Plano de Trabalho de Conclusão de Curso
Análise de arquiteturas de microsserviços empregados a jogos MMORPG voltada a otimização do uso de recursos de gerenciamento de mundos virtuais

Marlon Henry Schweigert – marlon.henry@magrathealabs.com
Charles Christian Miers – charles.miers@udesc.br *(orientador)*

Turma 2018/1 – Joinville/SC

12 de Março de 2018

Resumo

A crescente popularização de jogos massivos demanda por novas abordagens tecnológicas a fim de suprir as necessidades dos usuários com menor custo de recursos computacionais. Projetar essas arquiteturas, do ponto de vista da rede, é algo pertinente e impactante para o sucesso desses jogos. O objetivo deste trabalho é propor uma análise voltada a identificar abordagens para otimização dos recursos computacionais consumidos pelas arquiteturas identificadas. Esse objetivo será atingido após realizar uma pesquisa referenciada, seguida de uma análise das principais arquiteturas e, preferencialmente, a execução de simulações usando uma nuvem computacional para auxiliar na identificação de gargalos de recursos. Os resultados obtidos auxiliarão provedores de serviços Massively Multiplayer Online Role-Playing Game (MMORPG) a reduzir gastos de manutenção e melhorar a qualidade de tais serviços.

Palavras-chave: *arquitetura de microsserviços, desenvolvimento de jogos, rede de jogos, jogos massivos, otimização de recursos, nuvens computacionais*

1 Introdução e Justificativa

Os avanços tecnológicos de sistemas distribuídos estão permitindo que as pessoas utilizem serviços com volumes massivos de dados para aplicações sensíveis a latência. Essa situação é propícia à área de jogos massivos, e tem atraído pesquisadores para essa área de estudo [Kim, Kim e Park 2008, Huang, Ye e Cheng 2004, Saldana et al. 2012, Barri, Gine e Roig 2011]. O principal objetivo destas pesquisas é reduzir a carga e o impacto da latência para o usuário final nesses serviços. Reduzir a carga e impacto da latência em serviços para jogos massivos resulta em uma melhor experiência de jogabilidade aos usuários finais [Huang, Ye e Cheng 2004].

Jogos MMORPG são utilizados como negócio viável e lucrativo, sendo que experiência de jogabilidade na qual o usuário final será submitido é um fator crítico para o sucesso. O mercado de jogos MMORPG vem crescendo desde 2012 [Bilton 2011], sendo no ano de 2016 um dos mais lucrativos [Statista 2016]. A sua projeção para 2018 é que sejam arrecadados mais de 30 bilhões de

dólares americanos com esta categoria de jogos [Statista 2017], um aumento de 20% a mais sobre o ano de 2016.

MMORPG são jogos de interpretação de papéis massivos. A principal característica desse estilo de jogo é a comunicação e representação virtual de um mundo fantasia no qual cada jogador pode interagir com objetos virtuais compartilhados ou tomar ações sobre outros jogadores em tempo real, tendo como principais objetivos a resolução de problemas conforme a sua regra de *design*, o desenvolvimento do personagem e a interação entre os jogadores[Hanna 2015].

Um jogo MMORPG é arquitetado em duas partes [Kim, Kim e Park 2008]:

- **Serviço**: É o macrosserviço que implementa as regras de negócio e requisitos do jogo. O serviço disponibiliza uma interface com ações possíveis ao cliente sobre algum protocolo de rede.
- **Cliente**: Cliente é a aplicação que realizará as requisições com a interface do macrosserviço, exibindo o estado de jogo de forma imersiva ao jogador.

A maioria dos jogos MMORPG disponíveis no mercado estão implementados sobre uma arquitetura que executa sobre diversos servidores[Willson 2017], nos quais o desempenho destes servidores influencia tanto na experiência de jogabilidade do usuário final, quanto no custo de manutenção destes serviços [Huang, Ye e Cheng 2004]. Modelar um sistema de alto desempenho torna-se um trabalho essencial para a satisfação do usuário final [Huang, Ye e Cheng 2004]. As ocorrências geradas por um sistema de baixo desempenho podem acarretar em frustração do usuário com o serviço e/ou aumento dos gastos com recurso computacional para manter o serviço. Uma ocorrência é qualquer tipo de mal funcionamento em uma aplicação, não necessariamente aparente ao usuário final [Huang, Ye e Cheng 2004]. Evitar ou eliminar as ocorrências durante o projeto e desenvolvimento das arquiteturas do serviço é um processo crítico para o bom funcionamento desses jogos.

