João Pedro Sandrini Milanezi

Revisão Bibliográfica - Isolamento em Sistemas de Computação em Nuvem

Vitória, ES

Outubro/2024

João Pedro Sandrini Milanezi

Revisão Bibliográfica - Isolamento em Sistemas de Computação em Nuvem

Relatório referente à Tarefa A do plano de trabalho vinculado à bolsa IC-WP5-ISO - FAPESP

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Departamento de Engenharia Elétrica

Orientador: Tereza Cristina Melo de Brito Carvalho e Moisés Renato Nunes Ribeiro

Vitória, ES

Outubro/2024

Resumo

Sistemas de computação em nuvem são cada vez mais comuns atualmente, presentes nas mais diversas áreas de aplicação. Ademais, com o avanço das tecnologias de processadores, redes e armazenamento, esses sistemas passam a lidar com uma escala cada vez maior de complexidade e quantidade de dados. Diante disso, a segurança desses sistemas e dos dados utilizados é de grande relevância. Assim, este trabalho faz uma revisão de conceitos básicos de segurança utilizados em sistemas de nuvem e apresenta casos de implementações reais utilizando esses conceitos, principalmente no escopo de transporte inteligente, ramo que cresce atualmente como resultado do advento das chamadas Cidades Inteligentes.

Palavras-chave: Segurança, Virtualização, Tempo-Real.

Agradecimentos

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brasil. Processo nº 2024/23727-8.

Sumário

1 Introdução

Na computação, a segurança de aplicações críticas é fundamental, garantindo o funcionamento correto do sistema e a segurança dos dados. Com o crescimento da computação em nuvem, essa necessidade se tornou ainda maior, dada a quantidade massiva de dados que esses sistemas coletam, processam e armazenam. Assim, a virtualização é a base do funcionamento desse tipo de sistema, sendo responsável pelo compartilhamento dos recursos de *hardware* disponíveis (BAZM et al., 2019). Além disso, confere portabilidade, escalabilidade, resiliência, isolamento e segurança.

O isolamento desses sistemas é fundamental para garantir que não seja suscetível à ataques e vazamentos e dados, entretanto requisitos de sistema podem tornar essa tarefa mais complicada, como é o caso de aplicações de tempo real, por exemplo, sistemas de transporte inteligente. Nessas aplicações, o isolamento e segurança das instâncias deve ser garantido com o mínimo impacto em performance.

Com isso, surgem técnicas de virtualização e isolamento mais sofisticadas que conferem maior segurança e atendem aos requisitos de tempo real, com uma baixa relação de *trade-off* entre performance e segurança. Para a compreensão dessas técnicas é importante entender também os conceitos básicos que levam até elas, e também como são implementadas em cenários reais, daí vem o objetivo deste projeto.

2 Isolamento

O conceito de isolamento pode ser dividido em três: de Segurança, de Falhas e de Performance, que serão abordados a seguir. O estudo de Sklavos, Jansen e Trivedi (2023) aborda brevemente os três conceitos de isolamento e se aprofunda nas técnicas geralmente utilizadas para atingir os tres tipos de isolamento.

2.1 Isolamento de Segurança (*Security Isolation*)

O Isolamento de Segurança é o que garante que mesmo com a segurança de uma parte do sistema comprometida, o usuário mal-intencionado não consiga acessar outras partes do sistema (SKLAVOS; JANSEN; TRIVEDI, 2023). Além de garantir a segurança em geral, protegendo dados e processos, o isolamento de segurança permite executar programas não confiáveis sem comprometer o resto do sistema e executar aplicações confiáveis porém mais suscetíveis a ataques, com brechas conhecidas e que não foram corrigidas, por exemplo, sem afetar a segurança geral do sistema.

2.2 Isolamento de Falhas (*Fault Isolation*)

O Isolamento de falhas garante que um problema em uma parte do sistema não afetará as outras, evitando assim que falhas menores comprometam o sistema por completo (SKLAVOS; JANSEN; TRIVEDI, 2023). Por exemplo, impede que um serviço cause um *crash* do sistema inteiro por um erro de escrita na memória. Esse tipo de isolamento garante confiabilidade e resiliência ao sistema.

