Marlon Henry Schweigert

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

Joinville

2018

UNIVERSIDADE DO ESTADO DE SANTA CATARINA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Marlon Henry Schweigert

ANÁLISE DE ARQUITETURAS DE MICROSSERVIÇOS EMPREGADOS A JOGOS MMORPG VOLTADA A OTIMIZAÇÃO DO USO DE RECURSOS DE GERENCIAMENTO DE MUNDOS VIRTUAIS

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

Charles Christian Miers
Orientador

Joinville, Junho de 2018

ANÁLISE DE ARQUITETURAS DE MICROSSERVIÇOS EMPREGADOS A JOGOS MMORPG VOLTADA A OTIMIZAÇÃO DO USO DE RECURSOS DE GERENCIAMENTO DE MUNDOS VIRTUAIS

Marlon Henry Schweigert

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação Integral do CCT/UDESC.

Banca Examinadora

Charles Christian Miers - Doutor (orientador)

Débora Cabral Nazário - Doutora

Guilherme Piegas Koslovski - Doutor

Agradecimentos

AGRADECIMENTOS

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-chaves: Arquitetura de microsserviços, Desenvolvimento de jogos, Rede de jogos, Jogos massivos, Otimização de recursos, Nuvens computacionais

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords: Cloud computing, Traffic characterization, Management network, Traffic monitoring system, Performance analysis, OpenStack.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

API *Application Programming Interface*

AWS *Amazon Web Services*

CPU *Central Processing Unit*

CRUD *Create Read Update Delete*

C/S Cliente/Servidor

FPS *First-person shooter*

HTTP *Hypertext Transfer Protocol*

JSON *JavaScript Object Notation*

JWT *JSON Web Token*

LAN *Local Area Network*

MMO *Massively Multiplayer Online*

MMORPG *Massively Multiplayer Online Role-Playing Game*

MOBA *Multiplayer Online Battle Arena*

MVC *Model-View-Controller*

NPCs *Non-Playable Characters*

P2P *Peer-to-peer*

PaaS *Platform as a Service*

POF *Point of View*

REST *Representational State Transfer*

RPC *Remote Procedure Call*

RPG *Role-Playing Game*

RTS *Real-Time Strategy*

TCP *Transmission Control Protocol*

TPS *Third-person Shooter*

UDP *User Datagram Protocol*

WS *Web Services*

1 Introdução

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2 Fundamentação Teórica

2.1 Jogos Eletrônicos

O primeiro sistema de entretenimento interativo foi construído em 1947, utilizando como base de exibição um tubo de raios catódicos. Essa criação foi patenteada em janeiro de 1948, datando então o início dos jogos eletrônicos (ADAMS, 2014; GOLDSMITH, 1947).

O jogo eletrônico, ou entretenimento interativo, é uma atividade intelectual que integra um sistema de regras, na qual utiliza tal sistema a fim de definir seus objetivos ou pontuação por meio de um computador com o objetivo de despertar alguma emoção ao jogador (HANNA, 2015). Os jogos eletrônicos são aplicações convencionais, que executam sobre algum sistema operacional ou hardware apropriado a este fim. O sistema operacional, hardware ou base de execução da aplicação gráfica define a sua plataforma (*e.g.,* GNU/Linux, MS-Windows, Sony PS4, MS-XBox, web, etc.) (ADAMS, 2006).

Inicialmente os jogos eram implementados de forma simples por conta da limitação de hardware das plataformas dos anos 80. As implementações de jogos para *videogames* eram desenhadas diretamente para algum hardware proprietário, sem sistema operacional, por muitas vezes sem utilizar comunicação por rede ou memória de disco (ROLLINGS; ADAMS, 2003). Além de diversas plataformas não terem acesso a rede, os serviços para jogos eram inviabilizados pelo custo de manutenção e pela ausência de demanda a qual teriam os requisitos mínimos para jogar (ADAMS, 2006). Na década de 80, o *videogame* Atari foi uma plataforma popular, vendendo 30.000 unidades em seu lançamento contra apenas 2.000 unidades do seu concorrente Intellivision (YARUSSO, 2006).