1.1 Ocorrências de serviços massivos

Uma métrica popular para mensurar o desempenho de um serviço MMORPG é o número de conexões [Huang, Ye e Cheng 2004] simultâneas suportadas. Em geral, caso o serviço ultrapasse o limite para o qual este foi projetado, diversas falhas de conexão, problemas de lentidão ou dessincronização com o cliente podem ocorrer. Neste contexto, as ocorrências comuns são [Huang, Ye e Cheng 2004]:

- **Longo tempo de resposta aos clientes**: implica em uma qualidade insatisfatória de jogabilidade ao usuário ou até mesmo impossibilitando o uso do serviço.
- **Dessincronização com os clientes**: realiza reversão na aplicação. Reversão é definida pela situação na qual uma requisição é solicitada ao servidor, um pré-processamento aparente é executado e essa requisição é negada, sendo necessário desfazer o pré-processamento aparente realizado ao cliente.

- **Problemas internos ao serviço**: podem estar relacionados a diversos outros erros internos de implementação ou a capacidade de recurso computacional (*e.g.,* sobrecarga no banco de dados, gerenciamento lento do espaço ou inconsistências dentro do jogo perante a regra de negócios).
- **Falha de conexão entre o cliente e os microsserviços**: causa a negação de serviço ao usuário final.

Existem algumas causas comuns para essas as ocorrências descritas [Huang, Ye e Cheng 2004]:

- **Baixo poder computacional do servidor**: poder computacional baixo para a qualidade de experiência de jogabilidade do usuário final desejada.
- **Complexidade de algoritmos**: o serviço usa algoritmos de alta complexidade ou regras de negócios que demandam por um algoritmo complexo.
- **Limitado pela própria arquitetura**: está limitado diretamente pelo número de conexões, não suportando a carga recebida.

Tais ocorrências estão diretamente correlacionadas a carga a qual tais serviços estão submetidos e podem ser amenizadas utilizando técnicas de provisionamento de recursos e balanceamento de carga [Huang, Ye e Cheng 2004], mas não suficiente para eliminar tais ocorrências.

A área de desenvolvimento web compartilha várias ocorrências comuns geradas por sobrecarga do serviço [Khazaei et al. 2016]. Em desenvolvimento web é comum utilizar a abordagem de microsserviços para resolver o problema de sobrecarga, modularizando o funcionamento em módulos menores. Da mesma forma, faz sentido modularizar um serviço MMORPG em microsserviços para suportar cargas maiores e diminuir o custo de manutenção [Villamizar et al. 2016].

1.2 Arquiteturas de microsserviços

Entende-se por microsserviço as aplicações que executam operações menores de um macrosserviço, da melhor forma possível [Willson 2017]. Microsserviços devem funcionar separadamente e de forma autônoma. Seu funcionamento deve ser desenhado para permitir alinhamentos de alta coesão e baixo acoplamento entre os demais microsserviços existentes em um macrosserviço [Acevedo, Jorge e Patiño 2017].

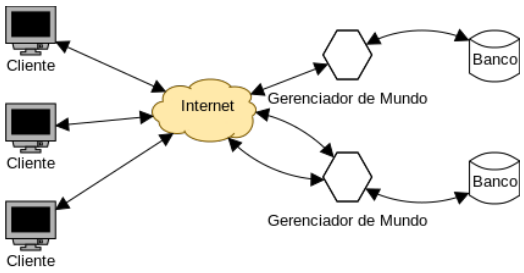
Arquiteturas de microsserviços iniciam uma nova linha de desenvolvimento de aplicações preparadas para executar sobre nuvens computacionais, promovendo maior flexibilidade, escalabilidade, gerenciamento e desempenho, sendo a principal escolha de arquitetura de grandes empresas como Amazon, Netflix e LinkedIn [Khazaei et al. 2016, Villamizar et al. 2016]. Um microsserviço é definido pelas seguintes características [Acevedo, Jorge e Patiño 2017]:

- Deve posibilitar a implementação como uma peça individual do macrosserviço.
- Deve funcionar individualmente.