2.3 Isolamento de Performance (*Performance Isolation*)

Por fim, o isolamento de performance certifica que os recursos físicos sejam distribuídos de forma justa entre todos os processos, de forma que a performance de um não cause impacto nos demais (SKLAVOS; JANSEN; TRIVEDI, 2023). Isso é importante já que recursos de hardware são limitados, pode ocorrer de uma aplicação demandar muita memória ou uso de processador, impedindo que as demais aplicações do sistema tenham acesso à esses recursos, ou que tenham acesso limitado, favorecendo a performance de uma aplicação e comprometendo a do restante do sistema.

3 Virtualização

Como descrito anteriormente, a camada de virtualização em sistemas de computação em nuvem permite a divisão dos recursos físicos disponíveis. Para isso, podem ser utilizados alguns tipos de arquiteturas, que serão discutidos em seguida. Além disso, a técnica de execução de aplicações diretamente no *hardware* local, sem a camada de virtualização é denominada *bare-metal*, porém esse tipo de implementação dificulta a distribuição de recursos, confere menor segurança e gera maiores problemas em relação a compatibilidade, em troca de mais performance (GIALLORENZO et al., 2021). A Figura 1 ilustra as três técnicas mais comuns de virtualização.

Técnicas mais aprimoradas como a de Contêineres Seguros e *Unikernels* buscam aprimorar as VMs e Contêineres, abordando os principais problemas de cada método e reduzindo seus lados negativos. Todas esses métodos são discutidos e avaliados mais profundamente por Sklavos, Jansen e Trivedi (2023), bem como os principais serviços utilizados para sua implementação.

3.1 Máquinas Virtuais

As máquinas virtuais (VM) foram a primeira forma de virtualização, tecnologia criada pela IBM em 1972 (IBM, 2024). Cada VM possui um sistema operacional próprio e permite a execução de múltiplas aplicações por VM.

Esse tipo de implementação possui um *overhead* maior, comprometendo a perfor-

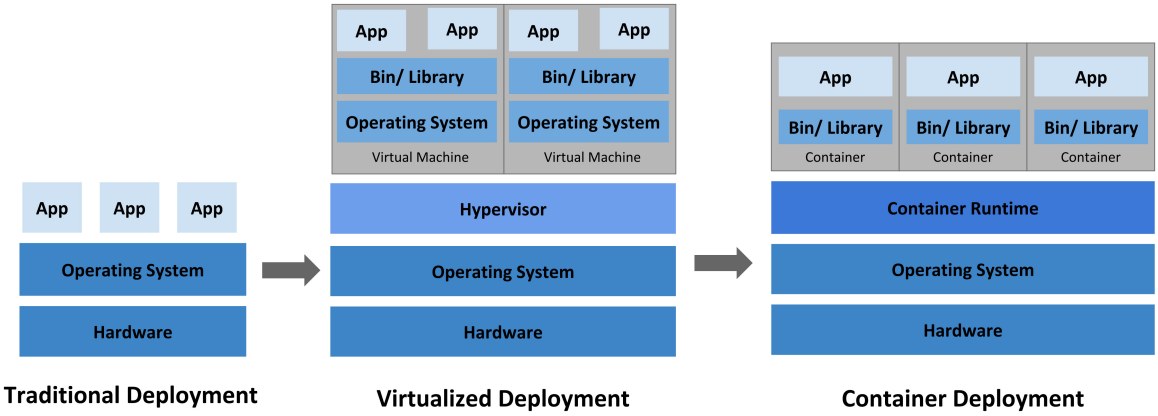

Figura 1 – Diagramas de implementação *Bare-Metal*, VM e *Container* (KUBERNETES, 2024).

mance, porém garante um ótimo nível de isolamento e segurança (SKLAVOS; JANSEN; TRIVEDI, 2023). Isso porque a camada do sistema virtualizada é o hardware, de modo que cada VM tenha acesso apenas ao seu hardware virtualizado, cada uma com um *kernel* próprio, gerenciadas por um *hypervisor*.

