Performance counters are useful to predict system resource usage to detect bottlenecks, check system availability, and detect system anomalies. However, in many systems, the use of large-scale meters generates an impact on performance and energy consumption, which are often undesirable because they impair the efficiency of the services available to the system. Microservices architectures are becoming more and more popular in different systems, given the ease of creating and maintaining services without the need to depend on other services already hosted on the same physical server. Virtual machines divide system resources among themselves and can generate inefficiency when they compete for the resources on which they are hosted. The use of virtual machines when on a large scale is often used in conjunction with performance metrics collection services. Therefore, in these environments where several applications run in parallel, monitoring with performance counters in these applications can become costly if not managed with care. This work involves the creation of an integration tool with performance metrics collection services, which reduces the impact that performance counters generate in relation to the host system using data science techniques developed in other works. It also compares the different parameters that can be passed on to this tool. The results include an adaptable, extensible, configurable and easy-to-implement tool that integrates the user metrics collector and automatically reports a reduced set of performance counters compared to the counters originally used.

Keywords: performance counters; performance improvement; virtualized environments.

LISTA DE FIGURAS

LISTA DE TABELAS

DBMS *Database Management Systems*
KDD *Knowledge Discovery from Data*
PCP *Performance co-Pilot*
PMCD *Performance Metrics Collection Daemon*
PMDAs *Performance Metrics Domain Agents*
PMID *Performance Metric Identifier*
PMNS *Performance Metrics Name Space*

SUMÁRIO

1 INTRODUÇÂO

Com o crescimento significativo de aplicações mais complexas de conteúdo geral, a utilização de microsserviços se tornou fundamental, pois tem como premissa a facilidade de replicação de sistemas escaláveis e a diminuição da dependência sobre o sistema hospedeiro (LEWIS; FOWLER, 2014). Com o crescimento da indústria de hardware, cada vez mais serviços virtuais são suportados em servidores físicos. Contudo, a complexidade de gerenciamento de recursos nesses servidores físicos aumenta quando submetida a diferentes cargas de trabalho oriundas de máquinas virtuais. Assim, são necessárias estratégias para otimização de desempenho e o provisionamento de recursos de forma mais eficiente dentro de sistemas baseados em microsserviços.

Constantemente a necessidade do monitoramento de sistemas baseados em microsserviços cresce, desde a avaliação de métricas de propósitos mais gerais como desempenho e consumo, até a avaliação de consequências relacionadas à estas métricas, como por exemplo a detecção de *malwares* no sistema (XIA *et al.*, 2012). Assim, a utilização de contadores de desempenho disponibilizados em hardware e software são importantes para a coleta de dados quando utilizadas em aplicações que visam otimização de desempenho, assim como funcionalidades diversas dependentes do monitoramento de recursos (SHANG *et al.*, 2015).

Entretanto, ao contrário de sistemas monolíticos que necessitam de apenas um contador de métricas centralizado, sistemas baseados em microsserviços necessitam muitas das vezes contadores relacionados ao serviço em questão, independentes dos demais serviços executados no sistema. Com isso, diversos sistemas possuem grandes quantidades de contadores de desempenho, porém possuem uma sobrecarga envolvida (POPIOLEK; MENDIZABAL, 2013; RAMESHAN, 2016). Esta sobrecarga é gerada quando há disputa de recursos no sistema conforme o crescimento na utilização dos contadores (POPIOLEK; MENDIZABAL, 2013). Como o objetivo do uso dos contadores no cenário proposto é a avaliação de desempenho e consumo, é necessário estabelecer um balanceamento entre a quantidade de contadores sem prejudicar na avaliação dos mesmos.

A utilização de técnicas relacionadas à ciência de dados é uma prática comum para o tratamento de problemas que envolvem grandes quantidades de dados (SKIENA, 2017), que visam a extração de conhecimento; a detecção de padrões; ou a percepção de soluções para o determinado problema. Existem na literatura métodos que abordam técnicas em ciência de dados aplicados a amostras de contadores de desempenho, tendo como objetivo tratar contadores que não apresentam dados relevantes e que possuem características comuns entre si, permitindo assim a redução da quantidade de contadores no sistema sem reduzir a integridade das métricas

em análise (POPIOLEK *et al.*, 2021). Neste trabalho pretende-se aplicar as técnicas disponíveis na literatura com o objetivo de diminuição no número de contadores de desempenho do sistema em nível de aplicação, em um serviço que visa a extensibilidade, escalabilidade e fácil manuseio.

1.1 JUSTIFICATIVA

Ferramentas de coleta de monitores de desempenho são utilizadas para o monitoramento de sistemas, principalmente em arquiteturas baseadas em microsserviços. Entretanto, estas ferramentas podem ocasionar sobrecustos não previstos no sistema monitorado. Em (NOWAK; BITZES, 2014) é feita uma análise da ferramenta *perf* em um sistema Linux, executando uma variedade de testes sobre *benchmarks* com diversas configurações, para assim quantificar a sobrecarga envolvida nestes ambientes. De acordo com os autores, é reportada uma dificuldade na apresentação dos resultados para alguns experimentos, na qual era possível obter grandes variações sobre os experimentos em diferentes momentos. Os resultados apresentados por (NOWAK; BITZES, 2014) indicam sobrecargas envolvidas conforme o crescimento tanto da quantidade de eventos monitorados quanto na quantidade de instâncias em paralelo executadas. Vale ressaltar também que a periodicidade destes eventos impactam drasticamente a sobrecarga nestes sistemas.

Em (POPIOLEK; MENDIZABAL, 2013) é realizada uma análise de impacto da utilização de virtualização em sistemas, introduzindo *workloads* sintéticos que reproduzem características específicas em diferentes perfis. De acordo com Popiolek e Mendizabal (2013) o impacto de performance devido à virtualização não é facilmente detectado, sendo necessário a execução dos experimentos em situações de extrema carga para visualização dos efeitos colaterais causados pela virtualização. Os resultados demonstrados apontam uma degradação na performance do sistema quando os recursos compartilhados são competidos conforme o aumento da quantidade de máquinas virtuais instanciadas.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um serviço de monitoramento com o objetivo de redução do número de contadores observados sem prejudicar a qualidade das informações monitoradas.

1.2.2 Objetivos Específicos

- Sintetizar a fundamentação teórica disponível na literatura com relação à redução da quantidade de contadores de desempenho tomando como referência a

abordagem de Popiolek *et al.* (2021).

- Gerar uma arquitetura de software que seja adaptável, extensível, de fácil configuração e adequada para integração com outras ferramentas de monitoramento já disponíveis na comunidade.
- Implementar um protótipo que implemente a técnica de redução automática de contadores de desempenho proposta originalmente em Popiolek *et al.* (2021).
- Avaliar a usabilidade do serviço criado no cenário atual, junto à ferramentas de coleta de métricas populares como o Perfmon.
- Realizar avaliações experimentais para a comparação da qualidade dos contadores selecionados no serviço proposto em relação ao ambiente sem o serviço, sem observar os custos computacionais.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante desse documento está organizado como segue. O Capítulo 2 apresenta os conceitos fundamentais relacionados ao trabalho. O Capítulo 3 apresenta os trabalhos relacionados a este estudo. O Capítulo 4 apresenta a metodologia proposta. O Capítulo 5 apresenta a avaliação da ferramenta desenvolvida. Por fim, o Capítulo 6 apresenta as considerações finais sobre este trabalho.

2 FUNDAMENTOS

Este capítulo tem como objetivo apresentar os fundamentos necessários para a compreensão deste trabalho. A Seção 2.1 apresenta os conceitos de arquiteturas em microsserviços, mostrando as diferenças comparado à arquiteturas monolíticas, as diferentes formas de virtualização e a importância de aplicabilidade neste trabalho e em trabalhos futuros. A Seção 2.2 apresenta as ferramentas Co-Pilot e Perfmon, que realizam a coleta de contadores de desempenho em nível de aplicação e muito utilizadas pela comunidade. A Seção 2.3 apresenta os conceitos relacionados a ciência de dados, em especial os métodos de agrupamento. A Seção 2.3.4 apresenta métricas de validação de agrupamento. Esses conceitos foram abstraídos neste trabalho, mas são importantes para a avaliação em trabalhos futuros.

2.1 ARQUITETURA DE MICROSSERVIÇOS

Microsserviços é um estilo arquitetural que visa reproduzir um projeto de software que opera sobre diferentes serviços de modo independente e que se comunicam entre si, compondo assim a aplicação pretendida (LEWIS; FOWLER, 2014).

Arquiteturas monolíticas concentram todas as funcionalidades da aplicação num único serviço. Com isso, apresentam uma maior dificuldade no gerenciamento de recursos e na manutenção do sistema conforme o crescimento do mesmo. Já arquiteturas baseadas em microsserviços possuem um grande potencial de escalabilidade, devido ao desacoplamento de funcionalidades do sistema, operando apenas nas características específicas de um determinado módulo sobre a aplicação alvo. Dessa forma, microsserviços apresentam um gerenciamento de recursos mais simplificado, tornando mais prático o desenvolvimento em aplicações mais complexas, devido a capacidade de utilização de serviços mais simples e que compõem o sistema ao todo. Possuem também o diferencial na realização de diversas técnicas para melhoria de usabilidade sobre o usuário final, como balanceamento de carga e a utilização de *clusters*, principalmente em aplicações *Web* (NOGUEIRA *et al.*, 2001).

Entretanto, arquiteturas baseadas em microsserviços também possuem custos envolvidos e devem ser muito bem pensadas quanto na utilização da aplicação alvo. Aplicações que demonstram poucas possibilidades de modularização por serem pequenas ou até mesmo por exigirem um alto desempenho e que funcionam muito bem em sistemas monolíticos, muitas das vezes trazem mais desvantagens do que vantagens.

Dessa maneira, é necessário ponderar a necessidade de implementação de uma arquitetura em nível de microsserviços. Migrações para microsserviços geralmente são feitas após a utilização do sistema pelo usuário final, no qual são levantados as necessidades relativas ao aumento na usabilidade do mesmo, entretanto são muito

custosas para o desenvolvedor (BECKER; LUCRÉDIO, 2020). Com isso, é vantajoso realizar o levantamento da possibilidade de migração anterior ao desenvolvimento, tendo assim um impacto menor caso a utilização seja adotada.

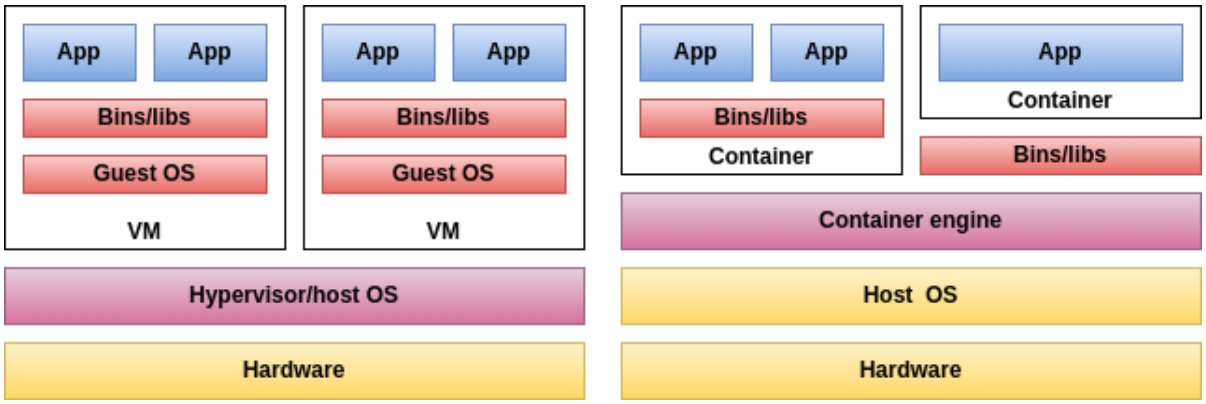
2.1.1 Virtualização em Máquinas virtuais

Máquinas virtuais são cópias de um sistema físico específico que permitem o isolamento total do ambiente de uma aplicação, sendo geridas pelo *hypervisor*. *Hypervisor* é uma camada de software que gerencia Sistemas Operacionais em paralelo, separando os recursos de cada sistema independentemente (IBM, 2019). São exemplos de *hypervisors*: Xen, VMware ESX e Microsoft Hyper-V (FELTER *et al.*, 2015).

2.1.2 Virtualização em containers

Diferente de Máquinas Virtuais, *containers* são virtualizações que ocorrem a nível de Sistema Operacional, não simulando integralmente o ambiente da máquina (PAHL, 2015). A utilização de virtualizações baseadas em *containers* dispõe da execução de ambientes isolados, compartilhados com os outros recursos da máquina hospedeira. Na Figura 1 são mostradas as diferenças de camadas presentes entre a virtualização em máquinas virtuais e em *containers*.

Figura 1 – Diferenças de virtualização em máquinas virtuais e em *containers*



Fonte: Adaptada de (PAHL, 2015)

Containers Engines são camadas de software responsáveis pelo gerenciamento dos *containers*, executando múltiplas instâncias no mesmo *kernel* do Sistema Operacional, e assim fornecendo um ambiente de fácil controle e de gerenciamento de dependências, como por exemplo o Docker. Já orquestradores de *containers* são ferramentas de gerenciamento em nível macro, fornecendo a coordenação, agendamento, comunicação e dimensionamento de muitos *containers*, como por exemplo Docker Swarm e Kubernetes.

2.2 MONITORAMENTO POR CONTADORES DE DESEMPENHO

Existem três definições para monitores de desempenho (POPIOLEK, 2018). A primeira definição diz que são recursos que mensuram e disponibilizam sobre a camada lógica do sistema as métricas associadas a eventos ocorridos no processador; a segunda define contadores como serviços utilizados na camada do Sistema Operacional que coletam e disponibilizam métricas associadas ao hardware; e a última define contadores de desempenho como ferramentas que disponibilizam as informações coletadas dos monitores na camada do SO de forma limpa e legível para o usuário, em nível de aplicação.

Esta seção compreende o detalhamento de algumas ferramentas em nível de aplicação para a coleta dos contadores de desempenho.

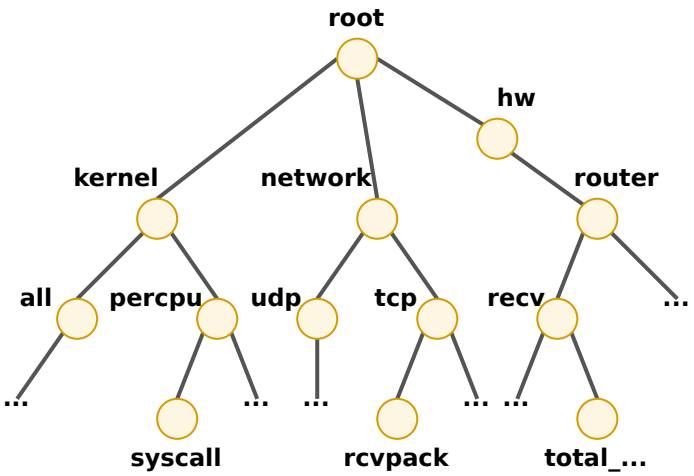
2.2.1 Performance Co-Pilot

Performance Co-Pilot (PCP, 2022) é um conjunto integrado de ferramentas, interfaces de usuário e serviços que suportam análise de desempenho em tempo real e retrospectiva. Com ele, se é possível realizar a coleta de métricas de desempenho a partir do *kernel* de um sistema operacional, de um *Database Management System* (DBMS), de um serviço de camadas ou de um aplicativo de usuário final. Além disso, o *Performance Co-Pilot* (PCP) pode gerar *Performance Metrics Domain Agents* (PMDAs), responsáveis por reunir as métricas que estão sendo coletadas, fornecendo apenas as selecionadas pelo usuário e encapsulando num único serviço.

Sobre as métricas de desempenho, é disponibilizada uma grande variedade, e cada uma possuindo sua própria estrutura e semântica. O PCP utiliza para a nomenclatura das métricas uma uniformização por meio de uma biblioteca padrão *Performance Metrics Name Space* (PMNS) que é extensível e modificável, com o intuito de obter uma melhor adequação ao serviço utilizado e permitindo uma fácil integração com novos contadores de métricas. Esta biblioteca disponibiliza uma classificação hierárquica para cada métrica a partir de um sistema hospedeiro. Na Figura 2 é exemplificado uma estrutura de árvore simples gerada pelo PMNS.

Cada métrica de desempenho possui seus metadados, possuindo seu identificador (ID) único representado pelo *Performance Metric Identifier* (PMID); o formato do dado representado; a semântica da métrica, possuindo um valor instantâneo ou de execução livre; a dimensão, sendo de evento, espaço ou tempo; a escala, que representa o tamanho da informação; o texto de descrição da métrica; e por fim a indicação de um ou mais valores associados à mesma métrica de desempenho. Para essas métricas que possuem mais de um domínio, o PCP representa com um conjunto de valores associando aquela métrica específica. Para cada sub-valor, existe um ID associado. Estas métricas são dependentes da arquitetura na qual o sistema está sendo

Figura 2 – Exemplo de estrutura PMNS



Fonte: Elaborado por (PCP, 2022)

rodado, sendo assim algumas métricas possuem diferentes instâncias dependendo da arquitetura.

Sobre a estrutura, o PCP visa principalmente a extensibilidade do produto. Sendo assim, possui uma arquitetura cliente/servidor para a troca de informações entre múltiplos serviços que estão sendo monitorados. Cada configuração recebe um tratamento automático ao ser realizadas mudanças, refletindo nos outros serviços interligados; além de possuir uma detecção de perda de conexão com cliente-servidor e suporte para reconexão automática. Existem dois tipos de serviços:

- Coletor: Responsável pela coleta e exportação das métricas, por meio de um PMCD e um ou mais PMDA.
- Monitor: Responsável pela importação das métricas de desempenho de um ou mais coletores, sendo consumidos por ferramentas de monitoramento, gerenciamento ou de registro.

Por fim, possui um suporte à entrega de métricas por meio de um histórico na forma de log. Logs de arquivos PCP podem ser guardados tanto no cliente, quanto no servidor, ou até mesmo em ambos, fornecendo assim uma plataforma para unificação de acesso.

2.2.2 PerfMon

Incluído no Windows server, o PerfMon (MICROSOFT, 2022) é um monitor de desempenho exclusivo para o Sistema Operacional Windows que coleta informações detalhadas sobre desempenho, disponibilizando uma ampla quantidade de contadores de desempenho; e com uma fácil configuração, integração e disponibilização, tudo em uma interface visual bastante simplificada.

O PerfMon possibilita a criação de conjuntos de coletores de dados, que são elementos da ferramenta que armazenam os coletores de desempenho que em algum momento possam começar a serem coletados. Na seleção dos contadores de desempenho, é possível definir um intervalo de tempo no qual cada coleta será processada, assim como o *host* que será executado.