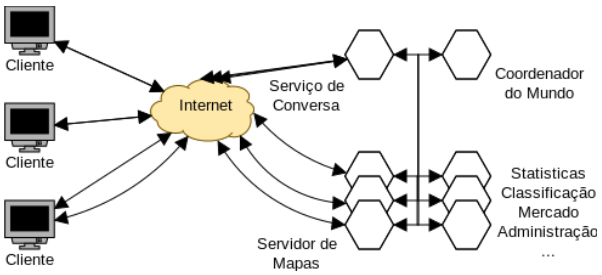
- Cada serviço deve ter uma interface. Essa interface deve ser o suficiente para utilizar o microsserviço.
- A interface deve estar disponível na rede para chamada de processamento remoto.
- O serviço pode ser utilizado por qualquer linguagem de programação e/ou plataforma.
- O serviço deve executar com as dependências mínimas.
- Ao agregar vários microsserviços, o macrosserviço resultante poderá prover funcionalidades complexas.

Alguns exemplos de arquitetura de microsserviços para jogos MMORPG são as arquiteturas apresentadas por Rudy (Figura 1), Salz (Figura 2) e a arquitetura escrita por Knowles (Figura 3).

Figura 1: Arquitetura Rudy, utilizada no jogo Tibia. **Figura 2:** Arquitetura Salz, utilizada no jogo Albion.

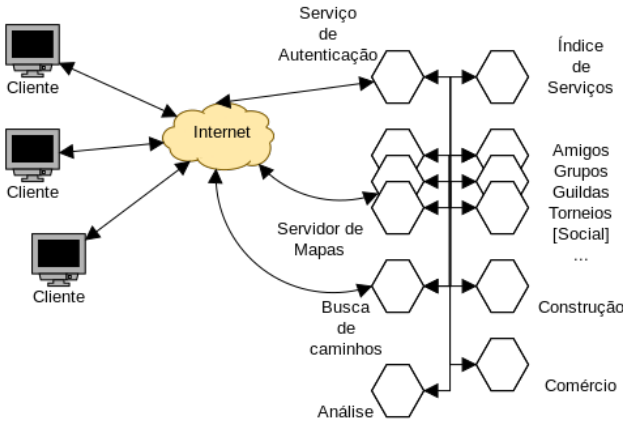


Adaptado de: [Ruddy 2011]



Adaptado de: [Salz 2016]

Figura 3: Arquitetura Knowles, utilizada no jogo Guild Wars 2.



Adaptado de: [Willson 2017]

A arquitetura Rudy (Figura 1) é formada por um sistema cliente-servidor monolítico, na qual cada microsserviço individual gerencia um mundo mútuo dos demais gerenciadores de mundo [Ruddy 2011]. Essa arquitetura dificulta a escalabilidade, modificações e manutenção [Acevedo, Jorge e Patiño 2017], além de segregar a comunidade de jogadores em servidores menores [Ruddy 2011]. Inicialmente essa arquitetura foi pensada para ser um sistema Cliente-Servidor monolítico. A arquitetura Rudy é uma arquitetura de microsserviços adaptada de um serviço cliente-servidor [Ruddy 2011]. O jogo Tibia[1],

[1]Tibia: `http://www.tibia.com`

operante sobre essa arquitetura, possui 68 mundos oficiais (Sendo 2 servidores de teste) [Ruddy 2011], com capacidade para 1.050 clientes em cada servidor, na qual encontra-se restringido pelo gerenciador de mundo.