O crescente recurso computacional disponível em computadores pessoais e *videogames* após os anos 90 permitiu que desenvolvedores criassem novos estilos de jogos que utilizavam de hardware mais especificado (ADAMS, 2006). Dentre esses hardwares, iniciou-se o uso da rede de computadores para prover a interação entre jogadores de má-

quinas distintas (STATISTA, 2018a). Jogos como EA Habitat[1], CipSoft Tibia[2] e Jajex Runescape[3] começam a utilizar, como requisito obrigatório do jogo, a conexão com a Internet para interagir em um mundo compartilhado com outos jogadores. Tais jogos popularizaram um novo gênero, trazendo inovação em sua de jogabilidade e desafio proposto ao jogar com milhares de jogadores (GUINNESS, 2013; HUANG; YE; CHENG, 2004), criando o gênero de jogos *Massively Multiplayer Online* (MMO) na árvore de gêneros.

Nesse sentido, as redes de computadores serviram como impulsionador para várias categorias de jogos que antes não eram possíveis por conta da limitação de comunicação entre computadores. Sendo assim, torna-se necessário ter uma visão geral das principais categorias de jogos eletrônicos.

2.1.1 Árvore de gêneros de jogos eletrônicos

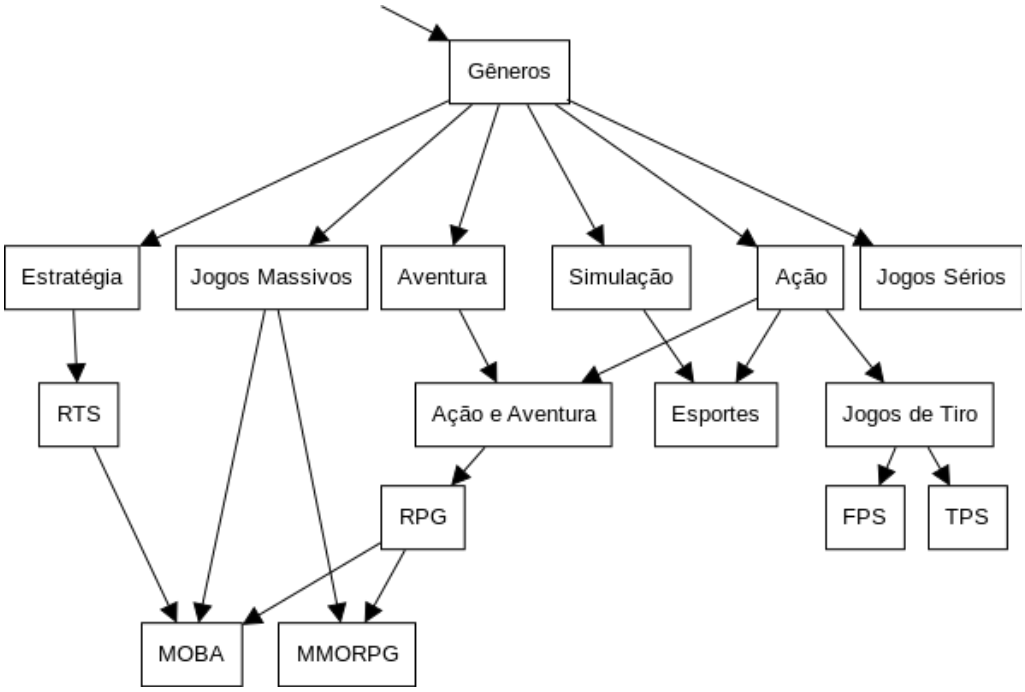
A classificação por gênero é uma ferramenta tradicional para auxiliar a fácil identificação de características de alguma literatura, arte e outras mídias. Dentro de jogos eletrônicos, o gênero permite que jogadores comprem jogos com características próximas conforme o seu gosto (CLARKE; LEE; CLARK, 2015).

Um gênero de jogo eletrônico é uma categoria específica para agrupar estilos de jogabilidade parecidos. Porém, gêneros não definem de forma explícita o conteúdo expresso em algum jogo eletrônico, mas sim um desafio comum presente no jogo analisado (ADAMS, 2006; HANNA, 2015). Cada gênero de jogo contém variações, para uma melhor classificação. A árvore pode ser visualizada pelo diagrama na Figura 2.1. O contexto breve de cada gênero é:

- Estratégia: São focados em uma jogabilidade que exija habilidades de raciocínio e/ou gerenciamento de recurso. Neste gênero, o jogador tem uma boa visualização do mundo, controlando indiretamente as suas tropas disponíveis (ROLLINGS; ADAMS, 2003). É comum encontrar jogos que disponibilizam algum modo de competição entre jogadores usando *Local Area Network* (LAN), Cliente/Servidor (C/S) ou *Peer-to-peer* (P2P) (ADAMS, 2006).