Assim como outros coletores, o PerfMon permite a utilização de contadores de desempenho em diferentes partes do sistema, como disco lógico, memória, informações do processador e informações do processo. Existem ainda centenas de áreas e subáreas do sistema que o PerfMon disponibiliza para a coleta das informações, sendo algumas citadas na Tabela 1.

Tabela 1 – Exemplos de contadores de desempenho do PerfMon

Métrica	Descrição
% Processor Time	Percentagem média de utilização de CPU.
% Privilege Time	Percentagem media de utilização de CPU em modo kernel.
% Interrupt Time	Percentagem média de utilização de CPU recebendo e servindo interrupções.
Context Switches/sec	Taxa média de trocas de contexto por segundo.
Processor Queue Length	Número de processos enfileirados em estado pronto.
% DPC Time	Percentagem média de utilização de CPU recebendo e servindo interrupções DPCs.
% User Time	Percentagem média de utilização de CPU em modo usuário.
% Idle Time	Percentagem média de CPU ociosa.
DPCs Queued/sec	Enfileiramento médio de DPCs por segundo.
Interrupts/sec	Taxa média de interrupções por segundo.

Fonte: Adaptado de (POPIOLEK, 2018)

2.3 CIÊNCIA DE DADOS

Ciência de dados é a área que une as áreas de estatística e ciências da computação, tendo como princípio a manipulação em domínios de aplicação, principalmente em grandes volumes de dados (SKIENA, 2017).

Nesta seção são apresentados processos e técnicas relacionadas a ciência de dados que foram necessários para o desenvolvimento deste trabalho. Na Subseção 2.3.1 é descrito o processo principal utilizado para a manipulação dos dados; na Subseção 2.3.2 são apresentados os diferentes modelos utilizados para automatização do processamento dos dados; na Subseção 2.3.3 são descritos os métodos específicos

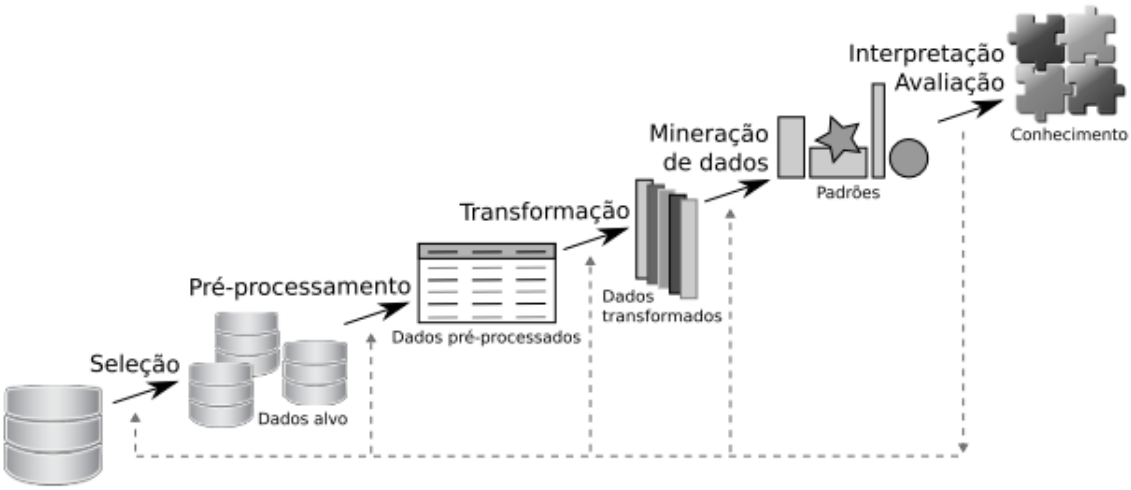
de agrupamento dos dados; e por fim na Subseção 2.3.5 é descrita a correlação de Pearson no contexto deste trabalho.

2.3.1 Processo KDD

Descoberta de conhecimento em dados ou KDD (do inglês *knowledge discovery from data*) é uma metodologia de processo utilizada para extração de conhecimento de um determinado volume de dados. Outros termos na literatura eram utilizados para este mesmo conceito, como *knowledge discovery*, *data discovery* e *knowledge extraction*. No presente, Ciência de Dados generaliza estes termos (STEELE *et al.*, 2016).

Esta metodologia é utilizada com o intuito de maximizar a produtividade no processamento de grandes volumes de dados. De acordo com (FAYYAD *et al.*, 1996), o processo KDD refere-se a toda a metodologia de descoberta de dados, sendo a mineração de dados uma das etapas deste processo.

Figura 3 – Métodos de agrupamento hierárquico aplicados em um conjunto de dados.



Fonte: Adaptado de (FAYYAD *et al.*, 1996)

O processo de KDD deve ser iterativo e interativo, com o objetivo de modificar a estratégia que foi adotada nos outros passos, sendo possível visualizar com as setas tracejadas na Figura 3. Por isso, este processo deve ser dividido em etapas, sendo apresentadas a seguir:

- Desenvolvimento e compreensão do domínio da aplicação: Fase de identificação do objetivo do processo, assim como o levantamento do conjunto de dados à ser analisado.
- Seleção: Fase de seleção dos dados que possuem significado para a aplicação de interesse, de acordo com o conjunto de dados levantado na etapa anterior.

- Pŕe-processamento: Realização da limpeza dos dados selecionados na etapa de seleção. Aqui são definidas as estratégias exclusão ou tratamento dos dados tanto faltantes quanto discrepantes, de acordo com o domínio da aplicação.
- Transformação: Etapa para a busca de formas na representação dos dados de acordo com o objetivo da análise, com o objetivo de reduzir o número de variáveis, assim como padronizá-las.
- Mineração de dados: Etapa de busca e execução de métodos para a busca por padrões de interesse de acordo com o domínio da aplicação.
- Interpretação/Avaliação: Fase de Interpretação dos padrões ou modelos extraídos.

2.3.2 Aprendizado de máquina

Aprendizado de máquina (ou no inglês *machine learning*) é uma classe de algoritmos orientados por dados, que tem como objetivo investigar como computadores podem aprender, baseados em dados previamente estabelecidos (AGARWAL, 2014). Seus casos de uso são para previsão ou tomada de decisões automáticas, a partir dos dados coletados. Existem dois tipos principais de aprendizado de máquina:

- Aprendizado supervisionado: utiliza dados previamente descritos da classe na qual pertencem, adequadas à aplicação de interesse. Existem dois modelos no aprendizado supervisionado:
  - Classificação: processo de busca de um modelo para obtenção da classes de dados alvo de acordo com parâmetros previamente estabelecidos.
  - Regressão: processo de busca de um modelo para a predição de um valor contínuo ou discreto de dados.
- Aprendizado não-supervisionado: utiliza dados na qual as classes não estão estabelecidas, agrupando os dados de acordo com uma quantidade pré estabelecida ou não de classes à serem encontradas. Existem dois modelos no aprendizado não-supervisionado:
  - Agrupamento: processo com o objetivo de agrupar dados de interesse ou separar elementos de um conjunto em subconjuntos ou grupos, no qual compartilham propriedades em comum.
  - Associação: processo que cria associações entre ocorrências de diferentes classes nos dados.

2.3.3 Agrupamento

Entre as diferentes técnicas, neste trabalho será detalhada a técnica de agrupamento. A técnica de agrupamento define-se como o agrupamento de objetos de dados baseando-se apenas nas informações que descrevem os objetos e suas relações (TAN *et al.*, 2013), com o objetivo de agrupar objetos similares e diferenciar objetos não relacionados. Grupos são criados de acordo com as similaridades, tendendo a ser maximizadas entre elementos no mesmo conjunto, e minimizadas entre diferentes grupos.

Medidas de similaridade possuem como objetivo calcular o quão similar dois atributos são, normalmente utilizando-se uma função de distância para descrevê-las. Em (SKIENA, 2017) é apresentado diversas métricas de cálculos de distância, dependentes dos tipos de variáveis e propriedades do problema.

- *Euclidean distance*: Medida mais popular para cálculo de distância, no qual é considerado apenas o tamanho de uma linha que conecta os dois atributos.
- *Manhattan distance*: Medida comumente utilizada em problemas onde há obstáculos entre dois atributos, no qual se deve representar a soma da distância entre pontos arbitrários que levam ao atributo de destino.

Em (AGARWAL, 2014), diversos tipos de agrupamentos são apresentados, com propósitos diferentes para determinadas classes de dados ou aplicações. A seguir é apresentado características gerais dos métodos mais utilizados:

- Métodos de particionamento:
  - Divisão das instâncias de dados em partições não sobrepostas.
  - Necessário informar quantos grupos devem ser criados.
  - Baseado na distância.
  - Utilização da média ou mediana para a representação do centro dos grupos gerados.
  - Eficaz em instâncias de dados de pequeno à médio porte.
- Métodos hierárquicos:
  - Construção de conjuntos de grupos aninhados em múltiplos níveis.
  - Não requer uma pré-definição do número de grupos que deverão ser criados.
  - Estrutura hierárquica pode ser utilizada para um maior entendimento no relacionamento dos dados.
  - Não corrige grupos já mesclados, ou grupos divididos incorretamente.

- Métodos baseados em densidade:
  - Grupos são gerados a partir de regiões de alta densidade de instâncias, separadas por regiões de baixa densidade.
  - Eficaz na busca de grupos com formatos arbitrários na amostra de dados.
  - Projetado para a busca de grupos com base na definição de centro.
  - Resistente a ruídos ou *outliers*.

Como neste trabalho foi utilizado apenas agrupamento do tipo hierárquico, esse tipo de algoritmo é detalhado a seguir. Agrupamentos hierárquicos organizam conjuntos de dados numa estrutura hierárquica com base na proximidade entre as instâncias, sendo muito úteis quando as fontes de dados representam naturalmente uma estrutura hierárquica. Normalmente as estruturas geradas são mostradas como árvores binárias ou dendogramas, sendo a raiz o conjunto inteiro; os nós intermediários representando os conjuntos gerados; e os nós folha representando as instâncias do conjunto. O resultado desse agrupamento é obtido através de um corte no nível desejado da árvore, sendo o número de grupos definido de acordo com o nível que foi feito o corte. Portanto, é necessário o conhecimento do domínio do problema para uma determinação mais exata do nível ideal do corte.

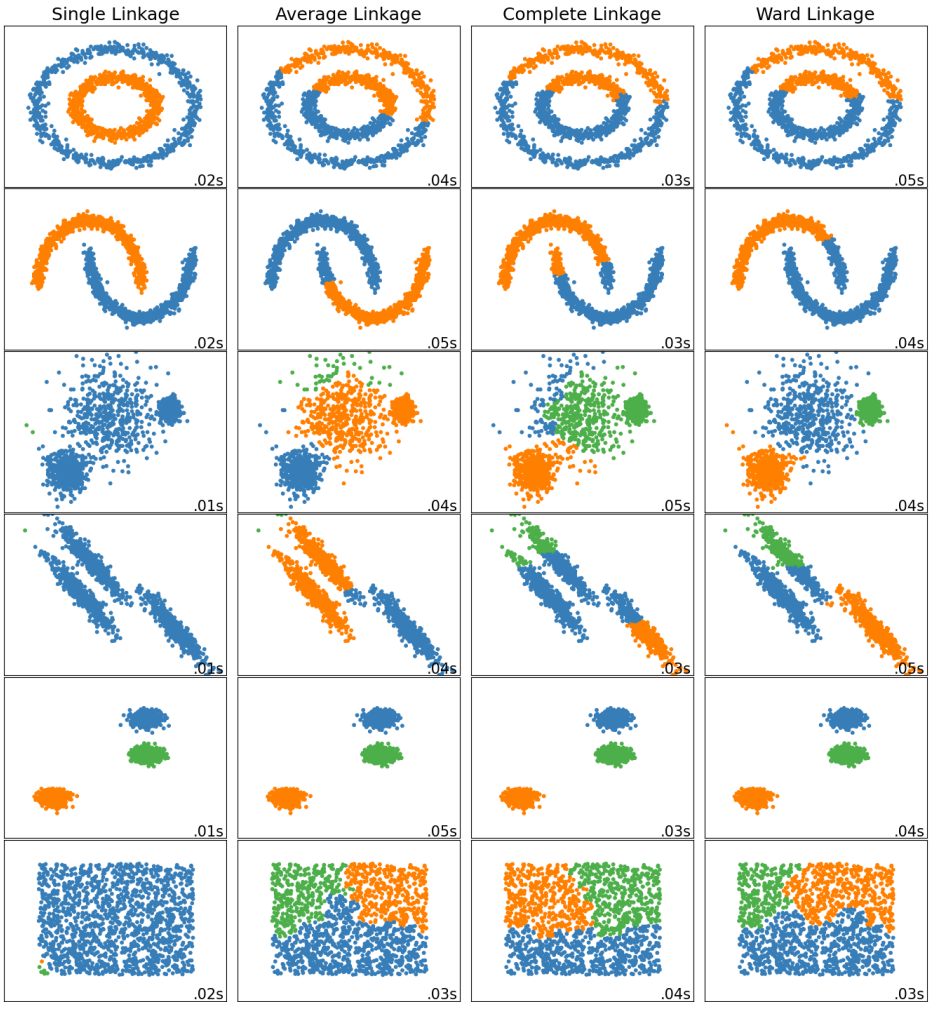
Existem duas abordagens em agrupamentos hierárquicos: aglomerativas e divisivas. Na abordagem aglomerativa, inicialmente todas as instâncias da amostra de dados são grupos individuais, e sucessivamente é realizado a união dos grupos de acordo com um cálculo de proximidade entre os mesmos, até que um único grupo seja formado. Já na abordagem divisiva, o oposto acontece, no qual inicialmente existe apenas um grupo correspondente ao conjunto de dados total, e gradualmente é feita a divisão em grupos menores de acordo com algum critério, até que todos os grupos tenham apenas um elemento.

Os algoritmos que fazem o agrupamento utilizam como base a medida de similaridade entre os grupos. Entretanto, existem diversos modos de se calcular esta similaridade. Em (SKIENA, 2017) é destacado os principais cálculos de similaridade entre grupos, utilizados no agrupamento hierárquico, cada um com vantagens e limitações na identificação dos grupos.

- *Single linkage*: Utiliza como base os elementos mais próximos dentre os dois grupos comparados. É bastante útil na captura de formas não elípticas, entretanto é extremamente sensível a ruídos e *outliers*.
- *Complete linkage*: Utiliza como base os elementos mais distantes dentre os dois grupos comparados. É menos suscetível a ruídos e *outliers*, entretanto pode quebrar grandes agrupamentos e também é enviesado na criação de grupos globulares.

- *Average linkage*: Utiliza como base a média da proximidade entre elementos dos dois grupos comparados, sendo assim utilizado a menor distância. Também menos sensível a ruídos e *outliers*, entretanto ainda tende a criar formas globulares.
- *Nearest centroid*: Utiliza como base o centróide de cada grupo, agrupando assim grupos com menores distâncias em relação ao centróide. Continua produzindo grupos similares ao *Average link*, mas com maior performance, dado que o cálculo de centróide possui menos complexidade comparado ao cálculo da média.
- *Ward linkage*: Utiliza como base o erro quadrático em relação à cada grupo e a união dos dois grupos, utilizando assim o agrupamento com o menor erro quadrático calculado. Também é menos suscetível a ruídos e *outliers*, e tendencioso a formas globulares.

Figura 4 – Métodos de agrupamento hierárquico aplicados em um conjunto de dados



Fonte: Elaborado por (SCIKIT-LEARN, 2022)

Na Figura 4 são representados os diferentes métodos para agrupamentos hierárquicos, sendo aplicados em um único conjunto de dados para representação do modo

de divisão dos grupos. As cores representam os grupos associados a cada instância dos conjuntos de dados. Observa-se que o método *Single Linkage* foi extremamente útil na classificação em conjuntos de dados globulares e parcialmente elípticos (coluna 1; linhas 1 e 2 da Figura 4), porém não conseguiu classificar outros conjuntos de dados em que possuía grupos visualmente distintos mas com pontos relativamente próximos um do outro (coluna 1; linhas 3, 4 e 6 da Figura 4). Já nos outros métodos, visualmente a classificação não obteve bons resultados para formas globulares (coluna 2, 3 e 4; linhas 1 e 2 da Figura 4); mas para conjuntos de dados dispersos, mesmo com dados relativamente próximos, obteve uma boa classificação, em especial o método *Ward Linkage* que classificou com os melhores resultados para grupos que visualmente se diferiam. Vale ressaltar que todos os métodos conseguiram identificar os agrupamentos nos quais os grupos são longes um do outro, formando subconjuntos pontuais e que não possuem *outliers* que invadem outros grupos identificados.

2.3.4 Métricas de validação de agrupamento

Algoritmos de agrupamento possuem como objetivo a partição do conjunto de dados, entretanto diferentes algoritmos ou parâmetros geram diferentes grupos ou até mesmo diferentes estruturas. A verificação e validação de agrupamentos explora a tendência de transição de um conjunto de dados em conjuntos de grupos. De acordo com Greenwood e Tyrrell (2007) as métricas de validações de agrupamento são importantes para verificação de validade conforme o escopo proposto do projeto. Existem 3 critérios de teste e validação:

- *External clustering validation*: Avalia a estrutura de grupos comparando os resultados da análise com um resultado externo que seja conhecido. Realiza a avaliação comparando uma partição do conjunto de dados utilizado, que seja independente do conjunto de grupos criado, com o próprio conjunto de grupos. Assim, a avaliação é realizada comparando pares de elementos do conjunto de dados, para todas as combinações do mesmo, os diferentes casos referentes a cada par a seguir:
  - Caso 1: Mesmo grupo dentro dos grupos gerados e mesma categoria relacionada a partição do conjunto de dados.
  - Caso 2: Mesmo grupo dentro dos grupos gerados e diferente categoria relacionada a partição do conjunto de dados.
  - Caso 3: Diferente grupo dentro dos grupos gerados e mesma categoria relacionada a partição do conjunto de dados.
  - Caso 4: Diferente grupo dentro dos grupos gerados e diferente categoria relacionada a partição do conjunto de dados.

Com estes cenários mensurados, é realizada a avaliação do modelo por meio de métodos estatísticos com os dados obtidos na etapa anterior (GREENWOOD; TYRRELL, 2007).

- *Internal clustering validation*: Avalia a estrutura de grupos utilizando as informações internas do processo de agrupamento, com o intuito de avaliar a qualidade dela sem a referência de informações externas. De acordo com Greenwood e Tyrrell (2007) a validação interna pode ser feita por meio do Coeficiente de Correlação Cofonética, índice utilizado para validação estruturas de agrupamento hierárquicos.
- *Relative clustering validation*: Avalia a estrutura de grupos criada testando diferentes variações dos parâmetros utilizados no algoritmo. Não depende de testes estatísticos utilizados nos outros tipos de validações, no qual são computacionalmente intensivos.