A arquitetura Salz (Figura 2) é formada por diversos microsserviços [Salz 2016]. O principal objetivo dessa arquitetura é modularizar o serviço visando melhorar a escalabilidade. Ela é atualmente utilizada no jogo Albion Online[2]. A arquitetura é planejada para funcionar conforme a seguinte especificação[Salz 2016]:

- O mundo é distribuído sobre os vários servidores de mapas. Cada microsserviço gerencia uma região do mundo, denominado *chunk*.
- Jogadores mudam a conexão com os microsserviços quando estão posicionados na borda de um *chunk*.
- A autorização de acesso aos microsserviços é obtido pelo banco de dados.
- O coordenador do mundo é responsável por tudo que seja de escopo global (*e. g.,* Grupos, chat global, guildas, *etc.*).

A arquitetura Knowles (Figura 3) é distribuída em diversos microsserviços, assim como a arquitetura Salz (Figura 2). A diferença em comparação a arquitetura Salz (Figura 2) está na decomposição da arquitetura para outros microsserviços e a conexão direta entre esses microsserviços e o cliente. O principal objetivo dessa arquitetura é facilitar a manutenção e desempenho de reinicialização do macrosserviço [Willson 2017]. Guild Wars 2[3] é um jogo que executa sobre a arquitetura Knowles. Ele é popularmente conhecido por ter seus serviços sempre ativos, visto que a arquitetura possibilita desativar pequenos pedaços do serviço para manutenções básicas e a sua reinicialização é rápida para manutenções críticas.

Conforme as arquiteturas de microsserviços dos jogos MMORPG aumentam para atender novas demandas do mercado, o seu custo computacional e complexidade de nós na nuvem computacional aumentam. Por esse motivo a análise e otimização dessas arquiteturas é importante para impactar em um melhor desempenho tanto ao usuário final, quanto a otimização de lucros das empresas que disponibilizam esses serviços.

1.3 Justificativa

A proposta de otimização das análises realizadas sobre as arquiteturas de microsserviços para jogos massivos focada ao gerenciamento de mundos virtuais, proposta pela literatura, traz impacto direto ao usuário final e a empresa que mantem esse sistema [Huang, Ye e Cheng 2004].

A proposta contida no atual trabalho visa indentificar meios para otimizar tais arquiteturas, visando melhorar algumas características específicas:

- Recursos utilizados.

[2]Albion Online: `https://albiononline.com`
[3]Guild Wars 2: `https://www.guildwars2.com`

- Escalabilidade.
- Disponibilidade.
- Flexibilidade.

2 Objetivos

Análisar e definir arquiteturas de microsserviços empregados a jogos MMORPG voltada a otimização do uso de recursos (*e.g., memória e processamento* [Huang, Ye e Cheng 2004]) de gerenciamento de mundos virtuais. Identificar e analisar soluções tangíveis (*e.g.,* protocolos, compressão, paralelismo, *etc.* [Huang, Ye e Cheng 2004]) para otimizar os recursos utilizados pelas arquiteturas analisadas.

Os objetivos específicos são:

- Identificar e definir arquiteturas empregadas na categoria de jogos do presente trabalho.
- Identificar e definir os protocolos utilizados nessas arquiteturas.
- Identificar e definir os microsserviços dessas arquiteturas.
- Identificar e analisar ferramentas de análise de recursos para definir métricas as arquiteturas identificadas e caracterizadas.
- Especificar requisitos para a arquitetura estudada.
- Aplicar as arquiteturas descritas na literatura em uma nuvem de computadores *OpenStack*.
- Analisar o comportamento das arquiteturas aplicadas, levantando questões de desempenho e recursos utilizados.
- Propor alternativas de otimização para os problemas encontrados nas devidas arquiteturas.

3 Metodologia

Para que seja possível atingir os objetivos, serão utilizados dois métodos: pesquisa referenciada, desenvolvida durante o Trabalho de Conclusão de Curso I, e pesquisa aplicada, desenvolvida durante o Trabalho de Conclusão de Curso II.