3.2 Contêineres

Contêineres são uma implementação com um *overhead* muito menor em relação à VMs. Esse tipo de virtualização é implementado dividindo o sistema operacional (SO) hospedeiro entre as aplicações, carregadas em Contêineres que possuem apenas os recursos necessários para o seu funcionamento, o que facilita a portabilidade e melhora a performance (BHARDWAJ; KRISHNA, 2021). Entretanto, com o compartilhamento do SO , o isolamento e segurança de Contêineres é menor em relação à VMs (SKLAVOS; JANSEN; TRIVEDI, 2023).

3.3 Contêineres Seguros

Esse tipo de virtualização tem o objetivo de unir as vantagens das VMs aos Contêineres. De forma sucinta, é a execução de Contêineres dentro de VMs leves e exclusivas para cada aplicação, reduzindo assim *overhead* e ganhando as vantagens de isolamento e segurança das VMs, com performance e facilidade de implementação comparável à de Contêineres (SKLAVOS; JANSEN; TRIVEDI, 2023). A OpenStack Foundation mantém uma implementação desse tipo de virtualização, chamada Kata Containers. Sua arquitetura está exemplificada na Figura 2.

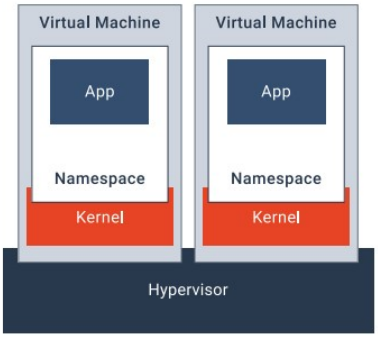

Figura 2 – Arquitetura Contêiner Seguro - Kata Container (KATACONTAINERS, 2023).

3.4 *Unikernels*

Unikernels consistem na junção da aplicação e todos os componentes necessários para seu funcionamento em uma imagem leve e que pode ser administrada por um *hypervisor*, assim como VMs. De forma simples, se trata de uma VM hiper-especializada, que ganha em performance e segurança. Entretanto, o desenvolvimento e implementação desse tipo de aplicação é mais difícil em relação às outras técnicas de virtualização, principalmente por problemas de portabilidade e limitação das ferramentas de máquinas virtuais (SKLAVOS; JANSEN; TRIVEDI, 2023).

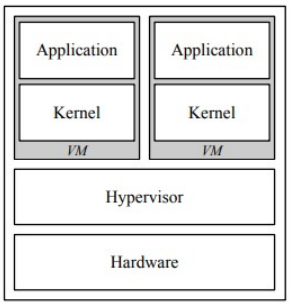

Figura 3 – Arquitetura Unikernel (SKLAVOS; JANSEN; TRIVEDI, 2023).

4 Aplicações

O foco deste trabalho é a aplicação de técnicas de isolamento e segurança em Serviços de Transporte Inteligente (STI). Esse tipo de aplicação é sensível ao tempo e requer baixas latências e respostas em tempo real. À seguir foram compiladas algumas aplicações, descrevendo sua arquitetura e as técnicas de isolamento implementadas, de acordo com os autores nos trabalhos mencionados.

4.1 COSMOS

Dubey et al. (2019) apresentam o sistema operacional COSMOS como solução e discorrem sobre as técnicas implementadas para garantir isolamento de performance,

segurança e falhas. Dessa forma, por mais que a arquitetura da implementação seja diferente, as técnicas utilizadas podem ser tomadas como princípios na implementação em STI.

O princípio utilizado na aplicação é o particionamento dos serviços do SO. Dentro de uma mesma partição, os processos dividem tempo de processamento, organizados de acordo com uma prioridade, que pode ser modificada e a execução interrompida por qualquer outro programa, desde que dentro da partição. Os programas de uma mesma partição podem se comunicar através de Memória compartilhada (*Blackboards*), *buffers*, Eventos e Semáforos. O isolamento de performance é garantido com um agendador (*scheduler*) que organiza as janelas de tempo de processamento de cada partição dentro de janelas maiores que se repetem em um intervalo de tempo que pode ser definido, chamado *hyperperiod*.

Cada partição é identificada e só pode ser executada em modo usuário, sem maiores permissões. As partições também possuem um sistema de arquivos próprio, isolado por um *chroot* e com outras características próprias: alocação física de armazenamento, direitos de acesso, descrição de arquivos, identificação de dispositivo de armazenamento e apelidamento de volumes de disco.