[1]EA Habitat: **http://www.mobygames.com/game/c64/habitat/credits**
[2]CipSoft Tibia: **http://www.tibia.com/**
[3]Jajex Runescape: **https://www.runescape.com**

Figura 2.1: Árvore de gêneros de jogos eletrônicos simplificada.



Adaptado de: (ADAMS, 2006)

  - *Real-Time Strategy* (RTS): Utiliza as características de um jogo de estratégia, porém esse subgênero indica que as jogadas dos jogadores não são atômicas. É comum encontrar modos de jogo competitivo utilizando LAN neste gênero (ADAMS, 2006).

- MMO: Preza pela interação com outros jogadores em um mundo compartilhado (ADAMS, 2006). SecondLife[4] é um jogo focado na interação social, com artifícios de comércio e relacionamentos em um mundo fictício criado pela comunidade (KLEINA, 2018). Em grande parte, esses jogos utilizam tecnologia C/S e *Web Services* (WS).

  - *Multiplayer Online Battle Arena* (MOBA): Coloca um número fixo de jogadores separados em dois times, no qual o time com maior estratégia de posicionamento e gerenciamento de recursos em equipe ganha a partida. Jogos MOBA perdem algumas características breves do gênero *Role-Playing Game* (RPG), deixando de lado a interpretação e contextualização de um mundo, fixando-se somente em um combate estratégico e momentânio (distribuído em partidas átomas) entre as equipes, carregando consigo somente as características de comércio e comunidade dos jogos MMO (ADAMS, 2006). Tal subgênero é po-

[4]SecondLife: **https://www.secondlife.com/**

pularmente conhecido pelos títulos Blizzard Dota 2[5] e Riot League of Legends[6]. O jogo League of Legends obteve 100 milhões de usuários ativos em 2016 (STATISTA, 2018c), além de ter um torneio nacional e mundial (SPORTV, 2018). É popular nesse subgênero utilizar tecnologias como LAN, P2P e C/S.

  - MMORPG: Herda características dos gêneros ação e aventura, RPG, e MMO diretamente. Nesse gênero se faz permitido interações em um mundo na qual outros jogadores também estão jogando, na qual a interação entre outros jogadores (herdado dos jogos MMO), com o mundo (herdado dos jogos de ação e aventura) e com objetivos guiados por *Non-Playable Characters* (NPCs) (herdados de jogos RPG) se faz como desafio e objetivo do jogo (ADAMS, 2006). Um título popular para esse gênero é o jogo Word of WarCraft[7]. A grande parte dos jogos MMORPG utilizam tecnologia C/S.

- Aventura: Caracterizado por desafios envolvendo ações com diversos NPCs ou com o ambiente para solucionar desafios (ADAMS, 2006). A grande parte desses jogos utilizam arquiteturas C/S, P2P ou LAN.

  - Ação e Aventura: Herda características da categoria de Ação e Aventura. O jogador é imerso em um mundo para iteragir com o ambiente e com NPCs, além de se preocupar com a movimentação no cenário (ADAMS, 2006). Um título popularmente conhecido desse gênero é a série de jogos nomeada Nintendo The Legend of Zelda[8]. É comum nesses jogos encontrar tecnologia LAN ou P2P para modo de jogo cooperativo.

- Simulação: Caracterizados por abordar temas da realidade. São comuns jogos de construção e gerenciamento, animais de estimação, vida social e simulação de veículos (ADAMS, 2006). A grande parte desses jogos não permite a interação entre os demais jogadores. É popular encontrar serviços como *ranking*, loja e janela de notícias utilizando WS.

  - Esportes: Trata somente da simulação de esportes, nos quais o(s) time(s) podem ser controlados tanto por uma inteligência artificial quanto por jogadores online (ADAMS, 2006). O jogo FIFA[9] é um título popular nesse segmento. É

[5]Blizzard Dota 2: **http://br.dota2.com/**
[6]Riot League of Legends: **https://br.leagueoflegends.com/pt/**
[7]Word of WarCraft: **https://worldofwarcraft.com/pt-br/**
[8]Nintendo The Legend of Zelda: **https://www.zelda.com/**
[9]FIFA: **https://www.easports.com/br/fifa**

comum encontrar tecnologias P2P e LAN.

