Proposta de uma Arquitetura de Middleware para Elasticidade Vertical em Nuvem

Gabriel Reis Panho, Dalvan Griebler[1]

[1]Escola Politécnica, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

gabriel.panho@edu.pucrs.br,dalvan.griebler@pucrs.br

Resumo. *Este trabalho propõe o elastic_cloud, um middleware voltado à implementação de elasticidade vertical autônoma em ambientes de nuvem privada. A solução utiliza um mecanismo de adaptação baseado no ciclo Monitor–Analyze–Plan–Execute (MAPE), responsável por monitorar métricas do sistema e ajustar dinamicamente os recursos das máquinas virtuais conforme Objetivos de Nível de Serviço (SLOs) definidos pelo usuário. A arquitetura proposta busca simplificar o gerenciamento de recursos e melhorar a eficiência da infraestrutura, sem exigir modificações nas aplicações.*

1. Introdução

Através da alocação dinâmica de recursos, possibilitada por tecnologias de virtualização, a computação em nuvem transformou a forma como aplicações são implantadas e executadas. Uma de suas principais características é a elasticidade, que permite que recursos (como *CPU* e memória) sejam adaptados de acordo com a variação na carga de trabalho [Buyya et al. 2013]. Essa capacidade se torna particularmente relevante para aplicações que utilizam *streaming* paralelo, que processam dados continuamente e apresentam flutuações na carga de trabalho, impactando a performance do sistema e a utilização eficiente de recursos [Griebler et al. 2019].

Gerenciar os recursos computacionais visando manter objetivos a nível de serviço (*SLOs*) em tais ambientes torna-se desafiador. Frequentemente, desenvolvedores precisam implementar mecanismos de monitoramento e adaptação para garantir desempenho adequado, evitando o superprovisionamento de recursos. Estes desafios se tornam mais significativos em sistemas paralelos modernos, nos quais é essencial equilibrar desempenho, consumo de recursos e escalabilidade [Vogel et al. 2021] [Griebler et al. 2019].

No campo da programação paralela, a elasticidade vertical tem sido subexplorada. Abordagens existentes, com frequência, dependem de estratégias de escalonamento horizontal ou exigem modificações significativas na aplicação para possibilitar a adaptação dinâmica de recursos [Vogel et al. 2021]. Como resultado, explorar a elasticidade vertical em ambientes de nuvem permanece uma tarefa desafiadora.

Para abordar esses desafios, este trabalho propõe um *middleware* para automatizar a elasticidade vertical em ambientes de nuvem. O sistema monitora o desempenho da aplicação, avalia sua conformidade com os *SLOs* e adapta dinamicamente os recursos de *CPU* e de memória em máquinas virtuais. Operando no nível de infraestrutura, o *middleware* permite que desenvolvedores se beneficiem de um gerenciamento autônomo de recursos sem a necessidade de modificar a aplicação. Em contraste com abordagens

existentes, a solução proposta foca em fornecer uma arquitetura leve e modular, capaz de gerenciar elasticidade vertical para aplicações de *streaming* paralelo, integrando-se diretamente a plataformas de gerenciamento de nuvens privadas, como o OpenNebula.