A arquitetura de segurança se baseia nos conceitos de isolamento de recursos de processamento e armazenamento abordados acima, juntamente com controle de acesso obrigatório e separação bem definida de recursos de comunicação, controlando o fluxo de informações com múltiplas camadas de segurança e controlando o uso de serviços existentes pelas aplicações.

O fluxo de informações é controlado por meio de um sistema desenvolvido pelos autores, chamado de *Secure Transport*. De forma simples, os terminais devem possuir um *flow*, indicando o sentido de comunicação e a origem/destino, assim a comunicação só pode ser estabelecida se os dois terminais possuírem essa associação. As mensagens enviadas devem conter um identificador e um terminal, o identificador é conferido com os do terminal especificado e no caso de qualquer inconsistência, a mensagem não é entregue. Isso pode afetar comunicação legítima caso ocorra algum erro, entretanto um agente mal-intencionado nunca conseguirá estabelecer comunicação.

Por fim, o isolamento de falhas é obtido com sistemas de gerenciamento de saúde e falhas. Os erros são tratados em três níveis do sistema: Módulo, Partição e Processo. Caso o sistema de gerenciamento de saúde detecte erros em um módulo ou partição, o erro é tratado pelo processo de tratamento de erros no *kernel* da partição, podendo ser tomadas as seguintes medidas: ignorar o erro; parar o módulo/partição; reiniciar o módulo/partição. Caso o erro seja no nível de processo, é tratado no mesmo nível, com funções predefinidas.

Utilizando essas técnicas e o SO criado, Dubey et al. (2019) apresentam a implementação de isolamento robusto em sistemas de tempo real. O exemplo testado pelos

autores simula uma rede de três satélites orbitando em volta da Terra, com sistemas para receber comandos, atualizar a posição do próprio satélite e obter a dos outros, além quatro unidades de processamento de imagem. Esse tipo de sistema, assim como STI, são sensíveis ao tempo e dependem de baixas latências para evitar colisões, por exemplo. Assim, é possível traçar um paralelo entre a implementação dos autores e STI.

4.2 *Implementation of the Smart Traffic Management System through Cloud Computing*

Priyan et al. (2023) no trabalho intitulado *Implementation of the Smart Traffic Management System through Cloud Computing* descreve a implementação de um sistema de trânsito inteligente por meio da nuvem, a fim de manter a boa conduta dos agentes envolvidos no trânsito, principalmente motoristas, e evitar acidentes. O sistema é baseado em outro já existente, proposto por Dhingra et al. (2021)

Para isso, a implementação na nuvem privada foi proposta como maneira de melhorar a segurança do sistema, tendo em vista a natureza não-segura da nuvem pública. Com o objetivo também de aumentar a segurança, um banco de dados *Oracle* foi utilizado para armazenar e transferir dados. Organizados em *datasets* a fim de delimitar as funções de cada sensor dentro do sistema, os dados são armazenados na nuvem. De acordo com o autor, essas medidas garantem alta segurança.

Como o objetivo do sistema é parar veículos fora de controle e motoristas imprudentes, os autores propõem uma espécie de armadilha posicionada no caminho do veículo, caso seja detectada imprudência ou acidentes pelo sistema de câmeras de monitoramento. De modo que um sinal é enviado para a armadilha, que é acionada no caminho do veículo, a tempo de impedir acidentes, o que remete ao requisito de tempo do sistema.

Um *Gateway* é utilizado para segurança, hospedando um *Firewall*. Todos os dados são protegidos por esse *Firewall*, reduzindo latência e melhorando a segurança. Os veículos possuem suas informações de localização, velocidade e direção armazenados, sendo utilizadas para detecção de possíveis colisões. O monitoramento de vídeo grava todas as ações dos veículos. Outros dispositivos periféricos são sensores de presença, velocidade e luz, que se comunicam por um protocolo baseado em internet e computação em neblina (*fog computing*). Além da ativação das armadilhas, o sistema comunica o administrador e outros usuários na mesma via da presença de um veículo imprudente ou fora de controle.

Em relação ao sistema original, o trabalho de Priyan et al. (2023) demonstra uma melhora de segurança e performance, graças às técnicas implementadas.