- Ação: Preza pela habilidade de coordenação motora e reflexos do jogador, para tomar uma atitude a fim de passar seus objetivos no cenário. Nesse gênero os objetivos são passar por uma série de desafios que incluam movimentação e posicionamento de outros objetos no cenário (ADAMS, 2006). É comum encontrar tecnologias LAN, P2P, C/S e WS.
  - Jogos de Tiro: Usa um número finito de armas para executar ações a distância. O posicionamento, movimentação estratégia e mira são fatores de desafio ao jogador nesse gênero (ADAMS, 2006). É comum encontrar tecnologias LAN, P2P ou C/S.
    * *First-person shooter* (FPS): Utiliza o método de gravação conhecido como *Point of View* (POF). Nesse método, o modo de exibição do mundo é dado como a visão de um personagem do jogo, na qual o jogador tem visão pelo próprio personagem (HANNA, 2015; ADAMS, 2006). É comum encontrar tecnologias LAN, P2P ou C/S.
    * *Third-person Shooter* (TPS): Diferente dos jogos FPS, os jogos TPS utilizam cameras soltas no cenário no qual o jogador é visível na cena exibida (HANNA, 2015; ADAMS, 2006). É comum encontrar tecnologias LAN, P2P ou C/S.
- Jogos sérios: Tem como objetivo transmitir um conteúdo educacional (HANNA, 2015). O jogo Sherlock Dengue 8 (BUCHINGER, 2014) é um título desenvolvido com o objetivo de conscientizar os problemas e a prevenção da Dengue no Brasil. É comum encontrar tecnologias LAN, P2P, C/S e WS.

Dentre vários gêneros, alguns utilizam popularmente algumas tecnologias de rede. A Tabela 2.1 indica a correlação de tecnologias de rede comuns nos gêneros, além de trazer a correlação de número de jogadores por gênero de jogo. Essa correlação é importante para identificar as características de jogabilidade referente a jogabilidade com multijogadores (HANNA, 2015).

Dentre todos os jogos, o gênero MMORPG é o mais impactado pela quantia de jogadores(KIM; KIM; PARK, 2008), visível na Tabela 2.1. Por esse motivo, a escolha

Tabela 2.1: Tipos de comunicação e quantia de jogadores populares em gêneros

	LAN	P2P	C/S	WS	Jogadores
ESTRATÉGIA	✓	✓	✓		até 10
RTS	✓				até 10
MMO			✓	✓	mais que 1000
MOBA	✓	✓	✓		até 10
MMORPG			✓	✓	mais que 1000
AVENTURA	✓	✓	✓		até 10
AÇÃO	✓	✓	✓	✓	até 10
AÇÃO E AVENTURA	✓	✓	✓		até 10
SIMULAÇÃO				✓	até 10
ESPORTES	✓	✓			até 10
FPS	✓	✓	✓		até 100
TPS	✓	✓	✓		até 100
JOGOS SÉRIOS	✓	✓	✓	✓	até 10

Fonte: Elaborado pelo autor

por abordar o gênero MMORPG se torna interessante do ponto de vista computacional, a fim de analisar o comportamento de rede e processamento das arquiteturas desses jogos.

2.2 MMORPG

Jogos MMORPG são utilizados como negócio viável e lucrativo, sendo que a 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 2017 um dos mais lucrativos (STATISTA, 2018b). A projeção deste mercado para 2018 é de mais de 30 bilhões de dólares americanos com esta categoria de jogos (STATISTA, 2017), porém foi ultrapassado no ano de 2017 com 30.7 bilhões de dólares (STATISTA, 2018b).

MMORPG são jogos de interpretação de papéis massivos, originados dos gêneros RPG. 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; PARK, 2008):

- **Cliente**: Aplicação que realizará as requisições com a interface do serviço, exibindo

o estado de jogo de forma imersiva ao jogador. Este tema será melhor abordado na Seção 2.5.

- **Servidor**: Conjunto de computadores que recebe as requisições do cliente a fim de ser processadas pelo Serviço.
- **Serviço**: Implementa as regras de negócio e requisitos do jogo. O serviço disponibiliza uma interface com ações possíveis ao cliente sobre algum protocolo de rede. Este tema será melhor abordado na Seção 2.6.

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; CHENG, 2004). Por sua vez, o Cliente é implementado em algum ambiente convencional a jogos, como motores gráficos, bibliotecas gráficas ou sobre alguma outra plataforma, como web.

Nesse sentido torna-se necessário descrever as características de jogabilidade de jogos MMORPG a fim de melhor compreender o funcionamento da arquitetura de um cliente e de um serviço para jogos MMORPG nas seções seguintes.