2. Arquitetura do *Middleware*

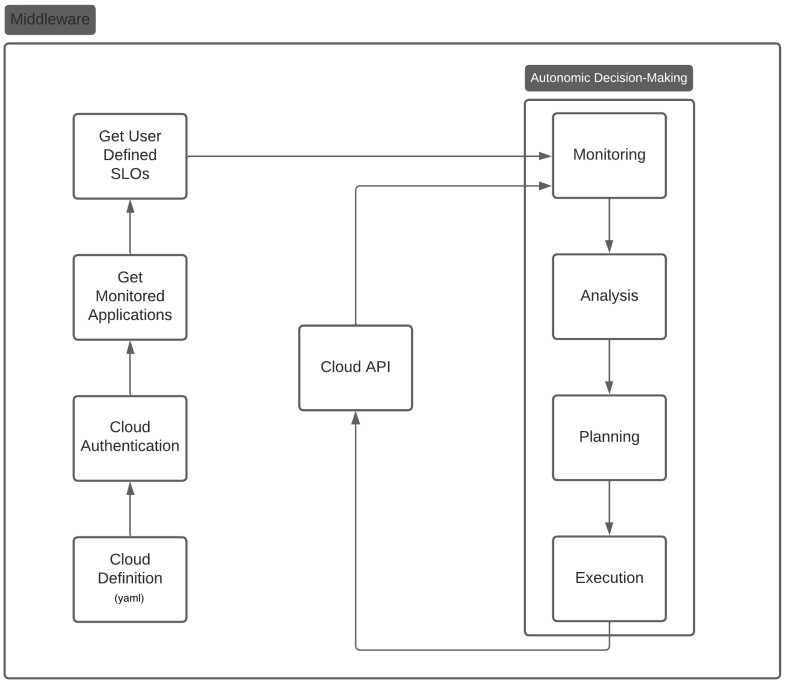

Figura 1. Arquitetura do *Middleware: Visão Geral*

O *middleware* proposto, denominado *Elastic Cloud*, implementa elasticidade vertical por meio de uma arquitetura de controle autônoma baseada no ciclo MAPE (*Monitor, Analyze, Plan, Execute*). Sua implementação inicial, em Python, foi concebida como um sistema modular que interage com infraestruturas em nuvem por meio das *APIs* de provedores. A arquitetura visa prover um mecanismo leve e extensível para gerenciar a adaptação de recursos em ambientes de nuvem privada, mantendo independência de implementações específicas de aplicações. A arquitetura foi projetada com os seguintes objetivos:

- **Modularidade**: separação dos componentes de monitoramento, análise e execução, facilitando manutenção e evolução;
- **Independência de provedor**: abstração das interações com *APIs*, permitindo integração com diferentes plataformas;
- **Adaptação não intrusiva**: aplicações não precisam ser modificadas para se beneficiar do sistema;
- **Lógica de decisão leve**: decisões baseadas em limiares simples reduzem a sobrecarga computacional.

Essas características tornam o *middleware* adequado para ambientes com carga de trabalho dinâmica e alta demanda por eficiência de recursos. A Figura 1 apresenta uma visão geral da arquitetura.

O *middleware* possui componentes responsáveis por monitorar o estado do sistema, avaliar as condições de desempenho e executar adaptações de recursos. Os principais componentes são:

- *Cloud Definition*: define parâmetros de interação com a infraestrutura, incluindo endpoints de *API*, credenciais e configurações de virtualização;
- *Cloud Authentication*: estabelece sessões autenticadas com a plataforma de nuvem;
- *Application Monitoring*: coleta métricas e metadados das máquinas virtuais e aplicações (como uso de *CPU* e memória);
- *User-defined SLOs*: armazena limites definidos pelo usuário para desempenho e uso de recursos;
- *Autonomic Decision System*: implementa o ciclo MAPE, avaliando continuamente a necessidade de adaptação;
- *Cloud API Interface*: realiza a comunicação com a plataforma de nuvem, permitindo a coleta de dados e a execução de ações.

O ciclo MAPE é executado nas seguintes etapas: *Monitor* (coleta de métricas), *Analyze* (comparação com *SLOs*), *Plan* (definição das ações) e *Execute* (aplicação das adaptações).

Essa estratégia, baseada em limites, permite adaptações leves e responsivas, adequadas a cargas de trabalho em tempo real.

3. Resultados Preliminares

Experimentos preliminares foram conduzidos para avaliar o comportamento do *middleware* em um ambiente de nuvem privada, utilizando o provedor OpenNebula e o *hypervisor* KVM. O *middleware* foi configurado como um controlador/observador externo, responsável por monitorar as métricas de *CPU* e memória e adaptar dinamicamente os recursos de máquinas virtuais de acordo com *SLOs* pré-definidos.