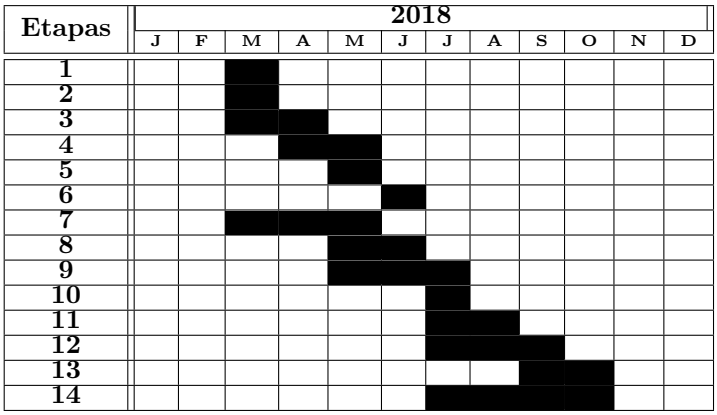
Na pequisa referenciada serão levantadas arquiteturas da literatura, buscando as mais adequadas ao escopo deste trabalho. Será dividido em quatro etapas: levantamento e especificação de arquiteturas de microsserviços descritas na literatura. Por fim, o levantamento e especificação de possíveis simulações para efetuar testes durante a pesquisa aplicada.

Na pesquisa aplicada, o resultado a ser obtido é a análise das arquiteturas de microsserviços definidas, visando uma análise sobre os recursos computacionais consumidos e identificação de seus gargalos. Divide-se em três etapas: aplicação das arquiteturas descritas e selecionadas, realização dos testes utilizando a simulação descrita e análise dos dados coletados.

Para que os resultados sejam alcançados, são definidas as seguintes etapas:

1. **Levantamento e fichamento das referências:** Pesquisa de fontes para embasamento teórico do trabalho, com base nos objetivos específicos;
2. **Consolidação das referências:** Compreensão e seleção de artefatos literários que permitam atingir o objetivo do Trabalho de Conclusão de Curso I;
3. **Identificação e definição de arquiteturas descritas na literatura:** Enumeração e definir das arquiteturas de microsserviços descritas na literatura, bem como os seus objetivos;
4. **Especificação das arquiteturas selecionadas:** Especificar o funcionamento das arquiteturas selecionadas.
5. **Identificação e definição de simulações aplicáveis ao teste:** Eleger e definir a simulação a ser aplicada nos testes;
6. **Especificação da simulação elegida:** Especificar os requisitos;
7. **Escrita Trabalho de Conclusão de Curso I**;
8. **Desenvolvimento da simulação:** Desenvolvimento da simulação para interagir com as arquiteturas de microsserviços;
9. **Desenvolvimento da arquitetura:** Desenvolvimento da arquitetura para executar os testes;
10. **Aplicação das arquiteturas selecionadas na pesquisa referênciada:** Aplicação das arquiteturas descritas sobre uma nuvem computacional;
11. **Realização dos testes utilizando a simulação elegida na pesquisa referênciada:** Execução de testes da arquitetura desenvolvida sobre a nuvem computacional;
12. **Análise das arquiteturas testadas:** Analisar as métricas obtidas dos testes e descrever resultados, identificando possíveis gargalos nas arquiteturas;
13. **Otimização para melhorar as métricas obtidas:** Identificar pontos de gargalo nos microsserviços identificados e propor soluções viáveis para aumentar o desempenho desses sistemas.
14. **Escrita Trabalho de Conclusão de Curso II**;

4 Cronograma proposto



5 Linha de Pesquisa

Este trabalho será desenvolvido junto ao Grupo de Redes e Aplicações Distribuídas (GRADIS) e ao Laboratório de Processamento Paralelo e Distribuído (LabP2D). Esta pesquisa abrange as áreas de Redes de Computadores, Sistemas Distribuídos, Segurança em Redes de Computadores, Processamento Paralelo e Engenharia de Software.

6 Forma de Acompanhamento/Orientação

O acompanhamento será realizado principalmente através de reuniões semanais ou quinzenais, com duração máxima de 1 (uma) hora. Eventualmente as reuniões poderam ser trocadas por vídeo-conferência, troca de mensagens de correio eletrônico ou telefone. O controle das tarefas será feito baseado em uma ata gerada a cada reunião. Metas semanais ou quinzenais serão atribuídas para melhor acompanhamento.