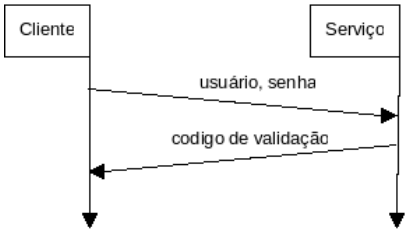
2.3 Jogabilidade de jogos MMORPG

É comum em jogos MMORPG ter sistemas com funcionalidades parecidas. Por esse motivo é fácil identificar os sistemas em jogos do mercado após a compreensão dos sistemas identificados.

O sistema de autenticação é o que inicia o cliente de algum jogo MMORPG (SALZ, 2016; RUDDY, 2011). Este sistema é implementado via protocolo web, a fim de disponibilizar um código para validar todas as futuras ações da seção do usuário. Ele pode ser visualizado de forma macro na Figura 2.2.

Esse método de autenticação é definido pela RFC7519 (JONES et al., 2015), com a tecnologia *JSON Web Token* (JWT). O código de validação repassado é auditorado em qualquer serviço pertencente ao jogo, visto que ele foi assinado pelo sistema de autenticação do serviço (IKEM, 2018).

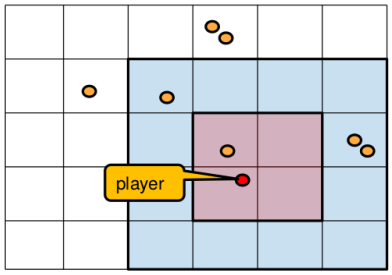
Figura 2.2: Sistema de autenticação para jogos



Fonte: Adaptado de (THOMPSON, 2008)

Após a autenticação, é comum existir um sistema para seleção de personagem, caso o jogo seja desenhado com este objetivo. Efetuada a seleção ou criação de um personagem, ele será imerso no mundo compartilhado do jogo com os demais jogadores (RUDDY, 2011).

Figura 2.3: Área de interesse com base na proximidade de um jogador



Fonte: (SALZ, 2016)

Nos jogos MMORPG é comum a restrição da visão do personagem, ora pelas características de jogabilidade do gênero MMORPG ora por motivos de desempenho e otimização. Como o jogador não precisa obter dados de regiões que não estão em sua área de interesse, não há necessidade da transmissão de informações dos objetos que estão fora desse contexto (SALZ, 2016). Esse caso pode ser visualizado na Figura 2.3, na qual o personagem selecionado (destacado em vermelho) tem uma área de interesse de baixa distância e uma área de interesse de longa distância (SALZ, 2016), sendo que o jogador não tem informações de demais objetos e jogadores fora de sua área de interesse. Esta característica impede trapaças (visto que o cliente não tem informações que só estão contidas no serviço) e reduz a frequência de atualização a cada cliente (SALZ, 2016).

Algumas ações comuns dentro do ambiente de um jogo MMORPG (JON, 2010):

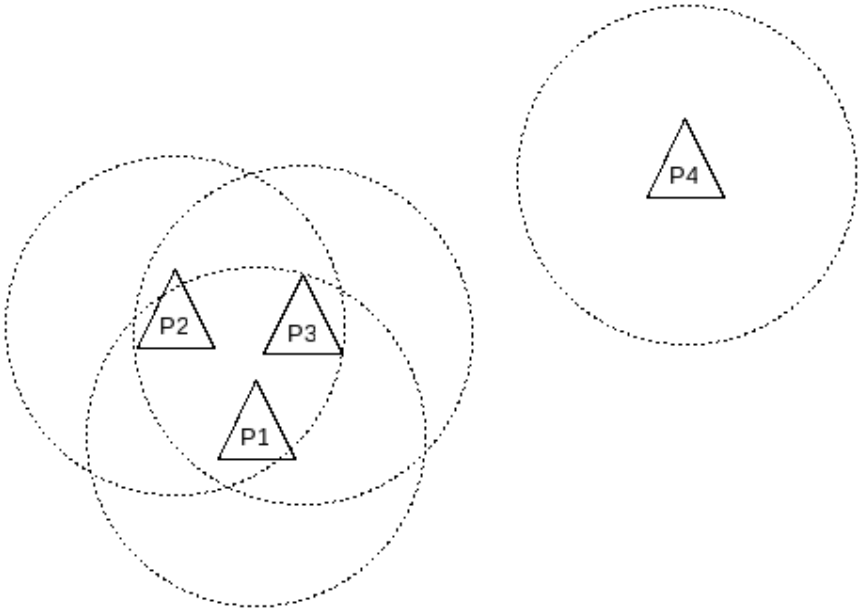
- Enviar e receber mensagem no chat;
- Mover-se pelo ambiente;

- Interagir com outros jogadores, NPCs ou objetos fixos do ambiente; e
- Obter itens do ambiente.