2.3.5 Coeficiente de correlação linear de Pearson

Por definição, a correlação de Pearson é utilizada em regressão para validar o modelo, calculando a correlação entre o valor predito e o valor real. No contexto deste trabalho, a correlação linear de Pearson tem o mesmo objetivo das medidas de similaridade, com a diferença de que é realizado o cálculo entre atributos do *dataset* em relação aos diferentes registros do mesmo. O grau de correlação é representado dentre o domínio de -1 a 1. O sinal representa a direção na qual a correlação está, sendo positiva ou negativa. Quanto mais próximo dos limites do domínio, menor a correlação; e quanto mais próximo de 0, mais forte a correlação (SKIENA, 2017). O coeficiente de correlação linear de Pearson é apresentado na Equação 1.

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

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os principais trabalhos relacionados que expõem métodos úteis para a redução de sobrecarga de contadores de desempenho no sistema.

3.1 REDUÇÃO DE SOBRECARGA

A utilização de métodos manuais e dependentes da experiência e intuição do profissional são amplamente utilizados (SHANG *et al.*, 2015). Entretanto, existe uma dificuldade na manipulação de múltiplos contadores de desempenho normalmente em sistemas de grande escala, além da sobrecarga gerada no monitoramento de contadores de desempenho em microsserviços. Dessa forma, alguns trabalhos apresentam alternativas com métodos automáticos e/ou inteligentes para a redução de forma eficiente e robusta.

Além disso, estes trabalhos possuem uma técnica comum de mineração de dados, sendo Shang *et al.* (2015) o primeiro a aplicar em contadores de desempenho. Esta técnica se resume no agrupamento de contadores de desempenho por meio da separação em *clusters*, e em seguida realizando a manipulação do mesmo de acordo com cada propósito diferente.

3.1.1 Detecção de regressões de desempenho

Em (SHANG *et al.*, 2015), é proposto um método para a detecção de regressões de desempenho por meio de modelos de regressão aplicados em dados de monitoramento, obtidos por conjuntos de contadores de desempenho.

A metodologia proposta é dividida em etapas. Primeiro, é realizada uma análise detalhada do conjunto de contadores disponibilizados no sistema, com o objetivo de remoção de contadores que são redundantes e que não apresentam variância, assim como a remoção de contadores cujos valores podem ser obtidos de forma direta por outros contadores. Em seguida é feito o agrupamento dos contadores restantes, separando em *clusters* de acordo com as similaridades entre eles. Assim que são criados, para cada *cluster* em específico é feita a eleição de um contador de desempenho, por meio de testes de distribuição de probabilidade nos conjuntos de contadores de cada *cluster*. Por fim, é construído, para cada *cluster*, um modelo linear no qual os contadores restantes representam as variáveis dependentes e, a partir destes modelos, é feita a análise destes resultados para definir qual ou quais clusters são representativos para modelos de regressão de desempenho no sistema alvo.

Foram realizados testes em um *benchmark* de suporte médio e numa aplicação consideravelmente grande em processamento e contadores disponibilizados. Dentre os resultados indicam que houve uma drástica redução no número de contadores quando

removido os contadores identificados redundantes. No *benchmark* de suporte médio, dos 28 contadores de desempenho, apenas 2 foram considerados. Já na aplicação consideravelmente maior, a quantidade de contadores foram reduzidos a apenas 4, sendo assim possível concluir que houve uma grande redução conforme o crescimento do sistema.

Portanto, (SHANG *et al.*, 2015) é de grande relevância a este trabalho, justamente ao propor um método automático de redução de contadores de desempenho, de forma eficiente e que escala muito bem conforme a quantidade de contadores.

3.1.2 Detecção de trechos críticos de código

Yao *et al.* (2018) propõe uma alternativa para complementar a avaliação de performance no sistema por meio de adição de contadores específicos em trechos críticos de código em ambientes *Web*. Neste trabalho, é explicado sobre a dificuldade de análise de desempenho no servidor alvo, sendo necessária experiência e intuição para seleção dos contadores mais representativos ou, em situações mais críticas, o conhecimento do código fonte para a adição manual de contadores. Portanto, a solução adotada adiciona trechos críticos de código sugeridos para o usuário de forma automática, com contadores previamente utilizados, e podendo ser descartados posteriormente.

A proposta é dividida em etapas. O primeiro passo é a identificação das requisições que geram mudanças no desempenho do sistema, por meio do monitoramento tanto das requisições quanto dos contadores internos no sistema. A partir dos *logs* gerados é executado o cruzamento dos *timestamps* para cada *log* de requisição e de desempenho, e aplicado posteriormente para cada cruzamento um método similar apresentado por Shang *et al.* (2015) para a avaliação do quão estatisticamente significante é o modelo de regressão gerado pelas métricas coletadas. O segundo passo é a identificação dos métodos no código fonte significantes para a avaliação de desempenho. Desse modo, é adicionado para cada método associado ao *endpoint* daquela requisição um contador, e novamente é feita a execução do sistema por meio de testes. Com estes *logs* gerados, remove-se os contadores que não apresentaram variações ou que pertenciam a métodos que eram chamados por outros em sequência, na qual possuíam também contadores alocados. No terceiro e último passo, é feita a identificação de cada sub-bloco dos métodos já identificados influentes, para a obtenção de uma melhor exatidão na identificação dos trechos mais críticos.

Para a avaliação do método proposto, foram realizados testes em três softwares diferentes, sendo um deles próprio do autor e relativamente maior em comparação ao processamento e a quantidade de contadores disponibilizados. Os resultados indicam uma melhor explicação de desempenho do sistema em relação aos habitualmente utilizados, mas com poucos ganhos em relação à predição. Entretanto, o principal

ganho refere-se à localização dos trechos do código com maior impacto, no qual a metodologia conseguiu atingir apenas 0.5% da quantidade de métodos no sistema, sendo melhor para a identificação dos trechos de código que geraram mais gargalos de desempenho no sistema.

Portanto, esta metodologia propõe uma técnica útil caso seja abordado uma redução dos contadores gerais do sistema, entretanto pode ser utilizada apenas para ambientes em que o propósito é a avaliação de desempenho do sistema como um todo. Diferente de nossa abordagem, que visa a redução de sobrecarga de monitoramento para ambientes de propósito geral.

3.1.3 Redução de sobrecarga em ambientes virtualizados

Popiolek *et al.* (2021) tem como objetivo reduzir a sobrecarga de desempenho no uso de contadores de desempenho em ambientes virtualizados, dado que são os mais afetados por conta da competitividade de recursos no sistema ao todo. A metodologia proposta é a redução de contadores de desempenho, selecionando os que são redundantes no determinado sistema virtualizado.

O método consiste na aplicação de um processo iterativo de extração de informações nos dados coletados uma única vez para a identificação dos contadores não representativos, no qual se divide em passos. Primeiro, realiza-se o processo de experimentação, no qual é feita a partir da injeção de diferentes cargas sintéticas no ambiente, assim tendo uma maior dispersão nos valores dos contadores coletados. Em seguida é realizado o processo de seleção, que se dá na remoção das métricas coletadas dentre um intervalo no início e no fim do experimento, removendo-se cargas coletadas em momentos de aquecimento e resfriamento do sistema, para que o processo de avaliação seja fiel à um sistema que esteja rodando normalmente. No próximo passo é realizado o pré-processamento destes dados, removendo-se contadores no qual não observou-se variância nas métricas coletadas para o determinado intervalo de tempo, e também contadores sem métricas coletadas devido à diversos fatores. A partir do pré-processamento, é realizada a mineração destes dados, que se dá pelo método similar ao proposto por Shang *et al.* (2015) para o agrupamento dos contadores em *clusters*. Em seguida é realizado o processo de descoberta de conhecimento, que consiste na separação de dois grupos de clusters, um com os da iteração atual e o outro com os da seleção de iterações passadas. Essa separação se dá para a identificação de correlações em independentes cenários no momento que foi gerado as cargas sintéticas no ambiente. Com isso, caso ainda tenha experimentos a serem injetados no sistema, o processo volta à primeira etapa, se não segue para a próxima. Por fim é executado o processo de seleção de um contador de desempenho mais representativo para cada cluster gerado na etapa anterior, a partir da avaliação da dissimilaridade dos contadores de cada cluster.

Para a avaliação, este trabalho segue dois objetivos, primeiro avaliar se houve uma redução em relação à sobrecarga no sistema utilizando apenas os contadores selecionados neste método, e segundo avaliar se os contadores selecionados para cada cluster é capaz de caracterizar os diferentes estados do ambiente no qual está sendo avaliado. Foram utilizados *microbenchmarks* para geração de cargas controladas e intensivas e *macrobenchmarks* para simulação de sistemas reais. A avaliação foi realizada pela análise da vazão de operações concluídas por estes benchmarks comparando a utilização do método proposto.

Dos resultados na redução de sobrecarga, ambos cenários se destacam. Nos *macrobenchmarks* a sobrecarga com o monitoramento completo ativado totalizava no pior caso 30% para operações de leitura e 20% leitura/escrita, e reduzindo-se para apenas 10% e 5% respectivamente. Entretanto, a sobrecarga nestes experimentos se dá justamente pela capacidade de vazão do sistema, sendo assim a sobrecarga gerada por meio do método proposto fica imperceptível comparada com a não utilização de contadores de desempenho, totalizando em média apenas 0.77%.

Em relação à avaliação dos conjuntos de contadores, em *microbenchmarks* são executados os mesmos tipos de instruções em relação à uma carga específica avaliada, com isso todos os testes as quantidade de contadores ficam próximas da média de 3.8% em relação ao número total de contadores do sistema. Já para *macrobenchmarks*, devido ao *benchmark* utilizado, as cargas geradas para cada teste são cumulativas, sendo assim as seguintes serão reflexos das anteriores. Portanto foram utilizados os registros de monitoramento dos *microbenchmarks* nas iterações para a estimulação dos componentes do sistema em mais níveis de intensidade. Sendo assim, a quantidade de grupos gerada a partir de *microbenchmarks* e *macrobenchmarks* simultaneamente geram em média 15% para cada carga de trabalho, em relação à quantidade total de contadores do sistema. Para a avaliação qualitativa destes contadores, foi feito o levantamento de contadores considerados importantes para o monitoramento do sistema. Dos 28 contadores que são considerados importantes, apenas 3 deles não foram identificados no método proposto.

A metodologia proposta por Popiolek *et al.* (2021) é de essencial importância para este trabalho, dado que o principal objetivo é a implementação deste método, entretanto num sistema de forma dinâmica e que seja adaptável e extensível.

4 SERVIÇO DE REDUÇÃO DE CONTADORES DE DESEMPENHO

Neste capítulo é apresentada a metodologia proposta para a criação da ferramenta de redução automática de contadores de desempenho. Esta ferramenta tem como objetivo indicar um conjunto reduzido de contadores que representam um sistema sem afetar a qualidade das métricas coletadas, de forma automática e transparente para o utilizador, fornecendo também estatísticas detalhadas no processamento dos dados. As seguintes características compõem a ferramenta proposta:

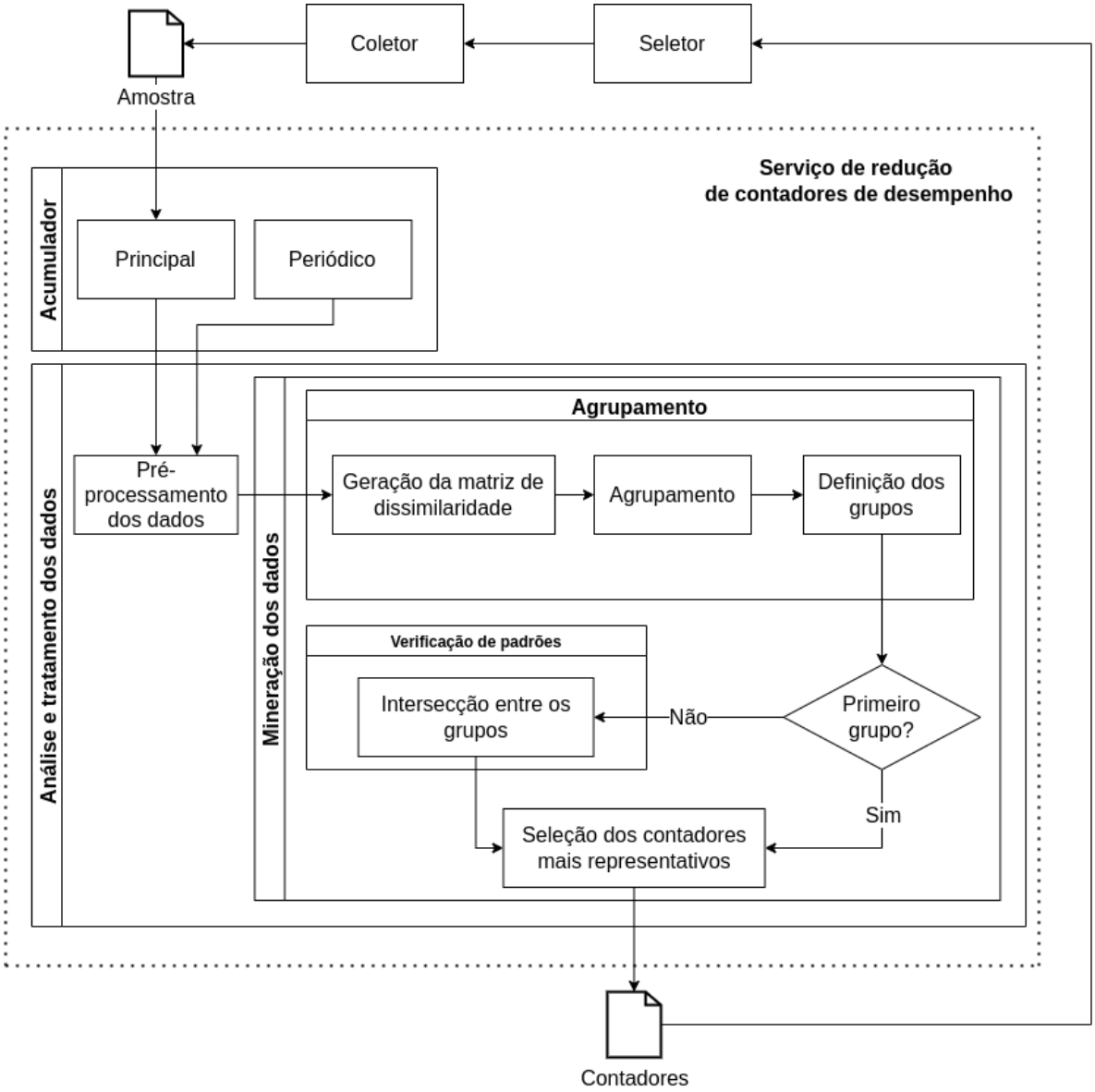
- Adaptável: diferentes módulos com funcionalidades específicas, implicando assim na facilidade de integração com outros serviços, ou a modificação de acordo com a necessidade do sistema.
- Extensível: capacidade de extensão de módulos já existentes ou adição de novos módulos com propriedades diferentes de acordo com os requisitos da aplicação.
- Configurável: fácil configuração, de acordo com o método de seleção dos contadores utilizado, assim como o tipo de algoritmo e as diversas propriedades que procedem dele.
- Adequável para integração: capacidade de integração com tecnologias de manipulação de contadores de desempenho já consolidadas na comunidade.

As vantagens na utilização desta ferramenta incluem: Menor utilização de armazenamento e processamento sobre os dados dos contadores, trazendo assim uma melhoria na performance do sistema; menor demanda de trabalho na seleção de contadores representativos; ou até mesmo a anulação da necessidade de um especialista na seleção manual dos contadores no sistema.

Na Figura 5 é representado o diagrama da arquitetura proposta neste trabalho, com base no modelo desenvolvido em (POPIOLEK *et al.*, 2021) e integrado de acordo com as propriedades de um serviço que pode ser reconfigurado e utilizado numa arquitetura de microsserviços. Abaixo são descritas as funcionalidades de cada módulo apresentado na Figura 5:

- Seletor: responsável pela seleção dos contadores de desempenho que fazem sentido para o sistema alvo.
- Coletor: ferramenta para a coleta dos contadores, definida de acordo com os interesses do usuário que está utilizando.
- Acumulador: módulo responsável pelo recebimento de amostras enviadas pelo coletor, no qual acumula e aciona outros módulos dado condições pré-estabelecidas nas configurações da ferramenta.

Figura 5 – Diagrama da arquitetura proposta



Fonte: Própria

  - Principal: processo principal responsável pela coleta das amostras e acionamento da análise caso o número máximo de amostras seja atingido.
  - Periódico: processo periódico que aciona a análise caso o número mínimo de amostras seja atingido.

- Análise e tratamento dos dados: módulo genérico que pode ser adaptável de acordo com a necessidade do sistema.
  - Pré-processamento dos dados: remoção de dados faltantes; dados que não representam o sistema como todo; e dados não relevantes para a aplicação.
  - Mineração dos dados: módulo genérico para a etapa de mineração dos dados.

    * Agrupamento: módulo genérico para o método de agrupamento.
        - Geração da matriz de dissimilaridade: processo de geração da matriz de dissimilaridade utilizada para o cálculo da distância.
        - Agrupamento: processo de agrupamento dos contadores a partir da matriz de dissimilaridade.
        - Definição dos grupos: processo de definição dos grupos de contadores mais significativos, em que realiza o corte no dendrograma gerado pelo agrupamento dos contadores.
    * Verificação de padrões: módulo genérico para verificação de padrões dos grupos de contadores gerados.
        - Intersecção entre os grupos: módulo que implementa a técnica de intersecção dos grupos de contadores gerados anteriormente (POPIOLEK *et al.*, 2021).
    * Seleção dos contadores mais representativos: processo de seleção dos contadores mais representativos para cada grupo selecionado na etapa anterior.

Esta ferramenta foi desenvolvida inteiramente na linguagem *Python*, utilizando conceitos de orientação a objetos para a modularização mais eficiente, trazendo assim maior independência dos módulos produzidos. A ferramenta está disponível em (PES, 2022).

Para a configuração da aplicação foi utilizado um arquivo que apresenta as características principais utilizadas para a configuração de cada módulo produzido. A seguir é apresentado o detalhamento das propriedades de cada argumento utilizado no arquivo de configuração da ferramenta.

- *DEFAULT*: define a configuração básica da ferramenta
    - *api_receiver*: endereço local ou remoto no qual a ferramenta está hospedada, para fácil integração em arquiteturas de microsserviços.
    - *devmode*: utilizado para geração de *logs* estatísticos, que posteriormente possam ser utilizados para melhoria dos coletores selecionados para o ambiente atual do usuário.
    - *test*: utilizado para execução de testes controlados sobre o ambiente.
- *ACCUMULATOR*: configurações sobre o acumulador
    - *min_samples*: mínimo de amostras necessárias, caso tenha, para que o processo periódico do acumulador acione o processo de análise e tratamento dos dados. Pode ser utilizado como valor 0 para desabilitação da funcionalidade.