4.3 *Smart Mobility in the Cloud: Enabling Real-Time Situational Awareness and Cyber-Physical Control Through a Digital Twin for Traffic*

Neste trabalho, Xu et al. (2023) propõem uma solução de gerenciamento de trânsito utilizando gêmeos digitais (DT), apoiada por tecnologias de Internet das Coisas (IoT), Inteligência Artificial (IA) e Computação em Nuvem chamada *CTwin*. Isso porquê o avanço dessas tecnologias habilita a implementação de modelos de DT, uma vez que permitem coletar e processar uma enorme quantidade de dados em tempo real, resultando em modelos digitais mais próximos do real e com maior capacidade de prever e controlar situações no ambiente em que foi implementado.

Na implementação, os dados necessários para o DT são coletados por diversos sensores, sendo: radares capazes de detectar a quantidade de veículos, velocidades médias e a ocupação por faixa em certos intervalos; câmeras que capturam e processam o movimento dos veículos em tempo real. Como essa aplicação tem o objetivo de reduzir congestionamento e gasto de energia na região, são coletados também dados dos sinais de trânsito e de fontes públicas como Waze.

A implementação da plataforma foi feita utilizando uma série de softwares: *frameworks* para desenvolvimento de aplicações web Angular e Java Spring, Docker como a plataforma de contêineres, Kubernetes para orquestração de contêineres, PostgreeSQL e Post-GIS como bancos de dados, pois permitem lidar com coordenadas geográficas e GeoServer, um servidor baseado em Java que também permite trabalhar com coordenadas. A figura 4 ilustra a arquitetura do sistema.

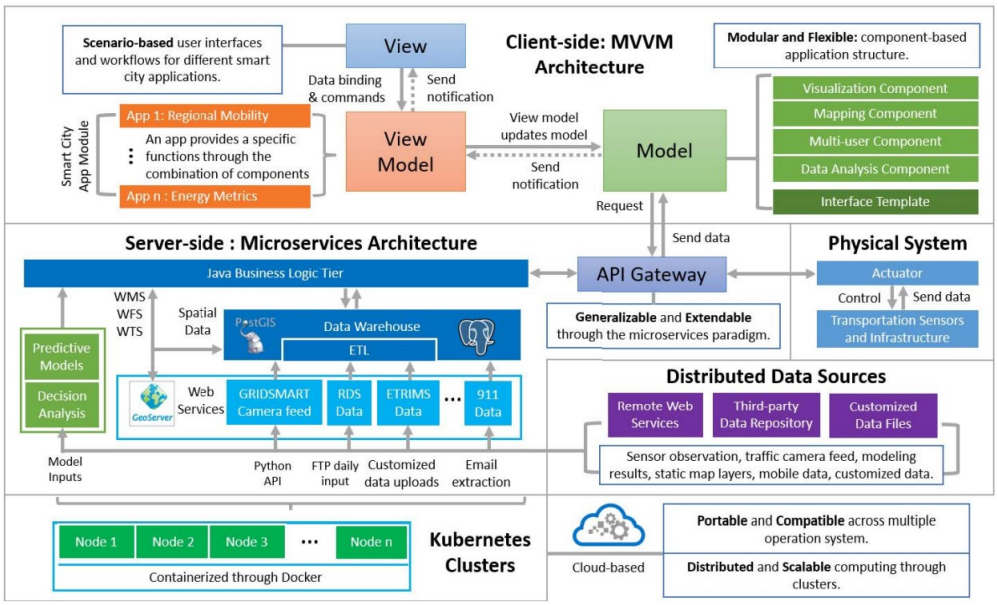

Figura 4 – Arquitetura CTwin (XU et al., 2023).

A implementação em contêineres permite uma arquitetura de microsserviços que garante compatibilidade com os dispositivos IoT, escalabilidade e portabilidade da aplicação

para outros cenários reais. Além disso, cada serviço possui uma estrutura de dados bem definida, o que também facilita o processo. A aplicação também adota o paradigma de borda "Nuvem das Coisas"(CoT) para a integração dos diversos sensores.

A plataforma CTwin foi implementada em duas situações em estradas importantes de Chattanooga, Tennessee, nos Estados Unidos. O primeiro caso foi implementado com o objetivo de demonstrar a capacidade do sistema de emular em quase tempo real (*near-real-time*) o trânsito na região. A segundo caso foi implementado para monitorar e localizar acidentes nas estradas de interesse.