O envio e recepção de mensagens do chat é dado com o contexto do posicionamento do personagem (SALZ, 2016), visível na Figura 2.4. Somente outros personagens dentro de uma distância podem receber alguma mensagem emitida pelo jogador P_n.

Essa distância pode ser calculada utilizando Distância Euclidiana (DEZA, 2009), na qual a distância entre dois personagens podem ser calculadas pela equação $d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$. Para diminuir a complexidade das comparações a fim de decidir quais personagens P_n devem receber a mensagem, é comum utilizar técnicas de divisão de área utilizando algoritmos como *Quadtree* ou *Octree* (LENGYEL, 2011), subdividindo os quadrantes de uma região do ambiente do jogo a fim de facilitar a consulta de quais personagens estão em determinada área deste ambiente.

Figura 2.4: Chat baseado em contexto de posicionamento, utilizando Distância Euclidiana



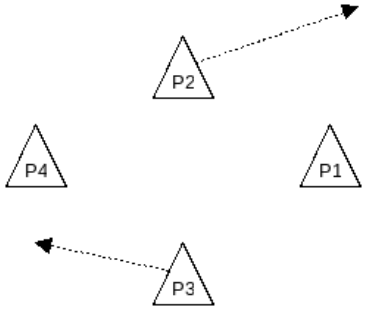
Fonte: Adaptado de (SALZ, 2016)

Nesse sentido, a Figura 2.4 mostra a interseção entre o raio de quatro personagens. Nesse exemplo, Mostra-se visível que a as mensagens de P_1 devem ser visíveis a P_2 e P_3, mas não a P_4, caso seja utilizado a Distância Euclidiana como regra de distância.

O sistema de movimento pelo ambiente do jogo possibilita que cada jogador movimente o seu personagem a fim de explorá-lo. Dessa maneira, este é um sistema crítico para um jogo MMORPG, visto que este sistema será utilizado para inúmeras consultas

de proximidade, além de ter uma frequência de atualização e consultas pelos jogadores muito frequente (SALZ, 2016).

Figura 2.5: Personagens e os seus pontos de destino



Fonte: Elaborado pelo Autor

Para os demais sistemas como combate, interação com o mundo, jogadores ou NPCs também utiliza-se do sistema de raio máximo de interação (SALZ, 2016). Este sistema não permite o personagem interagir com objetos que não estão dentro do contexto de seu posicionamento no ambiente (SALZ, 2016).

Essas operações precisam de desempenho para não causar frustração ao jogador final (HOWARD et al., 2014). Nesse sentido, torna-se necessário conhecer os problemas computacionais conhecidos com relação aos serviços de jogos MMORPG.

2.4 Problemas em jogos MMORPG

Uma métrica popular para mensurar o desempenho de um serviço MMORPG é o número de conexões (HUANG; YE; 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; 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 o serviço**: causa a negação de serviço ao usuário final.

Existem algumas causas comuns para essas as ocorrências descritas (HUANG; YE; 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; 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).

Do ponto de vista da arquitetura de computadores, as operações existentes em um jogo MMORPG seguem um padrão de interação com o mundo, criar, excluir ou manipular objetos deste mundo.

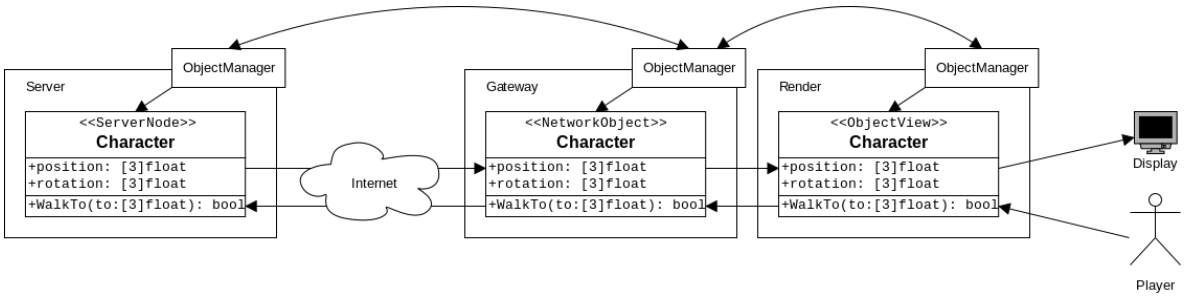
Para suprir o desenvolvimento de tais sistemas, se faz necessário compreender os padrões de desenvolvimento de tais arquiteturas na qual suprem as operações básicas de interação com o mundo.