  - *max_samples*: máximo de amostras necessárias armazenadas no acumulador, que aciona o processo de análise e tratamento dos dados caso atingido o número limite, utilizada pelo processo principal. Neste trabalho, o parâmetro *max_samples* é mencionado como o tamanho da janela.
  - *period_samples*: tamanho do período calculado em segundos no qual aciona o processo periódico do acumulador para verificação do número mínimo de amostras necessárias.
  - *max_intersection_window*: utilizado para seleção da quantidade de arquivos utilizados na fase de intersecção dos grupos gerados em cada arquivo. Arquivos que não foram selecionados são descartados do sistema, para a não sobrecarga do sistema. Também, é interessante esta opção em casos onde, depois de determinadas execuções, agrupamentos antigos não façam sentido no cenário. Pode ser desabilitado com o valor 0.
  - *periodicity_all_counters*: período no qual todos os contadores do sistema são considerados para a seleção dos contadores mais representativos. Vale ressaltar que este período é sobre a execução do processo de análise e tratamento dos dados, sendo incrementado a cada acionamento do mesmo. Isto é feito pela necessidade de que toda janela seja preenchida com os mesmos contadores, pois qualquer valor nulo descarta o contador na preparação dos dados.
  - *remove_priority*: prioridade de remoção dos arquivos que armazenam os grupos gerados para cada iteração. No momento utiliza apenas o tempo como prioridade, selecionando apenas os mais novos.

- *DATA_PROCESS_ANALYSIS*: define as configurações de análise
  - *nullstr*: valor utilizado para a representação de nulos nos conjuntos acumulados pelo acumulador.
  - *method_grouping*: método utilizado no processo de agrupamento. No momento utiliza o método de *pearson*. Como o objetivo deste trabalho é a extensibilidade, outros métodos de regressão também poderão ser adicionados.

- Métodos de agrupamento:
  - GROUPING_PEARSON:
    * group_linkage: algoritmo de agrupamento utilizado. No momento utiliza *Complete linkage*.
    * cut_height: valor do corte utilizado no dendrograma para definição dos grupos.

A seguir são apresentadas as características detalhadas de cada elemento implementado sobre o Serviço de redução de contadores de desempenho, apresentado na Figura 5.

4.1 ACUMULADOR

O Acumulador é o módulo responsável por receber amostras sequenciais do coletor, tanto unicamente quanto em lotes. Este módulo utiliza uma API Web (GRINBERG, 2018) no qual possui dois *endpoints*, que são endereços do serviço responsáveis pelo atendimento de requisições enviadas pelo cliente, neste caso o coletor.

O primeiro *endpoint* é responsável pelo recebimento das amostras de contadores de desempenho, no qual primeiramente lê-se as amostras já armazenadas na aplicação, e concatena com as amostras enviadas. Vale ressaltar que caso seja enviado uma quantidade de amostras que extrapole a quantidade máxima de amostras na janela, a quantidade extrapolada é transferida para a próxima janela que será composta.

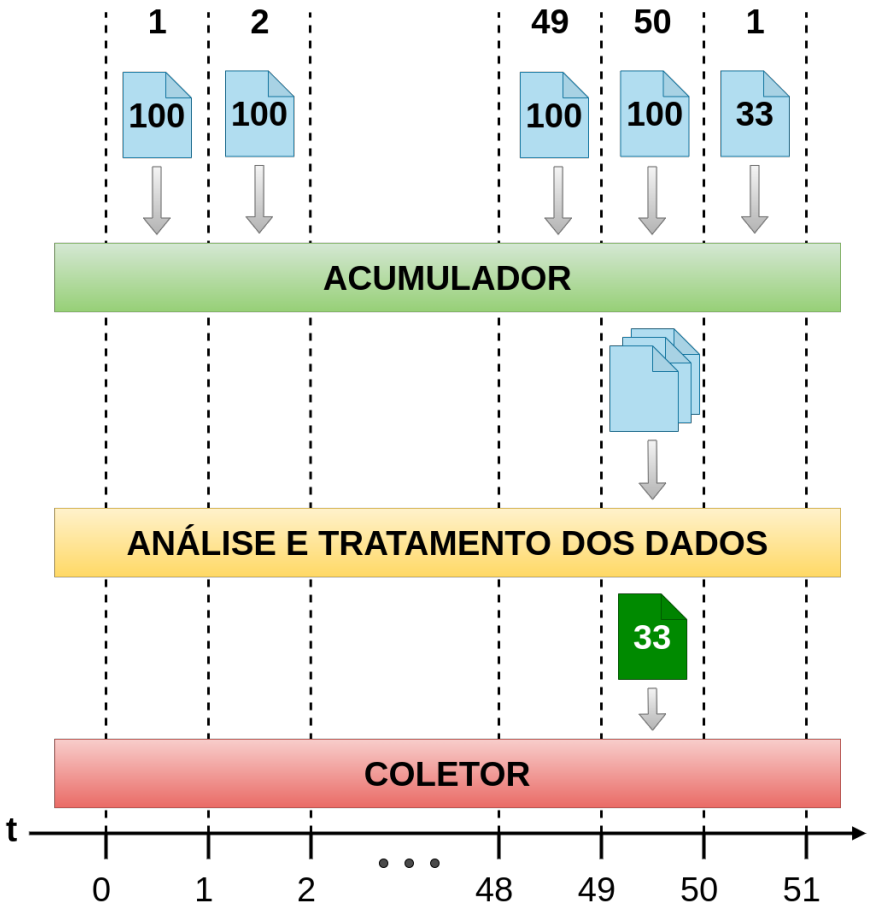
Este módulo possui também controle de concorrência sobre os conjuntos manipulados, como o conjunto de amostras; conjunto de grupos; e conjunto de amostras que estão aguardando a execução do processo de análise, garantindo assim a paralelização com outros processos que utilizam os recursos compartilhados em cada etapa. Também possui o controle sobre o acionamento do processo de análise dos dados ao atingir o número máximo de amostras em um único conjunto.

Caso a funcionalidade de periodicidade de todos os contadores seja ativada, o primeiro envio das amostras para este módulo deve conter todos os contadores que o usuário deseja utilizar, pois são guardados para futuras janelas periódicas. O processo de análise pode ou não ser acionado dependendo do momento no qual o sistema se encontra, sendo que se há periodicidade de todos os contadores, ao chegar no intervalo correspondente são sub-escritos os contadores representativos por todos os contadores contidos na configuração inicial. Portanto, na janela em que todos os contadores são disponibilizados, o coletor lê as métricas do conjunto completo de contadores, e só na próxima janela periódica que será feita a análise do conjunto de amostras da janela anterior.

A Figura 6 demonstra a funcionalidade da utilização de máximo de amostras de acordo com o tempo. Os elementos em azul representam as amostras e a quantidade de contadores de cada amostra enviadas pelo coletor para o acumulador, e acima delas a quantidade de amostras que o acumulador possui armazenada. Já o elemento em verde representa a quantidade de contadores de desempenho que a análise e tratamento dos dados selecionou como representativos para os conjuntos de amostras enviadas anteriormente. Neste exemplo, quando o acumulador atinge 50 amostras, aciona a análise de dados e consequentemente envia 33 contadores selecionados

como representativos para o coletor, sendo estes considerados para a próxima janela de coleta.

Figura 6 – Exemplo de máximo de amostras



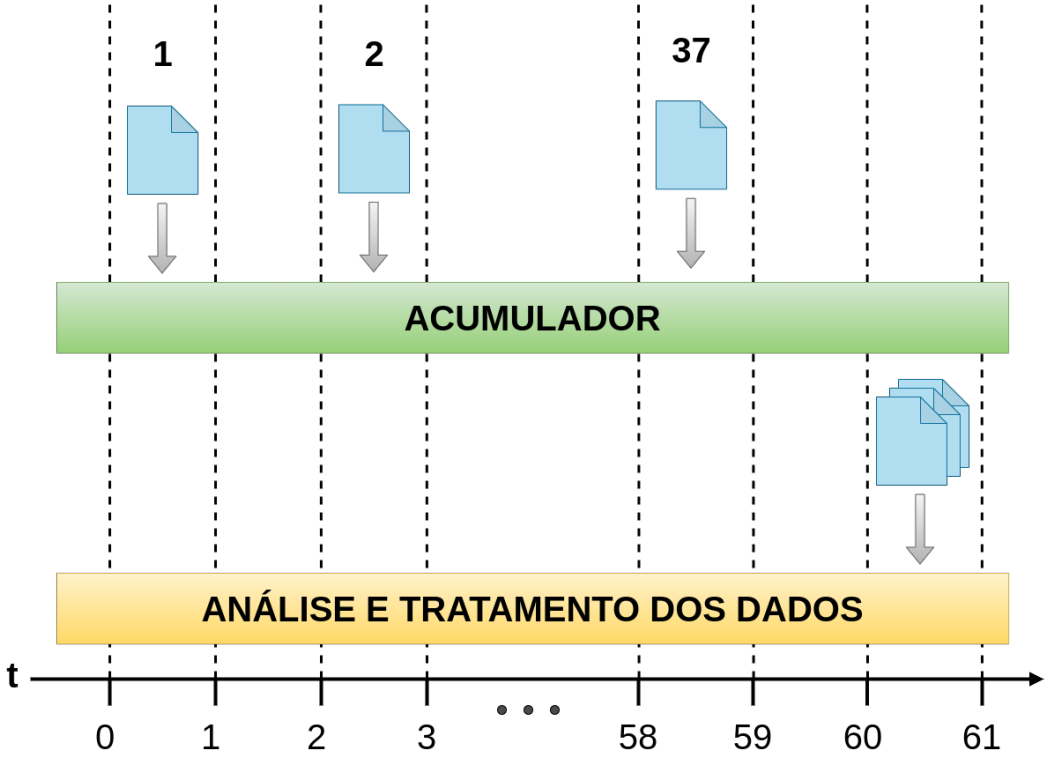
Fonte: Própria

Já a Figura 7 demonstra a funcionalidade de periodicidade de amostras. Em azul é apresentada a amostra enviada pelo coletor, acima dela a quantidade de amostras que o acumulador possui, e no eixo horizontal o tempo em segundos para cada evento. Neste exemplo, depois de 60 segundos o processo periódico envia para análise de dados o conjunto de 37 amostras, pois não houve um máximo de amostras para o acumulador acionar a análise de dados.

A Figura 8 apresenta o funcionamento do parâmetro *max_intersection_window* nas configurações, de acordo com a prioridade de remoção de acordo com o tempo. Neste exemplo no máximo dois conjuntos são levados em consideração para a análise de dados, ou seja, no terceiro período de execução o conjunto verde de amostras foi removido.

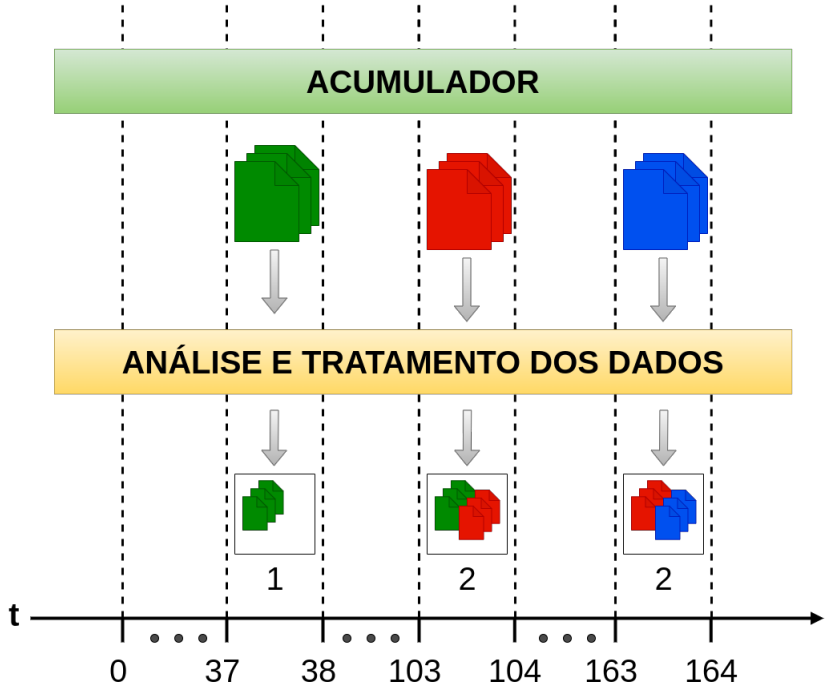
Por fim, a Figura 9 apresenta a funcionalidade de periodicidade de todos os contadores. Elementos em vermelho representam períodos em que todos contadores são considerados, em azul os envios de acordo com a seleção, e em verde a seleção dos contadores mais significativos. Neste exemplo, na primeira janela foi enviado todos

Figura 7 – Exemplo de periodicidade de amostras



Fonte: Própria

Figura 8 – Exemplo de máximo de janelas

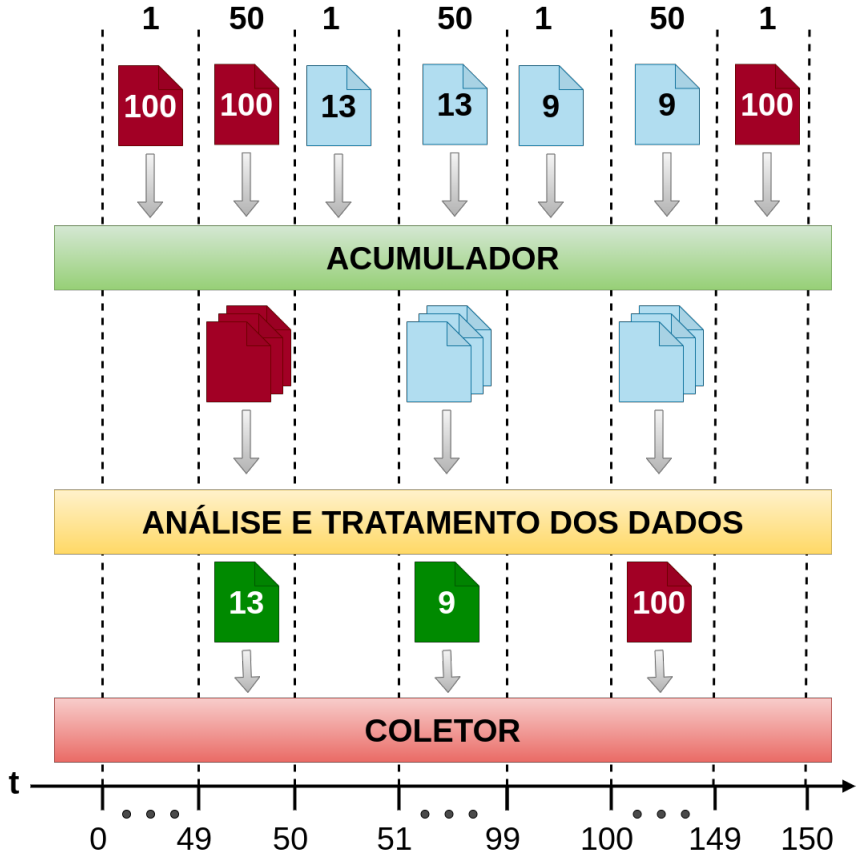


Fonte: Própria

contadores do sistema, e selecionados 13 para o coletor; na próxima foram selecionados 9 e enviados para o coletor; mas na terceira janela, a análise foi ignorada e todos

os contadores foram enviados para o coletor.

Figura 9 – Exemplo de periodicidade de todos os contadores



Fonte: Própria

O segundo *endpoint* é responsável pelo envio dos contadores mais significativos selecionados no momento em que foi solicitado, utilizado pelo coletor.

4.2 ANÁLISE E TRATAMENTO DOS DADOS

O módulo de Análise e tratamento dos dados é responsável pela execução dos conjuntos acumulados de amostras executadas pelo acumulador, realizando o pré-processamento dos contadores; a geração da matriz de dissimilaridade; o agrupamento; e a definição dos grupos, assim armazenando os grupos gerados de cada conjunto em arquivos, para serem utilizados nas próximas etapas. Este processo utiliza como base o algoritmo de seleção de contadores de desempenho disponibilizado em (POPIOLEK, 2018).

4.2.1 Pré-processamento dos dados

Responsável pela limpeza das amostras enviadas pelo acumulador, no qual realiza a eliminação de contadores em que qualquer amostra seja nula. Amostras podem ser nulas por diversos fatores, como contadores que estão indisponíveis no

sistema monitorado; ou em períodos diferentes do período em que foi monitorado, no qual a métrica não está presente no sistema. Também é responsável pela remoção de contadores que não possuem variância, ou que possuem mas que são mínimas, devido à convergência do algoritmo utilizado. Caso a propriedade *devmode* esteja habilitada, serão armazenadas estatísticas da etapa de remoção de contadores onde as amostras eram nulas, para avaliação do usuário. A Figura 10 apresenta alguns *logs* gerados na Avaliação Experimental, apresentando o momento no qual foi gerado o pré-processamento; o conjunto utilizado; o número de contadores do conjunto antes e depois do pré-processamento; a porcentagem de contadores que foram descartados sendo qualquer amostra nula; as variações de quantidades de contadores sobre as amostras para cada contador; e por fim a quantidade de contadores descartados de acordo com a porcentagem de valores nulos em cada contador. Estas informações podem ser utilizadas para verificação de anomalias, como muitos contadores que em poucas amostras apresentavam o valor nulo mas que foram descartados.

Figura 10 – Estatísticas sobre a remoção de contadores por amostras nulas

```
---------- 2022-07-16 00:16:13.562318 ----------
dataset: 1657941372.77166
Number of columns before null elimination: 9941
Number of columns after null elimination: 8156
Percentage of column with any value null: 17.96%
Number of removed columns if all values are null: 1744
Number of removed columns if any value are null: 1785
Number of removed columns where any value are null but not all: 41
Number of removed columns if percent of null is below to 75%: 41
Number of removed columns if percent of null is below to 50%: 29
Number of removed columns if percent of null is below to 25%: 17
Number of removed columns if percent of null is below to 10%: 17
---------- 2022-07-16 00:16:16.750613 ----------
dataset: 1657941376.727816
Number of columns before null elimination: 204
Number of columns after null elimination: 197
Percentage of column with any value null: 3.43%
Number of removed columns if all values are null: 0
Number of removed columns if any value are null: 7
Number of removed columns where any value are null but not all: 7
Number of removed columns if percent of null is below to 75%: 7
Number of removed columns if percent of null is below to 50%: 7
Number of removed columns if percent of null is below to 25%: 7
Number of removed columns if percent of null is below to 10%: 7
```

Fonte: Própria

4.2.2 Mineração dos dados

Esta subseção apresenta todos os processos para a seleção dos contadores representativos dos conjuntos de amostras enviadas.