5 Conclusão

O conteúdo exposto neste trabalho apresenta uma base importante de conhecimento para o entendimento de técnicas de isolamento e segurança para sistemas de computação em nuvem, baseadas principalmente na virtualização e arquitetura de microsserviços.

Além disso, foram expostos trabalhos com implementações teste em áreas de interesse, principalmente Sistemas de transporte inteligente, discorrendo sobre as técnicas de isolamento e segurança empregadas.

Em trabalhos futuros, as técnicas discutidas aqui podem ser implementadas e/ou aprimoradas em *test-beds* e aplicações reais de sistemas de computação em nuvem com propósitos diversos. Entretanto, este trabalho tem o objetivo de fundamentar a implementação de um sistema de monitoramento de cruzamentos em fases futuras do projeto, aplicando os conceitos de isolamento e segurança descritos.

Referências

BAZM, M.-M.; LACOSTE, M.; SÜDHOLT, M.; MENAUD, J.-M. *Isolation in cloud computing infrastructures: new security challenges*. [S.l.]: Springer, 2019. 197–209 p. Citado na página 5.

BHARDWAJ, A.; KRISHNA, C. R. Virtualization in cloud computing: Moving from hypervisor to containerization—a survey. *Arabian Journal for Science and Engineering*, Springer, v. 46, n. 9, p. 8585–8601, 2021. Citado na página 7.

DHINGRA, S.; MADDA, R. B.; PATAN, R.; JIAO, P.; BARRI, K.; ALAVI, A. H. Internet of things-based fog and cloud computing technology for smart traffic monitoring. *Internet of Things*, v. 14, p. 100175, 2021. ISSN 2542-6605. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2542660519302100>. Citado na página 10.

DUBEY, A.; EMFINGER, W.; GOKHALE, A.; KUMAR, P.; MCDERMET, D.; BAPTY, T.; KARSAI, G. Enabling strong isolation for distributed real-time applications in edge computing scenarios. *IEEE Aerospace and Electronic Systems Magazine*, v. 34, n. 7, p. 32–45, 2019. Citado 2 vezes nas páginas 8 e 9.

GIALLORENZO, S.; MAURO, J.; POULSEN, M. G.; SIROKY, F. Virtualization costs: benchmarking containers and virtual machines against bare-metal. *SN Computer Science*, Springer, v. 2, n. 5, p. 404, 2021. Citado na página 6.

IBM. *What is a Virtual Machine?* 2024. Acesso em: 18/09/2024. Disponível em: <https://www.ibm.com/topics/virtual-machines>. Citado na página 6.

KATACONTAINERS. *Kata Containers: Lightweight Virtualized Containers for Cloud-Native Workloads*. 2023. <https://katacontainers.io/collateral/kata-containers-1pager.pdf>. Acesso em: 10/09/2024. Citado na página 8.

KUBERNETES. *Overview of Kubernetes Concepts*. 2024. Accesso em: 18/09/2024. Disponível em: <https://kubernetes.io/docs/concepts/overview/>. Citado na página 7.

PRIYAN, S.; OLUWADARE, J.; OYEBODE, O.; RAO, A.; RAO, A.; MANIMEKALAI, K. Implementation of the smart traffic management system through cloud computing section a-research paper issn 2063-5346 5644 eur. v. 12, p. 5644–5652, 06 2023. Citado na página 10.

SKLAVOS, A.; JANSEN, M.; TRIVEDI, A. *Exploring the Performance-Isolation Trade-off for Isolation Mechanisms*. [S.l.], 2023. Citado 4 vezes nas páginas 5, 6, 7 e 8.

XU, H.; BERRES, A.; YOGINATH, S. B.; SORENSEN, H.; NUGENT, P. J.; SEVERINO, J.; TENNILLE, S. A.; MOORE, A.; JONES, W.; SANYAL, J. Smart mobility in the cloud: Enabling real-time situational awareness and cyber-physical control through a digital twin for traffic. *IEEE Transactions on Intelligent Transportation Systems*, v. 24, n. 3, p. 3145–3156, 2023. Citado na página 11.