2.5 Arquitetura de Clientes MMORPG

Do ponto de vista da rede de computadores, a arquitetura de um cliente de jogo MMORPG deve suportar consultas e chamadas de métodos remotos em um serviço (SALZ, 2016). Um cliente para um jogo MMORPG segue o estilo de arquitetura *Representational State Transfer* (REST), porém não obrigatoriamente sobre o protocolo *Hypertext Transfer Protocol* (HTTP). A fim de reduzir o custo operacional das requisições, as consultas podem ser escritas sobre um protocolo otimizado conveniente a aplicação na camada de transporte (*e.g.*,, *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), etc) (SALZ, 2016; WILLSON, 2017).

O módulo de *Gateway*, implementado dentro de um cliente de jogo MMORPG, é responsável por realizar as requisições ao servidor e aplicar as chamadas de métodos internas ao cliente para exibir o estado de jogo ao jogador (SALZ, 2016). Ele pode ser encontrado na Figura 2.6.

Figura 2.6: Requisição de uma chamada de método pela visão do cliente.



Adaptado de: (SALZ, 2016)

A Figura 2.6 refere-se a uma visão macro de uma requisição realizada pelo jogador. Nessa requisição, o jogador realiza o pedido para o seu personagem caminhar até um determinado ponto. Essa requisição será processada pelo servidor, que atualizará sua posição e retornará ao Gateway a nova posição do personagem, exibindo o personagem em uma nova posição.

Para facilitar o desenvolvimento, a aplicação de cliente é dividida em diversos módulos, sendo os dois principais (SALZ, 2016):

- **Renderização:** É o processo que transformará a estrutura de dados obtida do serviço de forma gráfica ao jogador utilizando um motor gráfico (*e.g.,* Unity3D[10],

[10]Unity3D: **https://www.unity3d.com**

GodotEngine[11], etc).

- **Gateway:** É o módulo responsável pela comunicação entre o serviço e o cliente, a fim de requisitar chamadas de métodos ou obter informações do servidor.

Também existe um módulo nomeado *ObjectManager* na Figura 2.6 que gerenciará a criação e deleção de objetos do mundo. Tanto os objetos em cena quanto o ObjectManager podem ser utilizados ora pelo jogador ora pelo serviço a fim de sincronizar o cliente ou realizar requisições ao serviço. Entretanto, existe um filtro de requisições maliciosas a fim de garantir a consistência do jogo (SALZ, 2016). Esse filtro pode ser implementado no Gateway e/ou no serviço (Seção **??**).

Uma característica importante do ObjectManager é a requisição e exibição de objetos com base na área de interesse do jogador. Essa característica reduz problemas de trapaça por meio de clientes adulterados (visto que limita a interação de objetos do cliente) e reduz o tráfego de dados na rede tanto para o cliente quanto para o serviço (SALZ, 2016; WILLSON, 2017).

A abordagem de engenharia por utilizar um Gateway permite que o serviço mude de arquitetura sem a necessidade de uma refatoração completa ao módulo de renderização (SALZ, 2016; WILLSON, 2017). Essa mesma abordagem também facilita o desenvolvimento de uma suíte de testes do módulo de renderização independente do serviço (FREEMAN; PRYCE, 2009), podendo realizar testes utilizando o padrão de objetos Mock (BECK; ANDRES, 2004) para simular as requisições. Dessa forma, é possivel garantir um padrão de integração desejado entre o serviço e o cliente utilizando baixo acoplamento, com ganho de desempenho na suíte de testes do cliente, sem a necessidade real de um serviço para testar o cliente.

2.6 Arquitetura de Microsserviços

Entende-se por microsserviço aplicações que executam operações menores de um macrosserviço, da melhor forma possível (WILLSON, 2017; NEWMAN, 2015). O objetivo de uma arquitetura de microsserviços é funcionar separadamente de forma autônoma, contendo baixo acoplamento (NEWMAN, 2015). Seu funcionamento deve ser desenhado para

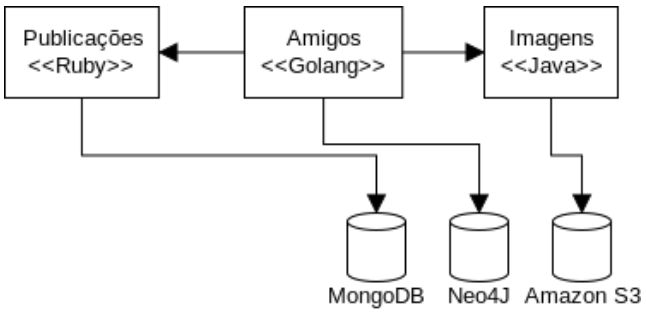
[11] GodotEngine: **https://www.godotengine.org**

permitir alinhamentos de alta coesão e baixo acoplamento entre os demais microsserviços existentes em um macrosserviço (ACEVEDO; JORGE; 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; PATIñO, 2017):

- Deve possibilitar 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 ou consulta de dados.
- 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.