4.2.2.1 Agrupamento

Método de geração dos grupos de contadores mais significativos para um determinado conjunto de amostras. Composto pela geração da matriz de dissimilaridade, o agrupamento a partir dos algoritmos previamente estabelecidos e a definição dos grupos. Também gera *logs* estatísticos para a avaliação de cada janela de amostras calculada caso o *devmode* esteja habilitado. A Figura 11 apresenta os *logs* gerados para avaliação das janelas, sendo as colunas na ordem: quantidade de contadores depois da preparação dos dados; quantidade de contadores depois do agrupamento; quantidade de contadores depois da intersecção do conjunto anterior com o restante dos conjuntos.

Figura 11 – Características dos grupos para cada iteração sequencial

set_name	counters	counters_after_preparation	counters_after_grouping	intersection
set1	9941	1391	204	-
set2	204	180	150	-
-	-	-	-	groups: 2; aft. int.: 298
set3	298	241	148	-
-	-	-	-	groups: 3; aft. int.: 352
set4	352	289	150	-
set5	9941	1391	205	-
-	-	-	-	groups: 5; aft. int.: 536
set6	536	444	189	-
-	-	-	-	groups: 6; aft. int.: 626
set7	626	498	196	-
-	-	-	-	groups: 7; aft. int.: 683
set8	9941	1263	201	-
set9	683	568	193	-
-	-	-	-	groups: 9; aft. int.: 791
set10	791	650	204	-
-	-	-	-	groups: 10; aft. int.: 845
set11	845	685	181	-
-	-	-	-	groups: 11; aft. int.: 882

Fonte: Própria

Este é um módulo genérico para a geração dos grupos, sendo possível a realização na mudança do método dependendo da aplicabilidade. Nesta versão, foi utilizado o modelo realizado em (POPIOLEK, 2018) para obtenção da matriz de distâncias e definição dos grupos. Para obtenção da matriz de distâncias, primeiro gera-se a matriz de dissimilaridade a partir da correlação de Pearson, e a seguir aplicado a função do coeficiente de distância (POPIOLEK, 2018). Em seguida, é realizado o agrupamento dos dados com o método *Complete Linkage*, utilizando a matriz de distâncias e o tamanho do corte no dendrograma como parâmetros.

Caso o sistema tenha apenas um único conjunto de grupos, no qual foi executado apenas uma janela de amostras, não é necessária a intersecção entre os grupos, sendo assim o próximo passo é a seleção dos contadores mais representativos de cada grupo. Caso contrário, o próximo passo é a verificação de padrões.

4.2.2.2 Verificação de padrões

Caso o sistema tenha executado outras janelas de amostras, a ferramenta realiza a intersecção entre os conjuntos de grupos criados anteriormente (POPIOLEK, 2018), retornando assim um único conjunto de grupos.

4.2.2.3 Seleção dos contadores mais representativos

Método para seleção do contador mais representativo de cada grupo, no qual utiliza a matriz de distâncias para seleção do contador que representa os demais contadores do mesmo grupo (POPIOLEK, 2018).

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo demonstra os benefícios e os meios na utilização da ferramenta proposta, mostrando testes produzidos com diferentes configurações com o objetivo de apresentar as diversas possibilidades na utilização da mesma. A avaliação da ferramenta exige a implementação e utilização de mecanismos para a análise dos contadores selecionados em diferentes cargas de trabalho, sendo que estas cargas podem ser inseridas tanto de forma sequencial em tempo real, quanto por cargas sintéticas que representam um ambiente já simulado.

Os experimentos nesta seção baseiam-se na utilização de *workloads* previamente coletados no trabalho de (POPIOLEK, 2018), sendo realizados com diferentes cargas de trabalho, simulando um ambiente mais controlado, e assim obtendo valores mais precisos sobre os experimentos para a garantia de uma melhor exatidão nas avaliações. O ambiente utilizado como base foi o MySQL Server, e os *workloads* foram gerados a partir do *benchmark Yahoo! Cloud Serving Benchmark* (YCSB). Os *workloads* utilizados foram:

- A: Atualização intensiva, sendo 50% operações de leitura e 50% operações de escrita.
- B: Leitura intensiva, com 95% operações de leitura e 5% operações de escrita.
- C: Apenas leitura, com 100% operações de leitura.

A quantidade de contadores iniciais em cada *workload* se resulta em 9941. Estes contadores são compostos pelos convencionais disponibilizados por *hardware*; e os contadores próprios da aplicação monitorada, como quantidade de acessos à métodos, ou estatísticas sobre o banco de dados. Estes *workloads* foram particionados em tamanhos distintos e enviados gradualmente para o acumulador da aplicação em intervalos que precedem a seleção dos contadores em cada rodada. Cada tamanho representa a quantidade máxima de amostras que um acumulador armazena, sendo que ao ser preenchida completamente, o processo de seleção dos contadores mais representativos é acionado. Assim, as próximas amostragens produzidas pelo coletor simulado serão com os contadores mais representativos selecionados na rodada anterior.

No processo de agrupamento, foram encontradas divergências entre contadores selecionados em testes consecutivos com o mesmo conjunto de amostras e nas mesmas configurações. Estas divergências acontecem devido à ordem no qual os contadores são inseridos na lista de *clusters* no algoritmo de agrupamento, sendo que ao processar o último passo do algoritmo, que é responsável pela seleção dos contadores mais significativos dentro de cada *cluster*, se algum contador estiver com as mesmas similaridades entre outros contadores do mesmo grupo, a escolha do contador será arbitrária.

Foram executados testes com diferentes configurações sobre características específicas da ferramenta, com o intuito de avaliar a eficácia comparado com o algoritmo desenvolvido em (POPIOLEK, 2018), que opera sobre dados estáticos e *offline*, e assim apresentando apenas um conjunto de monitoramento para cada tipo de carga de trabalho.

5.1 COMPARAÇÃO DOS CONTADORES REPRESENTATIVOS SOBRE O CONJUNTO NÃO PARTICIONADO

Como cada conjunto de cargas de trabalho utilizados para o processo de avaliação possuem diferentes tamanhos, para cada conjunto foi realizado a redução no tamanho de amostras em adequação ao tamanho da janela utilizada no acumulador. Também foi utilizado o critério de seleção das amostras sobre o meio do conjunto, descartando as amostras ao extremo do mesmo e assim removendo as que não representavam a carga de trabalho simulada (POPIOLEK, 2018). Sendo assim, cada *workload* foi reduzido até 300 amostras, no qual foram realizados testes com execuções da análise de dados múltiplas da quantidade de amostras, sendo elas representadas na Tabela 2.

Tabela 2 – Tamanho das janelas e quantidade de execuções do conjunto

Tamanho da janela	300	150	100	50	25
Quantidade de execuções	1	2	3	6	12

Assim, primeiramente foram executados os testes sobre o conjunto original, de 300 amostras, representado na Tabela 3, apresentando assim o total de contadores de desempenho representativos e selecionados ao final do experimento. A ideia destes experimentos é avaliar as diferenças entre os contadores selecionados em diferentes tamanhos de janelas comparados com os contadores da janela de tamanho de 300, que é considerada a que seleciona contadores representativos mais realísticos, no qual realiza uma única vez a análise de dados.

Tabela 3 – Contadores finais sobre os conjuntos originais

Workload A	451
Workload B	374
Workload C	324

5.1.1 Experimento 1: testes sem renovação dos contadores mais significativos

Em seguida, foram executados testes sobre as janelas periódicas do acumulador. A Tabela 4 apresenta a quantidade de contadores mais representativos para cada janela e para cada conjunto de carga de trabalho utilizada, representado pelo *workload*. Vale lembrar que este teste desconsidera os contadores representativos para cada envio, com o intuito de maximizar a semelhança com o conjunto original, sendo apresentados no texto como testes sem renovação. É possível observar que com a diminuição da janela, a quantidade de contadores representativos também aumenta. Isso acontece devido ao pré-processamento dos dados e pelo processo de verificação de dissimilaridade entre os contadores, no qual como cada conjunto é visto separadamente, novos contadores que foram descartados nestes processos começaram a ter relevância.

Já a Tabela 5 apresenta as intersecções dos grupos de contadores mais representativos que foram selecionados na execução dos testes com o conjunto original, de acordo com a configuração de máximo de amostras do acumulador indicado na tabela. Como a amostra de referência é o *workload* inteiro, que representa 300 amostras, a intersecção da janela de tamanho 300 resulta no conjunto original.

Tabela 4 – Contadores finais após todas as rodadas sem renovação

Tamanho da janela	300	150	100	50	25
Workload A	451	576	632	698	747
Workload B	374	529	595	664	703
Workload C	324	379	407	450	499

Tabela 5 – Intersecção entre os conjuntos de teste sem renovação e o conjunto completo

Tamanho da janela	300	150	100	50	25
Workload A	451	348	347	343	354
Workload B	374	274	264	268	274
Workload C	324	242	234	238	235

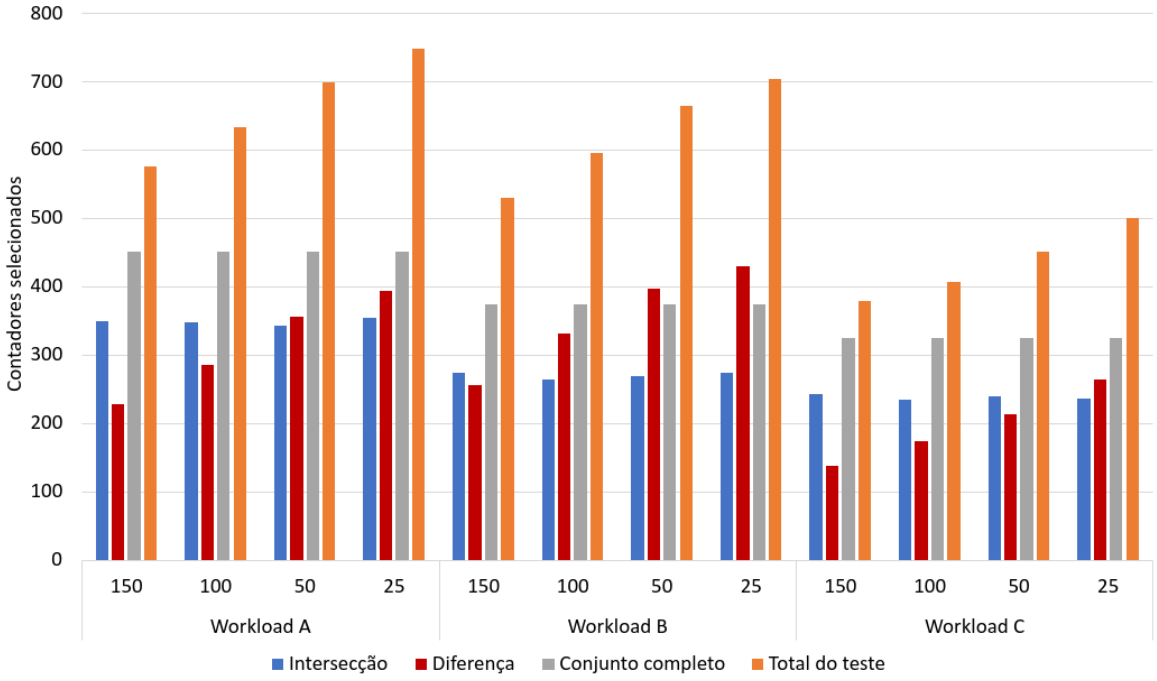
A Figura 12 apresenta a composição dos valores dos experimentos nos diferentes *workloads* utilizados, exibindo assim as características de cada janela sem utilizar os contadores mais significativos em cada iteração e sem utilizar a periodicidade de todos os contadores. No eixo vertical é apresentado a quantidade de contadores selecionados para cada conjunto. No eixo horizontal são apresentados em ordem o *workload*

utilizado e o tamanho da janela correspondente. Para cada *workload*, são apresentados os conjuntos:

- Conjunto completo: Conjunto dos contadores representativos do tamanho da janela de 300.
- Total do teste: Conjunto dos contadores representativos do *workload* e da janela em questão.
- Intersecção: Intersecção entre o conjunto completo e total do teste.
- Diferença: Diferença entre o conjunto completo e total do teste.

Podemos inferir que a quantidade de contadores de desempenho que são realmente representativos, representados em azul, continuam a mesma conforme a redução do tamanho da janela. Contudo, houve um aumento significativo na quantidade de contadores representativos que foram identificados nas janelas de teste, e que não foram identificados na janela de 300.

Figura 12 – Semelhanças entre os conjuntos de contadores sem renovação e sem periodicidade de todos os contadores



Fonte: Própria

5.1.2 Experimento 2: testes com renovação dos contadores mais significativos

Para a execução dos testes com renovação dos contadores mais significativos de cada rodada, foram utilizadas as mesmas janelas dos testes anteriores. A Tabela 6

apresenta os testes realizados sobre os conjuntos de teste, com a renovação dos contadores mais significativos para cada rodada. Ou seja, na primeira janela são enviados todos os contadores, já nas próximas são enviados os contadores representativos em relação à rodada anterior. Foi possível observar que a quantidade de contadores significativos pouco diminuíram em relação ao teste sem renovação, totalizando mínimos abaixo de 5% e máximos de 20%, dependendo do *workload* avaliado.

Tabela 6 – Contadores finais após todas as rodadas com renovação

Tamanho da janela	300	150	100	50	25
Workload A	451	505	541	598	592
Workload B	374	454	503	586	631
Workload C	324	362	400	436	416

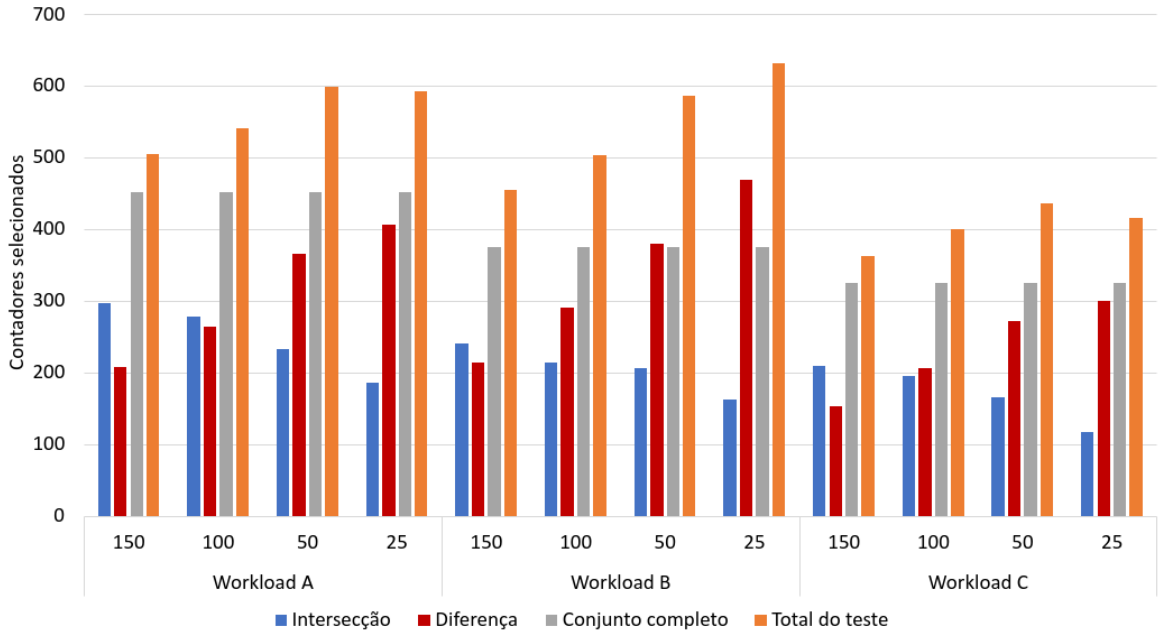
Tabela 7 – Intersecção entre os conjuntos com renovação e o conjunto completo

Tamanho da janela	300	150	100	50	25
Workload A	451	297	278	233	186
Workload B	374	240	213	206	162
Workload C	324	209	194	165	116

Para avaliar a eficácia da solução, devemos verificar a intersecção dos contadores representativos de cada experimento comparado aos conjuntos originais. A Tabela 7 apresenta as intersecções dos contadores com renovação em relação aos conjuntos originais de cada *workload*. Observou-se que a quantidade de contadores diminuiu comparados ao teste anterior.

A Figura 13 apresenta a composição dos valores dos experimentos com renovação, para a melhor visualização da disparidade de contadores significativos que foram selecionados, comparados ao conjunto original. É possível observar a partir da intersecção, representada em azul, que poucos contadores são semelhantes ao conjunto completo em diferentes tamanhos de janela, resultando numa pior qualidade pois, conforme se reduz o tamanho da janela, os contadores que eram considerados representativos e que não estavam na janela de 300 são levados mais em consideração para a seleção de futuras rodadas de execução dos métodos de similaridade e agrupamento. Com isso, o aumento da quantidade de contadores diferentes também aumentaram conforme o crescimento do tamanho da janela, resultando assim em extremos no qual pequenas janelas apresentam o dobro de contadores diferentes do conjunto completo, em vermelho, comparados com contadores que estavam no conjunto completo, em azul.

Figura 13 – Semelhanças entre os conjuntos de contadores com renovação e sem periodicidade de todos os contadores



Fonte: Própria

5.1.3 Experimento 3: testes com renovação dos contadores mais significativos e com periodicidade de todos os contadores

O próximo experimento tem como objetivo avaliar a execução da aplicação utilizando a configuração de reutilização de todos os contadores de desempenho, com o intuito de minimizar o erro dos contadores comparados ao conjunto original, sem o custo associado de cálculo de todos os contadores disponíveis em cada iteração. Como os valores periódicos de alimentação com todos os contadores podem ser múltiplos do tamanho das janelas de cada *workload*, é possível que no final do experimento todos os contadores do sistema sejam significativos. Para contornar esta informação, foi habilitado o mecanismo de periodicidade de execução com mínimo de amostras, utilizando um período suficientemente grande para que sejam executados todos os experimentos e logo em seguida seja executado o processamento via periodicidade. A Tabela 8 apresenta a quantidade de vezes para cada janela de experimento que todos os contadores do sistema foram utilizados. Como não há execuções suficientes para janelas maiores para a avaliação, estas foram descartadas pois se assemelham aos testes anteriores com renovação. Vale ressaltar que os experimentos foram executados apenas sobre um *workload* devido à quantidade de características dos testes.