Os experimentos de *CPU* foram conduzidos em máquinas virtuais inicialmente configuradas com 2 núcleos de *vCPU*. A adaptação foi acionada quando o uso da CPU por núcleo excedeu 60% (aumento de recursos) ou caiu abaixo de 40% (redução de recursos), com um intervalo mínimo de 1 segundo entre adaptações.

Para avaliação, foi utilizado o *benchmark* SPBench, voltado a aplicações de *streaming* paralelo [Garcia et al. 2023], para gerar cargas controladas e medir a vazão da aplicação durante o processamento contínuo de dados. Durante os experimentos, o *middleware* monitorou o uso de recursos e adaptou o número de núcleos de *vCPU* alocados via *API* do OpenNebula.

Os resultados apresentados referem-se ao padrão de carga binário, que alternaria entre taxas de entrada altas e baixas, gerando variações abruptas de carga. A Figura 2 ilustra a relação entre o número de *vCPUs* alocados e a vazão obtida.

Os resultados demonstram que o *middleware* foi capaz de adaptar recursos de *vCPU* de acordo com as variações da carga de trabalho. Quando a taxa de entrada aumenta, o sistema detecta maior utilização de recursos e solicita alocação adicional. À medida que a carga diminui, os recursos são reduzidos, evitando superprovisionamento.

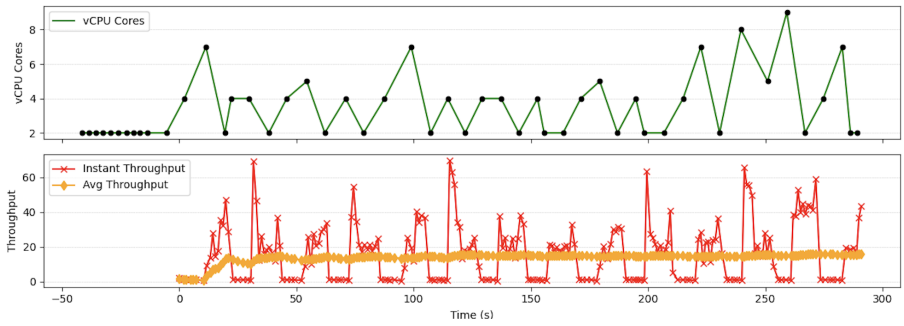

Figura 2. Padrão de Carga Binário: Vazão e núcleos de vCPU

Esses resultados preliminares indicam que o *middleware* é capaz de ajustar dinamicamente os recursos das máquinas virtuais sem impactar negativamente a vazão da aplicação, demonstrando a viabilidade de uma abordagem baseada no ciclo *MAPE* para elasticidade vertical em ambientes de nuvem.

4. Conclusão

Este trabalho apresentou uma proposta de arquitetura de *middleware* que viabiliza elasticidade vertical para aplicações executadas em ambientes de nuvem privada. O sistema implementa um ciclo de controle autônomo capaz de monitorar o desempenho da aplicação e do sistema, avaliá-los em relação aos *SLOs* definidos e adaptar dinamicamente os recursos computacionais.

Ao externalizar o gerenciamento de recursos da lógica da aplicação e integrar-se diretamente às plataformas de nuvem, o *middleware* simplifica o desenvolvimento de aplicações adaptativas, ao mesmo tempo em que melhora a utilização de recursos e mantém o desempenho do sistema.

Referências

Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering Cloud Computing: Foundations and Applications Programming*. Morgan Kaufmann, Amsterdam, The Netherlands.

Garcia, A. M., Griebler, D., Schepke, C., and Fernandes, L. G. (2023). SPBench: a framework for creating benchmarks of stream processing applications. *Computing*, 105(5):1077–1099.

Griebler, D., Vogel, A., De Sensi, D., Danelutto, M., and Fernandes, L. G. (2019). Simplifying and implementing service level objectives for stream parallelism. *Journal of Supercomputing*, 76:4603–4628.

Vogel, A., Griebler, D., Danelutto, M., and Fernandes, L. G. (2021). Self-adaptation on Parallel Stream Processing: A Systematic Review. *Concurrency and Computation: Practice and Experience*, 34(6):e6759.