Referências

[Acevedo, Jorge e Patiño 2017]ACEVEDO, C. A. J.; JORGE, J. P. G. y; PATIñO, I. R. Methodology to transform a monolithic software into a microservice architecture. In: *2017 6th International Conference on Software Process Improvement (CIMPS)*. Zacatecas, Mexico: IEEE, 2017. p. 1–6.

[Barri, Gine e Roig 2011]BARRI, I.; GINE, F.; ROIG, C. A hybrid p2p system to support mmorpg playability. In: *2011 IEEE International Conference on High Performance Computing and Communications*. Banff, Alberta, Canada: IEEE, 2011. p. 569–574.

[Bilton 2011]BILTON, N. *Search Bits SEARCH Video Game Industry Continues Major Growth, Gartner Says*. 2011. Acessado em: 19/01/2018. Disponível em: <https://bits.blogs.nytimes.com/2011/07/05/video-game-industry-continues-major-growth-gartner-says/>.

[Hanna 2015]HANNA, P. *Video Game Technologies*. 2015. Acessado em: 19/01/2018. Disponível em: <https://www.di.ubi.pt/ agomes/tjv/teoricas/01-genres.pdf>.

[Huang, Ye e Cheng 2004]HUANG, G.; YE, M.; CHENG, L. Modeling system performance in mmorpg. In: *IEEE Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004.* Northwestern University, USA: IEEE, 2004. p. 512–518.

[Khazaei et al. 2016]KHAZAEI, H. et al. Efficiency analysis of provisioning microservices. In: *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. Luxembourg, Austria: IEEE, 2016. p. 261–268.

[Kim, Kim e Park 2008]KIM, J. Y.; KIM, J. R.; PARK, C. J. Methodology for verifying the load limit point and bottle-neck of a game server using the large scale virtual clients. In: *2008 10th International Conference on Advanced Communication Technology*. Phoenix Park, Korea: IEEE, 2008. v. 1, p. 382–386. ISSN 1738-9445.

[Ruddy 2011]RUDDY, M. *Inside Tibia, The Technical Infrastructure of an MMORPG*. 2011. Disponível em: <http://twvideo01.ubm-us.net/o1/vault/gdceurope2011/slides/Matthias_Rudy_ProgrammingTrack_InsideTibiaArchitecture.pdf>.

[Saldana et al. 2012]SALDANA, J. et al. Traffic optimization for tcp-based massive multiplayer online games. In: *2012 International Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS)*. Genoa, Italy: IEEE, 2012. p. 1–8.

[Salz 2016]SALZ, D. *Albion Online - A Cross-Platform MMO (Unite Europe 2016, Amsterdam)*. 2016. Disponível em: <https://www.slideshare.net/davidsalz54/albion-online-a-crossplatform-mmo-unite-europe-2016-amsterdam>.

[Statista 2016]STATISTA. *Statistics and Facts on MMO/MMORPG gaming*. 2016. Acessado em: 19/01/2018. Disponível em: <https://www.statista.com/topics/2290/mmo-gaming/>.

[Statista 2017]STATISTA. *Games market revenue worldwide in 2015, 2016 and 2018, by segment and screen (in billion U.S. dollars)*. 2017. Acessado em: 19/01/2018. Disponível em: <https://www.statista.com/statistics/278181/video-games-revenue-worldwide-from-2012-to-2015-by-source/>.

[Villamizar et al. 2016]VILLAMIZAR, M. et al. Infrastructure cost comparison of running web applications in the cloud using aws lambda and monolithic and microservice architectures. In: *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. Cartagena, Colombia: IEEE, 2016. p. 179–182.

[Willson 2017]WILLSON, S. C. *Guild Wars Microservices and 24/7 Uptime*. 2017. Disponível em: <http://twvideo01.ubm-us.net/o1/vault/gdc2017/Presentations/Clarke-Willson_Guild Wars 2 microservices.pdf>.

Charles Christian Miers

Marlon Henry Schweigert

Rafael Rodrigues Obelheiro
(Coordenador do GRADIS)