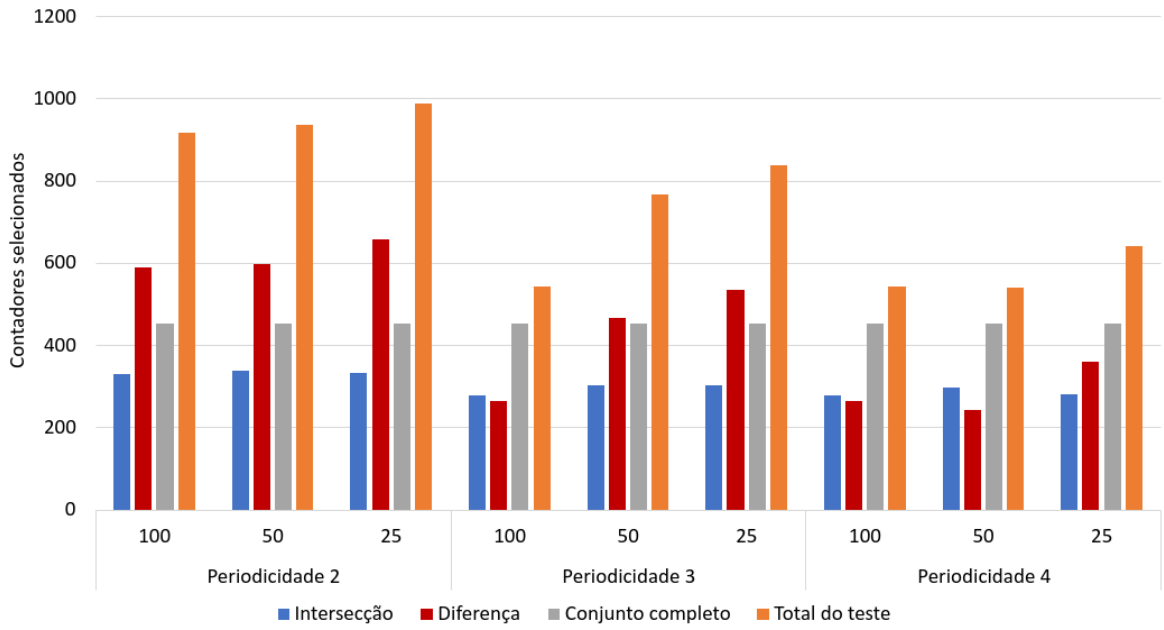
A Figura 14 apresenta a composição dos experimentos com renovação e com frequência de recebimento de todos os contadores disponíveis no sistema. Podemos observar que houve uma melhor quantidade de contadores de desempenho selecionados corretamente, cerca de 10%, comparados com os contadores no experimento com

Tabela 8 – Quantidade de vezes que todos contadores foram acionados

Tamanho da janela	100	50	25
Periodicidade 2	2	3	6
Periodicidade 3	1	2	4
Periodicidade 4	1	2	3

apenas renovação. Também, em periodicidades maiores, a quantidade de contadores corretos, representados em azul, perdem pouca precisão comparados com períodos menores, continuando ainda com menores contadores diferentes do conjunto completo, em vermelho.

Figura 14 – Semelhanças entre os conjuntos de contadores com renovação e periodicidade de todos os contadores



Fonte: Própria

Assim, existem diferentes abordagens que podem ser adotadas para a garantia de uma melhor precisão na seleção dos contadores mais representativos, no qual vai depender do sistema em que está sendo utilizado. Janelas de tamanho grande apresentam uma melhor eficiência na seleção pois executam menos vezes a análise dos dados, entretanto podem não representar o sistema quando há mudanças repentinas de comportamento. Janelas de tamanho pequeno apresentam uma menor eficiência na seleção, contudo oferecem contadores representativos mais rapidamente, sendo assim viáveis em sistemas mais dinâmicos. E por fim, periodicidades de todos os contadores melhoram a qualidade dos contadores representativos, entretanto não são tão eficientes em periodicidades pequenas, pois aumentam muito a quantidade de conta-

dores que não eram identificados como representativos, e também podem aumentar o *overhead* quando utilizados em pequenas periodicidades ou grandes janelas.

6 CONSIDERAÇÕES FINAIS

Contadores de desempenho estão cada vez sendo mais utilizados, principalmente e majoritariamente em sistemas distribuídos, no qual prioriza-se muito tanto a robustez quanto performance dos sistemas hospedados. Assim, a utilização de contadores de desempenho no sistema é imprescindível para o monitoramento de falhas; a visualização de inconsistências; assim como a visualização e monitoramento na utilização dos componentes no sistema. Este trabalho descreve os fundamentos, a metodologia e a implementação de um serviço que seja adaptável, extensível, configurável e adequável para a integração com outras ferramentas de coletas de métricas do sistema, priorizando o desempenho no sistema ao todo. Como visto, foi necessário o conhecimento de diversas técnicas de modularização e configuração, para a garantia de robustez, legibilidade e adaptabilidade do sistema para diferentes modos, caso sejam adicionados em trabalhos futuros. Também foram utilizados conhecimentos em Ciência de Dados para avaliação dos padrões resultantes de cada experimento, para análise dos parâmetros aplicados nas configurações da ferramenta.

Nos experimentos, foi possível identificar que a quantidade de contadores de desempenho que a ferramenta selecionou ficou extremamente inferior à quantidade de contadores de desempenho no sistema ao todo. Os experimentos também indicaram as características de cada configuração, mostrando que é possível realizar a avaliação sobre diferentes configurações para uma melhor aplicabilidade no sistema. Assim, foi possível notar que pequenas janelas de amostras tornam os dados enviesados, crescendo assim a quantidade de contadores que não representam o sistema ao todo, mas grandes janelas não são tão aplicáveis em diferentes sistemas. Também, foi notado que janelas com renovação, que são as que representam a aplicação, possuem o custo na diminuição da previsibilidade de contadores que representavam o sistema, assim como a maior quantidade de contadores que não representam o sistema. Por fim, utilizando periodicidades de todos os contadores, a quantidade de contadores que realmente são representativos aumenta, mas com o custo de processamento de todos os contadores em períodos estipulados.

6.1 TRABALHOS FUTUROS

Este trabalho foi realizado utilizando mecanismos que facilitam a incorporação de ambientes baseados em microsserviços, pois coletores de métricas são utilizados em conjunto com sistemas de grande escala. Entretanto, não foram utilizadas ferramentas como Kubernetes e Docker Swarm para orquestração do serviço implementado junto com serviços de coletores de métricas, em um sistema realístico. Um próximo passo seria a implementação numa arquitetura de microsserviços, utilizando *Benchmarks* para a visualização da eficácia e escalabilidade da ferramenta nestes ambientes.

Também, nestes mesmos ambientes, é interessante a análise de comportamento do sistema em tempo real, dado que os experimentos realizados neste trabalho foram gerados à partir de amostras sintéticas já processadas.

Como a ideia deste trabalho foi o desenvolvimento de uma aplicação adaptável para diferentes técnicas de mineração de dados, seria interessante realizar experimentos com outras estratégias de agrupamento para avaliação dos contadores obtidos, assim como a utilização de algoritmos adaptativos de valor de corte do dendrograma gerado pelo agrupamento.

As análises deste trabalho identificam a quantidade de contadores que foram selecionados, mas não faz a verificação exata da qualidade dos grupos de contadores e os contadores de desempenho representativos gerados. Um próximo passo seria o estabelecimento de mecanismos de avaliação das configurações da ferramenta utilizando as métricas de validação de agrupamento na Seção 2.3.4. Isso possibilitaria a análise dos algoritmos utilizados, assim como as propriedades mais básicas de configuração, como o tamanho das janelas e a periodicidade de todos os contadores.

REFERÊNCIAS

AGARWAL, Shivam. **Data mining: Data mining concepts and techniques**. [*S.l.*: *s.n.*], 2014. P. 203–207. ISBN 9780769550138. DOI: 10.1109/ICMIRA.2013.45.

BECKER, Alex Malmann; LUCRÉDIO, Daniel. The impact of microservices on the evolution of a software product line. *In*: p. 51–60. DOI: 10.1145/3425269.3425275.

FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. The KDD process for extracting useful knowledge from volumes of data. **Communications of the ACM**, ACM New York, NY, USA, v. 39, n. 11, p. 27–34, 1996.

FELTER, Wes; FERREIRA, Alexandre; RAJAMONY, Ram; RUBIO, Juan. An updated performance comparison of virtual machines and Linux containers. *In*: p. 171–172. DOI: 10.1109/ISPASS.2015.7095802.

GREENWOOD, Garrison W.; TYRRELL, Andy M. **Introduction to evolvable hardware : a practical guide for designing self-adaptive systems**. [*S.l.*]: Wiley-IEEE Computer Society, 2007. P. 192. ISBN 9780471719779.

GRINBERG, Miguel. **Flask web development: developing web applications with python**. [*S.l.*]: "O'Reilly Media, Inc.", 2018.

IBM. Hypervisors. **Cloud Education**, 2019. Disponível em: https://www.ibm.com/br-pt/cloud/learn/hypervisors.

LEWIS, James; FOWLER, Martin. Microservices: A definition of this new architectural term, 2014. Disponível em: https://martinfowler.com/articles/microservices.html.

MICROSOFT. **PerfMon**. [*S.l.*: *s.n.*], jun. 2022. Disponível em: https://docs.microsoft.com/pt-br/windows-server/administration/windows-commands/perfmon.

NOGUEIRA, Mauro Lúcio Baioneta; YAMIN, Adenauer Corrêa; VARGAS, Patrıcia Kayser; GEYER, Cláudio Fernando Resin. Balanceamento de Carga em Sistemas Distribuıdos: Uma Proposta de Ambiente para Avaliação. *In*: CONFERÊNCIA LATINOAMERICANA DE INFORMÁTICA. [*S.l.*: *s.n.*], 2001.

NOWAK, Andrzej; BITZES, Georgios. The overhead of profiling using PMU hardware counters. **Openlab.Web.Cern.Ch**, CERN Openlab Report 2014, 2014.

PAHL, Claus. Containerization and the PaaS Cloud. **IEEE Cloud Computing**, Institute of Electrical e Electronics Engineers Inc., v. 2, p. 24–31, 3 mai. 2015. ISSN 23256095. DOI: 10.1109/MCC.2015.51.

PCP. **Performance Co-Pilot**. [*S.l.*: *s.n.*], jun. 2022. Disponível em: https://pcp.io/.

PES, Gabriel. **Serviço de monitoramento de contadores de desempenho com baixa sobrecarga**. [*S.l.*: *s.n.*], jun. 2022. Disponível em: https://github.com/biepes/reducer-tool.

POPIOLEK, Pedro Freire. Redução de sobrecarga de monitoramento em ambientes virtualizados através de contadores de desempenho, p. 1–89, 2018.

POPIOLEK, Pedro Freire; MACHADO, Karina dos Santos; MENDIZABAL, Odorico Machado. Low overhead performance monitoring for shared infrastructures. **Expert Systems with Applications**, Elsevier Ltd, v. 171, December 2020, p. 114558, 2021. ISSN 09574174. DOI: 10.1016/j.eswa.2020.114558. Disponível em: https://doi.org/10.1016/j.eswa.2020.114558.

POPIOLEK, Pedro Freire; MENDIZABAL, Odorico Machado. Monitoring and analysis of performance impact in virtualized environments. **Journal of Applied Computing Research**, v. 2, n. 2, p. 75–82, 2013. ISSN 2236-8434. DOI: 10.4013/jacr.2012.22.03.

RAMESHAN, Navaneeth. **On the Role of Performance Interference in Consolidated Environments**. [*S.l.*: *s.n.*], 2016. P. 128. ISBN 978-91-7729-113-8 (ISBN). Disponível em: http://kth.diva-portal.org/smash/get/diva2:974058/FULLTEXT01.pdf%0Ahttp://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-192960.

SCIKIT-LEARN. **Clustering methods**. [*S.l.*: *s.n.*], jun. 2022. Disponível em: https://scikit-learn.org/stable/modules/clustering.html.

SHANG, Weiyi; HASSAN, Ahmed E; NASSER, Mohamed; FLORA, Parminder. Automated detection of performance regressions using regression models on clustered

performance counters. *In*: PROCEEDINGS of the 6th ACM/SPEC International Conference on Performance Engineering. [*S.l.*: *s.n.*], 2015. P. 15–26.

SKIENA, Steven S. **The data science design manual**. [*S.l.*: *s.n.*], 2017. P. 1–445. ISBN 978-3-319-55443-3. DOI: 10.1007/978-3-319-55444-0. Disponível em: http://link.springer.com/10.1007/978-3-319-55444-0.

STEELE, Brian; CHANDLER, John; REDDY, Swarna. **Algorithms for data science**. [*S.l.*]: Springer, 2016.

TAN, P. N.; STEINBACH, M.; KUMAR, V. Data mining cluster analysis: basic concepts and algorithms. **Introduction to data mining**, v. 487, n. 533. 2013.

XIA, Yubin; LIU, Yutao; CHEN, Haibo; ZANG, Binyu. CFIMon: Detecting violation of control flow integrity using performance counters. **Proceedings of the International Conference on Dependable Systems and Networks**, IEEE, 2012. DOI: 10.1109/DSN.2012.6263958.

YAO, Kundi; B. DE PÁDUA, Guilherme; SHANG, Weiyi; SPOREA, Steve; TOMA, Andrei; SAJEDI, Sarah. Log4Perf, p. 127–138, 2018. DOI: 10.1145/3184407.3184416.

APÊNDICE A – SERVIÇO DE MONITORAMENTO DE CONTADORES DE DESEMPENHO

A última versão do serviço desenvolvido neste trabalho está disponível em https://github.com/biepes/reducer-tool.

APÊNDICE B – ARTIGO NO FORMATO SBC

Implementação de um serviço de monitoramento de contadores de desempenho com baixa sobrecarga

Gabriel R. C. G. Pes[1], Odorico M. Mendizabal[1], Karina dos Santos Machado[2]

[1]Departamento de Informática e Estatística (INE) - Universidade Federal de Santa Catarina (UFSC)
Florianópolis, SC – Brazil

[2]Universidade Federal do Rio Grande (FURG) – Rio Grande, RS – Brazil

gabriel.pes@grad.ufsc.br, odorico.mendizabal@ufsc.br, karina.machado@furg.br

Abstract. *Performance counters are useful to predict system resource usage to detect bottlenecks, check system availability, and detect system anomalies. However, in many systems, the use of large-scale meters generates an impact on performance and energy consumption, which are often undesirable because they impair the efficiency of the services available to the system. Microservices architectures are becoming more and more popular in different systems, given the ease of creating and maintaining services without the need to depend on other services already hosted on the same physical server. Virtual machines divide system resources among themselves and can generate inefficiency when they compete for the resources on which they are hosted. The use of virtual machines when on a large scale is often used in conjunction with performance metrics collection services. Therefore, in these environments where several applications run in parallel, monitoring with performance counters in these applications can become costly if not managed with care. This work involves the creation of an integration tool with performance metrics collection services, which reduces the impact that performance counters generate in relation to the host system using data science techniques developed in other works. It also compares the different parameters that can be passed on to this tool. The results include an adaptable, extensible, configurable and easy-to-implement tool that integrates the user metrics collector and automatically reports a reduced set of performance counters compared to the counters originally used.*

Resumo. *Contadores de desempenho são úteis para prever a utilização de recursos no sistema para detecção de gargalos, verificar a disponibilidade do sistema, além de detectar anomalias presentes no mesmo. Entretanto, em muitos sistemas a utilização de contadores em grande escala geram um impacto no desempenho e no consumo de energia, que muitas das vezes são indesejáveis pois prejudicam a eficiência dos serviços disponíveis ao sistema. Arquiteturas de microsserviços estão cada vez se tornando mais populares em diversos sistemas, dado a facilidade de criação e manutenção de serviços sem a necessidade de dependência dos demais serviços já hospedados no mesmo servidor físico. Máquinas virtuais dividem entre si os recursos do sistema e podem gerar uma ineficiência quando disputam os recursos na qual estão hospedadas. O uso de máquinas virtuais quando em grande escala é frequentemente utilizado em conjunto com serviços de coletas de métricas de desempenho. Portanto, nestes ambientes em que diversas aplicações são executadas em paralelo, o monitoramento com contadores de desempenho nestas aplicações pode se tornar custoso se não administrado com cautela. Este trabalho envolve a criação de uma ferramenta de integração com serviços de coleta de métricas de desempenho, que reduz o impacto que os contadores de desempenho geram em relação ao sistema hospedeiro utilizando*

técnicas em ciência de dados desenvolvidas em outros trabalhos. Também, realiza a comparação dos diferentes parâmetros que podem ser passados sobre esta ferramenta. Os resultados incluem uma ferramenta adaptável, extensível, configurável e de fácil implementação, que integra o coletor de métricas do usuário, e que informa automaticamente um conjunto reduzido de contadores de desempenho em comparação com os contadores originalmente utilizados.

1. Introdução

Com o crescimento significativo de aplicações mais complexas de conteúdo geral, a utilização de microsserviços se tornou fundamental, pois tem como premissa a facilidade de replicação de sistemas escaláveis e a diminuição da dependência sobre o sistema hospedeiro (Lewis, Fowler, 2014). Com o crescimento da indústria de hardware, cada vez mais serviços virtuais são suportados em servidores físicos. Contudo, a complexidade de gerenciamento de recursos nesses servidores físicos aumenta quando submetida a diferentes cargas de trabalho oriundas de máquinas virtuais. Assim, são necessárias estratégias para otimização de desempenho e o provisionamento de recursos de forma mais eficiente dentro de sistemas baseados em microsserviços.

Constantemente a necessidade do monitoramento de sistemas baseados em microsserviços cresce, desde a avaliação de métricas de propósitos mais gerais como desempenho e consumo, até a avaliação de consequências relacionadas à estas métricas, como por exemplo a detecção de malwares no sistema (Xia et al., 2012). Assim, a utilização de contadores de desempenho disponibilizados em hardware e software são importantes para a coleta de dados quando utilizadas em aplicações que visam otimização de desempenho, assim como funcionalidades diversas dependentes do monitoramento de recursos (Shang, Hassan, 2015).

Entretanto, ao contrário de sistemas monolíticos que necessitam de apenas um contador de métricas centralizado, sistemas baseados em microsserviços necessitam muitas das vezes contadores relacionados ao serviço em questão, independentes dos demais serviços executados no sistema. Com isso, diversos sistemas possuem grandes quantidades de contadores de desempenho, porém possuem uma sobrecarga envolvida (Popiolek et al., 2013; Rameshan, 2016). Esta sobrecarga é gerada quando há disputa de recursos no sistema conforme o crescimento na utilização dos contadores (Popiolek et al., 2013). Como o objetivo do uso dos contadores no cenário proposto é a avaliação de desempenho e consumo, é necessário estabelecer um balanceamento entre a quantidade de contadores sem prejudicar na avaliação dos mesmos.

A utilização de técnicas relacionadas à ciência de dados é uma prática comum para o tratamento de problemas que envolvem grandes quantidades de dados (Skiena, 2017), que visam a extração de conhecimento; a detecção de padrões; ou a percepção de soluções para determinado problema. Existem na literatura métodos que abordam técnicas em ciência de dados aplicados a amostras de contadores de desempenho, tendo como objetivo tratar contadores que não apresentam dados relevantes e que possuem características comuns entre si, permitindo assim a redução da quantidade de contadores no sistema sem reduzir a integridade das métricas em análise (Popiolek et al., 2021). Neste trabalho pretende-se aplicar as técnicas disponíveis na literatura com o objetivo de

diminuição no número de contadores de desempenho do sistema em nível de aplicação, em um serviço que visa a extensibilidade, escalabilidade e fácil manuseio.

2. Trabalhos relacionados

Esta seção apresenta os principais trabalhos relacionados que expõem métodos úteis para a redução de sobrecarga de contadores de desempenho no sistema.

2.1 Redução de sobrecarga

A utilização de métodos manuais e dependentes da experiência e intuição do profissional são amplamente utilizados (Shang, Hassan, 2015). Entretanto, existe uma dificuldade na manipulação de múltiplos contadores de desempenho normalmente em sistemas de grande escala, além da sobrecarga gerada no monitoramento de contadores de desempenho em microsserviços. Dessa forma, alguns trabalhos apresentam alternativas com métodos automáticos e/ou inteligentes para a redução de forma eficiente e robusta.

Além disso, estes trabalhos possuem uma técnica comum de mineração de dados, sendo Shang e Hassan (2015) o primeiro a aplicar em contadores de desempenho. Esta técnica se resume no agrupamento de contadores de desempenho por meio da separação em clusters, e em seguida realizando a manipulação do mesmo de acordo com cada propósito diferente.

2.1.1 Detecção de regressões de desempenho

Em (Shang, Hassan, 2015), é proposto um método para a detecção de regressões de desempenho por meio de modelos de regressão aplicados em dados de monitoramento, obtidos por conjuntos de contadores de desempenho.

A metodologia proposta é dividida em etapas. Primeiro, é realizada uma análise detalhada do conjunto de contadores disponibilizados no sistema, com o objetivo de remoção de contadores que são redundantes e que não apresentam variância, assim como a remoção de contadores cujos valores podem ser obtidos de forma direta por outros contadores. Em seguida é feito o agrupamento dos contadores restantes, separando em clusters de acordo com as similaridades entre eles. Assim que são criados, para cada cluster em específico é feita a eleição de um contador de desempenho, por meio de testes de distribuição de probabilidade nos conjuntos de contadores de cada cluster. Por fim, é construído, para cada cluster, um modelo linear no qual os contadores restantes representam as variáveis dependentes e, a partir destes modelos, é feita a análise destes resultados para definir qual ou quais clusters são representativos para modelos de regressão de desempenho no sistema alvo.

Foram realizados testes em um benchmark de suporte médio e numa aplicação consideravelmente grande em processamento e contadores disponibilizados. Dentre os resultados indicam que houve uma drástica redução no número de contadores quando removido os contadores identificados redundantes. No benchmark de suporte médio, dos 28 contadores de desempenho, apenas 2 foram considerados. Já na aplicação

consideravelmente maior, a quantidade de contadores foram reduzidos a apenas 4, sendo assim possível concluir que houve uma grande redução conforme o crescimento do sistema.

Portanto, (Shang, Hassan, 2015) é de grande relevância a este trabalho, justamente ao propor um método automático de redução de contadores de desempenho, de forma eficiente e que escala muito bem conforme a quantidade de contadores.

2.1.2 Detecção de trechos críticos de código

Yao et al. (2018) propõe uma alternativa para complementar a avaliação de performance no sistema por meio de adição de contadores específicos em trechos críticos de código em ambientes Web. Neste trabalho, é explicado sobre a dificuldade de análise de desempenho no servidor alvo, sendo necessária experiência e intuição para seleção dos contadores mais representativos ou, em situações mais críticas, o conhecimento do código fonte para a adição manual de contadores. Portanto, a solução adotada adiciona trechos críticos de código sugeridos para o usuário de forma automática, com contadores previamente utilizados, e podendo ser descartados posteriormente.

A proposta é dividida em etapas. O primeiro passo é a identificação das requisições que geram mudanças no desempenho do sistema, por meio do monitoramento tanto das requisições quanto dos contadores internos no sistema. A partir dos logs gerados é executado o cruzamento dos timestamps para cada log de requisição e de desempenho, e aplicado posteriormente para cada cruzamento um método similar apresentado por Shang e Hassan (2015) para a avaliação do quão estatisticamente significante é o modelo de regressão gerado pelas métricas coletadas. O segundo passo é a identificação dos métodos no código fonte significantes para a avaliação de desempenho. Desse modo, é adicionado para cada método associado ao endpoint daquela requisição um contador, e novamente é feita a execução do sistema por meio de testes. Com estes logs gerados, remove-se os contadores que não apresentaram variações ou que pertenciam a métodos que eram chamados por outros em sequência, na qual possuíam também contadores alocados. No terceiro e último passo, é feita a identificação de cada sub-bloco dos métodos já identificados influentes, para a obtenção de uma melhor exatidão na identificação dos trechos mais críticos.

Para a avaliação do método proposto, foram realizados testes em três softwares diferentes, sendo um deles próprio do autor e relativamente maior em comparação ao processamento e a quantidade de contadores disponibilizados. Os resultados indicam uma melhor explicação de desempenho do sistema em relação aos habitualmente utilizados, mas com poucos ganhos em relação à predição. Entretanto, o principal ganho refere-se à localização dos trechos do código com maior impacto, no qual a metodologia conseguiu atingir apenas 0.5% da quantidade de métodos no sistema, sendo melhor para a identificação dos trechos de código que geraram mais gargalos de desempenho no sistema.

Portanto, esta metodologia propõe uma técnica útil caso seja abordado uma redução dos contadores gerais do sistema, entretanto pode ser utilizada apenas para ambientes em que o propósito é a avaliação de desempenho do sistema como um todo. Diferente de nossa abordagem, que visa a redução de sobrecarga de monitoramento para ambientes de propósito geral.

2.1.3 Redução de sobrecarga em ambientes virtualizados

Popiolek et al. (2021) tem como objetivo reduzir a sobrecarga de desempenho no uso de contadores de desempenho em ambientes virtualizados, dado que são os mais afetados por conta da competitividade de recursos no sistema ao todo. A metodologia proposta é a redução de contadores de desempenho, selecionando os que são redundantes no determinado sistema virtualizado.

O método consiste na aplicação de um processo iterativo de extração de informações nos dados coletados uma única vez para a identificação dos contadores não representativos, no qual se divide em passos. Primeiro, realiza-se o processo de experimentação, no qual é feita a partir da injeção de diferentes cargas sintéticas no ambiente, assim tendo uma maior dispersão nos valores dos contadores coletados. Em seguida é realizado o processo de seleção, que se dá na remoção das métricas coletadas dentre um intervalo no início e no fim do experimento, removendo-se cargas coletadas em momentos de aquecimento e resfriamento do sistema, para que o processo de avaliação seja fiel à um sistema que esteja rodando normalmente. No próximo passo é realizado o pré-processamento destes dados, removendo-se contadores no qual não observou-se variância nas métricas coletadas para o determinado intervalo de tempo, e também contadores sem métricas coletadas devido à diversos fatores. A partir do pré-processamento, é realizada a mineração destes dados, que se dá pelo método similar ao proposto por Shang e Hassan (2015) para o agrupamento dos contadores em clusters. Em seguida é realizado o processo de descoberta de conhecimento, que consiste na separação de dois grupos de clusters, um com os da iteração atual e o outro com os da seleção de iterações passadas. Essa separação se dá para a identificação de correlações em independentes cenários no momento que foi gerado as cargas sintéticas no ambiente. Com isso, caso ainda tenha experimentos a serem injetados no sistema, o processo volta à primeira etapa, se não segue para a próxima. Por fim é executado o processo de seleção de um contador de desempenho mais representativo para cada cluster gerado na etapa anterior, a partir da avaliação da dissimilaridade dos contadores de cada cluster.

Para a avaliação, este trabalho segue dois objetivos, primeiro avaliar se houve uma redução em relação à sobrecarga no sistema utilizando apenas os contadores selecionados neste método, e segundo avaliar se os contadores selecionados para cada cluster é capaz de caracterizar os diferentes estados do ambiente no qual está sendo avaliado. Foram utilizados microbenchmarks para geração de cargas controladas e intensivas e macrobenchmarks para simulação de sistemas reais. A avaliação foi realizada pela análise da vazão de operações concluídas por estes benchmarks comparando a utilização do método proposto.

Dos resultados na redução de sobrecarga, ambos cenários se destacam. Nos macrobenchmarks a sobrecarga com o monitoramento completo ativado totalizava no pior caso 30% para operações de leitura e 20% leitura/escrita, e reduzindo-se para apenas 10% e 5% respectivamente. Entretanto, a sobrecarga nestes experimentos se dá justamente pela capacidade de vazão do sistema, sendo assim a sobrecarga gerada por meio do método proposto fica imperceptível comparada com a não utilização de contadores de desempenho, totalizando em média apenas 0.77%.

Em relação à avaliação dos conjuntos de contadores, em microbenchmarks são executados os mesmos tipos de instruções em relação à uma carga específica avaliada, com isso todos os testes as quantidade de contadores ficam próximas da média de 3.8% em relação ao número total de contadores do sistema. Já para macrobenchmarks, devido ao benchmark utilizado, as cargas geradas para cada teste são cumulativas, sendo assim as seguintes serão reflexos das anteriores. Portanto foram utilizados os registros de monitoramento dos microbenchmarks nas iterações para a estimulação dos componentes do sistema em mais níveis de intensidade. Sendo assim, a quantidade de grupos gerada a partir de microbenchmarks e macrobenchmarks simultaneamente geram em média 15% para cada carga de trabalho, em relação à quantidade total de contadores do sistema. Para a avaliação qualitativa destes contadores, foi feito o levantamento de contadores considerados importantes para o monitoramento do sistema. Dos 28 contadores que são considerados importantes, apenas 3 deles não foram identificados no método proposto.

A metodologia proposta por Popiolek et al. (2021) é de essencial importância para este trabalho, dado que o principal objetivo é a implementação deste método, entretanto num sistema de forma dinâmica e que seja adaptável e extensível.

3. Desenvolvimento

Neste capítulo é apresentada a metodologia proposta para a criação da ferramenta de redução automática de contadores de desempenho. Esta ferramenta tem como objetivo indicar um conjunto reduzido de contadores que representam um sistema sem afetar a qualidade das métricas coletadas, de forma automática e transparente para o utilizador, fornecendo também estatísticas detalhadas no processamento dos dados. As seguintes características compõem a ferramenta proposta:

- Adaptável: diferentes módulos com funcionalidades específicas, implicando assim na facilidade de integração com outros serviços, ou a modificação de acordo com a necessidade do sistema.
- Extensível: capacidade de extensão de módulos já existentes ou adição de novos módulos com propriedades diferentes de acordo com os requisitos da aplicação.
- Configurável: fácil configuração, de acordo com o método de seleção dos contadores utilizado, assim como o tipo de algoritmo e as diversas propriedades que procedem dele.
- Adequável para integração: capacidade de integração com tecnologias de manipulação de contadores de desempenho já consolidadas na comunidade.

As vantagens na utilização desta ferramenta incluem: Menor utilização de armazenamento e processamento sobre os dados dos contadores, trazendo assim uma

melhoria na performance do sistema; menor demanda de trabalho na seleção de contadores representativos; ou até mesmo a anulação da necessidade de um especialista na seleção manual dos contadores no sistema.

Na Figura 1 é representado o diagrama da arquitetura proposta neste trabalho, com base no modelo desenvolvido em (Popiolek et al., 2021) e integrado de acordo com as propriedades de um serviço que pode ser reconfigurado e utilizado numa arquitetura de microsserviços.

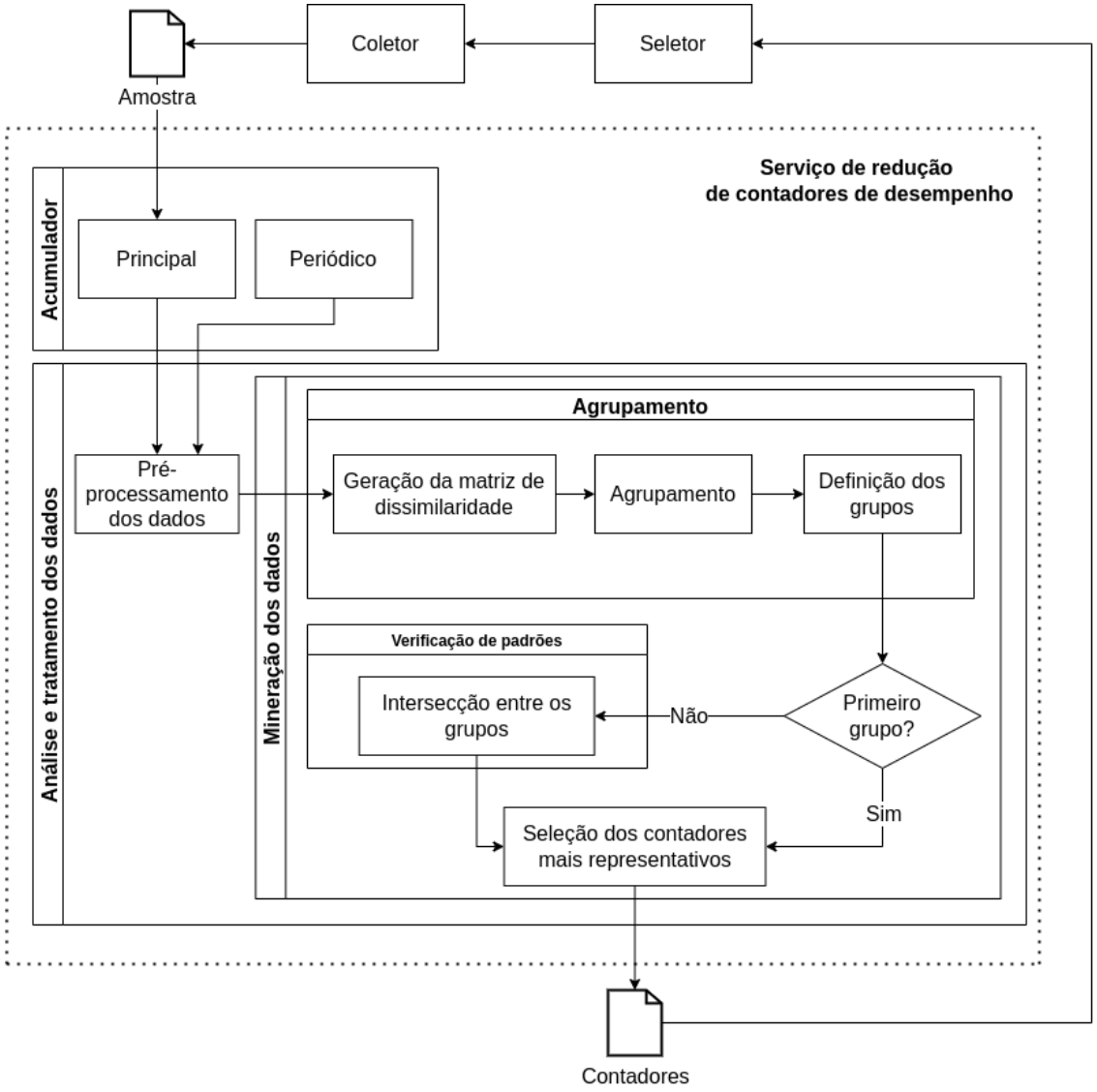


Figura 1. Diagrama da arquitetura proposta

Esta ferramenta foi desenvolvida inteiramente na linguagem Python, utilizando conceitos de orientação a objetos para a modularização mais eficiente, trazendo assim maior independência dos módulos produzidos. A ferramenta está disponível em (Pes, 2022).

A seguir são apresentadas as características detalhadas de cada elemento implementado sobre o Serviço de redução de contadores de desempenho, apresentado na Figura 1.

3.1. Acumulador

O Acumulador é o módulo responsável por receber amostras sequenciais do coletor, tanto unicamente quanto em lotes. Este módulo utiliza uma API Web Flask no qual possui dois endpoints, que são endereços do serviço responsáveis pelo atendimento de requisições enviadas pelo cliente, neste caso o coletor.

O primeiro endpoint é responsável pelo recebimento das amostras de contadores de desempenho, no qual primeiramente lê-se as amostras já armazenadas na aplicação, e concatena com as amostras enviadas. Vale ressaltar que caso seja enviado uma quantidade de amostras que extrapole a quantidade máxima de amostras na janela, a quantidade extrapolada é transferida para a próxima janela que será composta.

Este módulo possui também controle de concorrência sobre os conjuntos manipulados, como o conjunto de amostras; conjunto de grupos; e conjunto de amostras que estão aguardando a execução do processo de análise, garantindo assim a paralelização com outros processos que utilizam os recursos compartilhados em cada etapa. Também possui o controle sobre o acionamento do processo de análise dos dados ao atingir o número máximo de amostras em um único conjunto.

Caso a funcionalidade de periodicidade de todos os contadores seja ativada, o primeiro envio das amostras para este módulo deve conter todos os contadores que o usuário deseja utilizar, pois são guardados para futuras janelas periódicas. O processo de análise pode ou não ser acionado dependendo do momento no qual o sistema se encontra, sendo que se há periodicidade de todos os contadores, ao chegar no intervalo correspondente são sub-escritos os contadores representativos por todos os contadores contidos na configuração inicial. Portanto, na janela em que todos os contadores são disponibilizados, o coletor lê as métricas do conjunto completo de contadores, e só na próxima janela periódica que será feita a análise do conjunto de amostras da janela anterior.

O segundo endpoint é responsável pelo envio dos contadores mais significativos selecionados no momento em que foi solicitado, utilizado pelo coletor.

3.2. Análise e tratamento dos dados

O módulo de Análise e tratamento dos dados é responsável pela execução dos conjuntos acumulados de amostras executadas pelo acumulador, realizando o pré-processamento dos contadores; a geração da matriz de dissimilaridade; o agrupamento; e a definição dos grupos, assim armazenando os grupos gerados de cada conjunto em arquivos, para serem utilizados nas próximas etapas. Este processo utiliza como base o algoritmo de seleção de contadores de desempenho disponibilizado em (Popiolek et al., 2018).

3.3. Pré-processamento dos dados

Responsável pela limpeza das amostras enviadas pelo acumulador, no qual realiza a eliminação de contadores em que qualquer amostra seja nula. Amostras podem ser nulas por diversos fatores, como contadores que estão indisponíveis no sistema monitorado; ou em períodos diferentes do período em que foi monitorado, no qual a métrica não está presente no sistema. Também é responsável pela remoção de contadores que não possuem variância, ou que possuem mas que são mínimas, devido à convergência do algoritmo utilizado. Caso a propriedade devmode esteja habilitada, serão armazenadas estatísticas da etapa de remoção de contadores onde as amostras eram nulas, para avaliação do usuário.

3.4. Mineração dos dados

Esta subseção apresenta todos os processos para a seleção dos contadores representativos dos conjuntos de amostras enviadas.

3.4.1 Agrupamento

Método de geração dos grupos de contadores mais significativos para um determinado conjunto de amostras. Composto pela geração da matriz de dissimilaridade, o agrupamento a partir dos algoritmos previamente estabelecidos e a definição dos grupos. Também gera logs estatísticos para a avaliação de cada janela de amostras calculada caso o devmode esteja habilitado.

Este é um módulo genérico para a geração dos grupos, sendo possível a realização na mudança do método dependendo da aplicabilidade. Nesta versão, foi utilizado o modelo realizado em (Popiolek et al., 2018) para obtenção da matriz de distâncias e definição dos grupos. Para obtenção da matriz de distâncias, primeiro gera-se a matriz de dissimilaridade a partir da correlação de Pearson, e a seguir aplicado a função do coeficiente de distância (Popiolek et al., 2018). Em seguida, é realizado o agrupamento dos dados com o método Complete Linkage, utilizando a matriz de distâncias e o tamanho do corte no dendrograma como parâmetros.

Caso o sistema tenha apenas um único conjunto de grupos, no qual foi executado apenas uma janela de amostras, não é necessária a intersecção entre os grupos, sendo assim o próximo passo é a seleção dos contadores mais representativos de cada grupo. Caso contrário, o próximo passo é a verificação de padrões.

3.4.2 Verificação de padrões

Caso o sistema tenha executado outras janelas de amostras, a ferramenta realiza a intersecção entre os conjuntos de grupos criados anteriormente (Popiolek et al., 2018), retornando assim um único conjunto de grupos.

3.4.3 Seleção dos contadores mais representativos

Método para seleção do contador mais representativo de cada grupo, no qual utiliza a matriz de distâncias para seleção do contador que representa os demais contadores do mesmo grupo (Popiolek et al., 2018).

4. Avaliação experimental

Este capítulo demonstra os benefícios e os meios na utilização da ferramenta proposta, mostrando testes produzidos com diferentes configurações com o objetivo de apresentar as diversas possibilidades na utilização da mesma. A avaliação da ferramenta exige a implementação e utilização de mecanismos para a análise dos contadores selecionados em diferentes cargas de trabalho, sendo que estas cargas podem ser inseridas tanto de forma sequencial em tempo real, quanto por cargas sintéticas que representam um ambiente já simulado.

Os experimentos nesta seção baseiam-se na utilização de workloads previamente coletados no trabalho de (Popiolek et al., 2018), sendo realizados com diferentes cargas de trabalho, simulando um ambiente mais controlado, e assim obtendo valores mais precisos sobre os experimentos para a garantia de uma melhor exatidão nas avaliações. O ambiente utilizado como base foi o MySQL Server, e os workloads foram gerados a partir do benchmark Yahoo! Cloud Serving Benchmark (YCSB). Os workloads utilizados foram:

- A: Atualização intensiva, sendo 50% operações de leitura e 50% operações de escrita.
- B: Leitura intensiva, com 95% operações de leitura e 5% operações de escrita.
- C: Apenas leitura, com 100% operações de leitura.

A quantidade de contadores iniciais em cada workload se resulta em 9941. Estes contadores são compostos pelos convencionais disponibilizados por hardware; e os contadores próprios da aplicação monitorada, como quantidade de acessos à métodos, ou estatísticas sobre o banco de dados. Estes workloads foram particionados em tamanhos distintos e enviados gradualmente para o acumulador da aplicação em intervalos que precedem a seleção dos contadores em cada rodada. Cada tamanho representa a quantidade máxima de amostras que um acumulador armazena, sendo que ao ser preenchida completamente, o processo de seleção dos contadores mais representativos é acionado. Assim, as próximas amostragens produzidas pelo coletor simulado serão com os contadores mais representativos selecionados na rodada anterior.

Foram executados testes com diferentes configurações sobre características específicas da ferramenta, com o intuito de avaliar a eficácia comparado com o algoritmo desenvolvido em (Popiolek et al., 2018), que opera sobre dados estáticos e offline, e assim apresentando apenas um conjunto de monitoramento para cada tipo de carga de trabalho.

4.1. Comparação dos contadores representativos sobre o conjunto não particionado

Como cada conjunto de cargas de trabalho utilizados para o processo de avaliação possuem diferentes tamanhos, para cada conjunto foi realizado a redução no tamanho de amostras em adequação ao tamanho da janela utilizada no acumulador. Também foi utilizado o critério de seleção das amostras sobre o meio do conjunto, descartando as amostras ao extremo do mesmo e assim removendo as que não representavam a carga de trabalho simulada (Popiolek et al., 2018). Sendo assim, cada workload foi reduzido até 300 amostras, no qual foram realizados testes com partições múltiplas da quantidade de amostras, sendo elas representadas na Tabela 1.

Tabela 1. Tamanho das janelas e quantidade de partições do conjunto

Tamanho da janela	300	150	100	50	25
Quantidade de execuções	1	2	3	6	12

Assim, primeiramente foram executados os testes sobre o conjunto original, de 300 amostras, representado na Tabela 2, apresentando assim o total de contadores de desempenho representativos e selecionados ao final do experimento. A ideia destes experimentos é avaliar as diferenças entre os contadores selecionados em diferentes tamanhos de janelas comparados com os contadores da janela de tamanho de 300, que é considerada a que seleciona contadores representativos mais realísticos, no qual realiza uma única vez a análise de dados.

Tabela 2. Contadores finais sobre os conjuntos originais

Workload A	451
Workload B	374
Workload C	324

4.1.1. Experimento 1: testes com renovação dos contadores mais significativos

Para a execução dos testes com renovação dos contadores mais significativos de cada rodada, foram utilizadas as mesmas janelas dos testes anteriores. A Tabela 3 apresenta os testes realizados sobre os conjuntos de teste, com a renovação dos contadores mais significativos para cada rodada. Ou seja, na primeira janela são enviados todos os contadores, já nas próximas são enviados os contadores representativos em relação à rodada anterior.

Tabela 3. Contadores finais após todas as rodadas com renovação

Tamanho da janela	150	100	50	25
Workload A	505	541	598	592
Workload B	454	503	586	631
Workload C	362	400	436	416

Tabela 4. Intersecção entre os conjuntos com renovação e o conjunto completo

Tamanho da janela	150	100	50	25
Workload A	297	278	233	186
Workload B	240	213	206	162
Workload C	209	194	165	116

Para avaliar a eficácia da solução, devemos verificar a intersecção dos contadores representativos de cada experimento comparado aos conjuntos originais. A Tabela 4 apresenta as intersecções dos contadores com renovação em relação aos conjuntos originais de cada workload. Observou-se que a quantidade de contadores diminuiu comparados ao teste anterior.

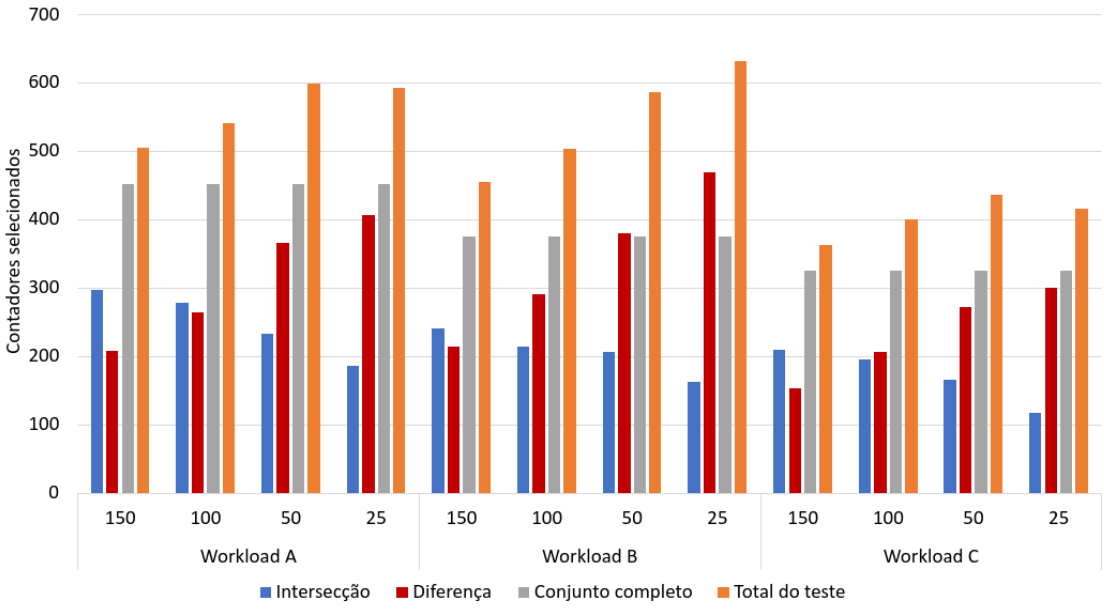


Figura 2. Semelhanças entre os conjuntos de contadores com renovação e sem periodicidade de todos os contadores

A Figura 2 apresenta a composição dos valores dos experimentos com renovação, para a melhor visualização da disparidade de contadores significativos que foram selecionados, comparados ao conjunto original. É possível observar a partir da intersecção, representada em azul, que poucos contadores são semelhantes

ao conjunto completo em diferentes tamanhos de janela, resultando numa pior qualidade pois, conforme se reduz o tamanho da janela, os contadores que eram considerados representativos e que não estavam na janela de 300 são levados mais em consideração para a seleção de futuras rodadas de execução dos métodos de similaridade e agrupamento. Com isso, o aumento da quantidade de contadores diferentes também aumentaram conforme o crescimento do tamanho da janela, resultando assim em extremos no qual pequenas janelas apresentam o dobro de contadores diferentes do conjunto completo, em vermelho, comparados com contadores que estavam no conjunto completo, em azul.

4.1.2. Experimento 2: testes com renovação dos contadores mais significativos e com periodicidade de todos os contadores

O próximo experimento tem como objetivo avaliar a execução da aplicação utilizando a configuração de reutilização de todos os contadores de desempenho, com o intuito de minimizar o erro dos contadores comparados ao conjunto original, sem o custo associado de cálculo de todos os contadores disponíveis em cada iteração. Como os valores periódicos de alimentação com todos os contadores podem ser múltiplos do tamanho das janelas de cada workload, é possível que no final do experimento todos os contadores do sistema sejam significativos. Para contornar esta informação, foi habilitado o mecanismo de periodicidade de execução com mínimo de amostras, utilizando um período suficientemente grande para que sejam executados todos os experimentos e logo em seguida seja executado o processamento via periodicidade.

A Figura 3 apresenta a composição dos experimentos com renovação e com frequência de recebimento de todos os contadores disponíveis no sistema. Podemos observar que houve uma melhor quantidade de contadores de desempenho selecionados corretamente, cerca de 10%, comparados com os contadores no experimento com apenas renovação. Também, em periodicidades maiores, a quantidade de contadores corretos, representados em azul, perdem pouca precisão comparados com períodos menores, continuando ainda com menores contadores diferentes do conjunto completo, em vermelho.

Assim, existem diferentes abordagens que podem ser adotadas para a garantia de uma melhor precisão na seleção dos contadores mais representativos, no qual vai depender do sistema em que está sendo utilizado. Janelas de tamanho grande apresentam uma melhor eficiência na seleção pois executam menos vezes a análise dos dados, entretanto podem não representar o sistema quando há mudanças repentinas de comportamento. Janelas de tamanho pequeno apresentam uma menor eficiência na seleção, contudo oferecem contadores representativos mais rapidamente, sendo assim viáveis em sistemas mais dinâmicos. E por fim, periodicidades de todos os contadores melhoram a qualidade dos contadores representativos, entretanto não são tão eficientes em periodicidades pequenas, pois aumentam muito a quantidade de contadores que não eram identificados como representativos, e também podem aumentar o overhead quando utilizados em pequenas periodicidades ou grandes janelas.

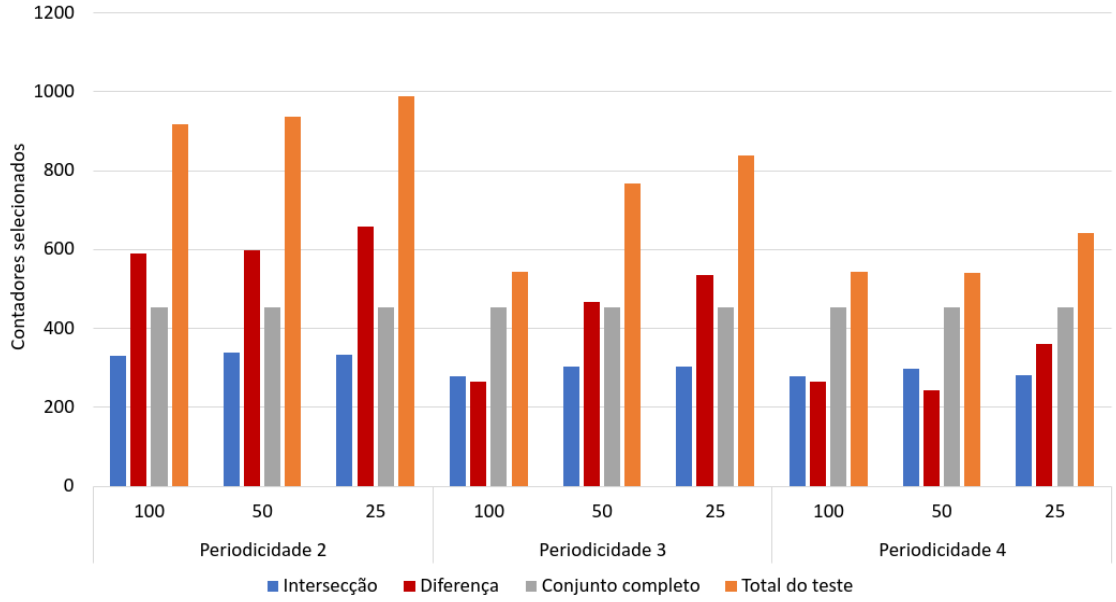


Figura 3. Semelhanças entre os conjuntos de contadores com renovação e periodicidade de todos os contadores

5. Considerações finais

Contadores de desempenho estão cada vez sendo mais utilizados, principalmente e majoritariamente em sistemas distribuídos, no qual prioriza-se muito tanto a robustez quanto performance dos sistemas hospedados. Assim, a utilização de contadores de desempenho no sistema é imprescindível para o monitoramento de falhas; a visualização de inconsistências; assim como a visualização e monitoramento na utilização dos componentes no sistema. Este trabalho descreve os fundamentos, a metodologia e a implementação de um serviço que seja adaptável, extensível, configurável e adequável para a integração com outras ferramentas de coletas de métricas do sistema, priorizando o desempenho no sistema ao todo. Como visto, foi necessário o conhecimento de diversas técnicas de modularização e configuração, para a garantia de robustez, legibilidade e adaptabilidade do sistema para diferentes modos, caso sejam adicionados em trabalhos futuros. Também foram utilizados conhecimentos em Ciência de Dados para avaliação dos padrões resultantes de cada experimento, para análise dos parâmetros aplicados nas configurações da ferramenta.

Nos experimentos, foi possível identificar que a quantidade de contadores de desempenho que a ferramenta selecionou ficou extremamente inferior à quantidade de contadores de desempenho no sistema ao todo. Os experimentos também indicaram as características de cada configuração, mostrando que é possível realizar a avaliação sobre diferentes configurações para uma melhor aplicabilidade no sistema. Assim, foi possível notar que pequenas janelas de amostras tornam os dados enviesados, crescendo assim a quantidade de contadores que não representam o sistema ao todo, mas grandes janelas não são tão aplicáveis em diferentes sistemas. Também, foi notado que janelas com renovação, que são as que representam a aplicação, possuem o custo na diminuição da previsibilidade de contadores que representavam o sistema, assim como a maior quantidade de contadores que não representam o sistema. Por fim, utilizando

periodicidades de todos os contadores, a quantidade de contadores que realmente são representativos aumenta, mas com o custo de processamento de todos os contadores em períodos estipulados.

6. Trabalhos futuros

Este trabalho foi realizado utilizando mecanismos que facilitam a incorporação de ambientes baseados em microsserviços, pois coletores de métricas são utilizados em conjunto com sistemas de grande escala. Entretanto, não foram utilizadas ferramentas como Kubernetes e Docker Swarm para orquestração do serviço implementado junto com serviços de coletores de métricas, em um sistema realístico. Um próximo passo seria a implementação numa arquitetura de microsserviços, utilizando benchmarks para a visualização da eficácia e escalabilidade da ferramenta nestes ambientes. Também, nestes mesmos ambientes, é interessante a análise de comportamento do sistema em tempo real, dado que os experimentos realizados neste trabalho foram gerados a partir de amostras sintéticas já processadas.

Como a ideia deste trabalho foi o desenvolvimento de uma aplicação adaptável para diferentes técnicas de mineração de dados, seria interessante realizar experimentos com outras estratégias de agrupamento para avaliação dos contadores obtidos, assim como a utilização de algoritmos adaptativos de valor de corte do dendrograma gerado pelo agrupamento.

As análises deste trabalho identificam a quantidade de contadores que foram selecionados, mas não faz a verificação exata da qualidade dos grupos de contadores e os contadores de desempenho representativos gerados. Um próximo passo seria o estabelecimento de mecanismos de avaliação das configurações da ferramenta utilizando as métricas de validação de agrupamento. Isso possibilitaria a análise dos algoritmos utilizados, assim como as propriedades mais básicas de configuração, como o tamanho das janelas e a periodicidade de todos os contadores.

7. Referências

PES, Gabriel. Serviço de monitoramento de contadores de desempenho com baixa sobrecarga. [S.I: s.n.], ago. 2022. Disponível em: https://github.com/biepes/reducer-tool

Popiolek, P. F., & Mendizabal, O. M. (2013). Monitoring and analysis of performance impact in virtualized environments. Journal of Applied Computing Research, 2(2), 75–82. https://doi.org/10.4013/jacr.2012.22.03

Popiolek, P. F. (2018). Redução de sobrecarga de monitoramento em ambientes virtualizados através de contadores de desempenho. 1–89.

Popiolek, P. F., Machado, K. dos S., & Mendizabal, O. M. (2021). Low overhead performance monitoring for shared infrastructures. Expert Systems with Applications, 171(December 2020), 114558. https://doi.org/10.1016/j.eswa.2020.114558

Xia, Y., Liu, Y., Chen, H., & Zang, B. (2012). CFIMon: Detecting violation of control flow integrity using performance counters. Proceedings of the International

Conference on Dependable Systems and Networks. https://doi.org/10.1109/DSN.2012.6263958

LEWIS, James; FOWLER, Martin. Microservices: A definition of this new architectural term, 2014. Disponível em: https://martinfowler.com/articles/microservices.html.

SHANG, Weiyi et al. Automated detection of performance regressions using regression models on clustered performance counters. In: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering. 2015. p. 15-26.

Skiena, S. S. (2017). The data science design manual. In Springer. https://doi.org/10.1007/978-3-319-55444-0

Yao, K., B. de Pádua, G., Shang, W., Sporea, S., Toma, A., & Sajedi, S. (2018). Log4Perf. 127–138. https://doi.org/10.1145/3184407.3184416