Figura 2.7: Microsserviços podem ter diferentes tecnologias

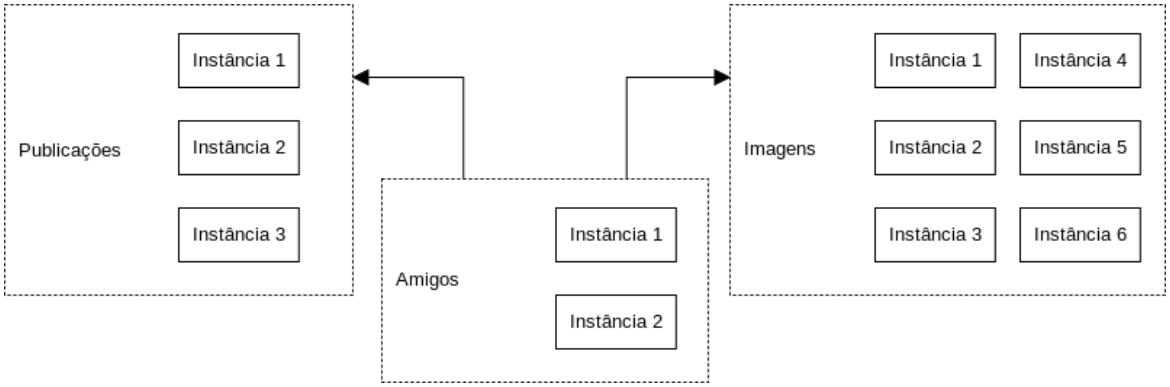


Adaptado de: (NEWMAN, 2015)

O microsserviço deverá ser uma entidade separada. Ela deve ser implantada como um sistema independente em um *Platform as a Service* (PaaS). Toda a comunicação entre os microsserviços de um macrosserviço será executada sobre a rede, a fim de

reforçar a separação entre cada serviço. As chamadas pela rede com o cliente ou entre os microsserviços será executada através de uma *Application Programming Interface* (API), permitindo a liberdade de tecnologia em que cada um será implementado (NEWMAN, 2015).

Figura 2.8: Microsserviços são escaláveis



Adaptado de: (NEWMAN, 2015)

Uma arquitetura de microsserviços é escalável, como visível na Figura 2.8. Ela permite o aumento do número de microsserviços sob demanda para suprir a necessidade de escalabilidade. Este modelo computacional obtém maior desempenho, principalmente se executar sobre plataformas de computação elástica, na qual o orquestrador do macrosserviço pode aumentar o número de instâncias conforme a necessidade de requisições (NADAREISHVILI et al., 2016).

Microsserviços desenvolvidos para web utilizam arquitetura REST baseado sobre o protocolo HTTP. É uma boa prática utilizar o corpo com conteúdo da requisição e resposta no formato *JavaScript Object Notation* (JSON) nas chamadas a uma API de microsserviço web (NADAREISHVILI et al., 2016).

Entretanto não é uma prática comum para um serviço MMORPG utilizar o protocolo HTTP pela sua elevada carga desnecessária na requisição (HUANG; YE; CHENG, 2004). Por esse motivo se faz necessário ver as diferenças entre uma arquitetura de microsserviços para MMORPG comparados a microsserviços web.

2.6.1 Protocolos para microsserviços MMORPG

Um serviço MMORPG não utilizará protocolos com base no protocolo HTTP, mas ainda assim implementará um protocolo REST sobre o protocolo TCP a fim de realizar requisições a um serviço utilizando uma arquitetura de software *Model-View-Controller*

(MVC) (CHADWICK; SNYDER; PANDA, 2012; THOMPSON, 2008). O protocolo REST permitirá ele, similar a técnica de armazenamento persistente CRUD, executar quatro métodos aos controladores de cada microsserviço (SALDANA et al., 2012):

1. Criar: Permite criar um dado no serviço (*e.g.,* criar um personagem em sua conta, criar um pedido de amizade, etc).

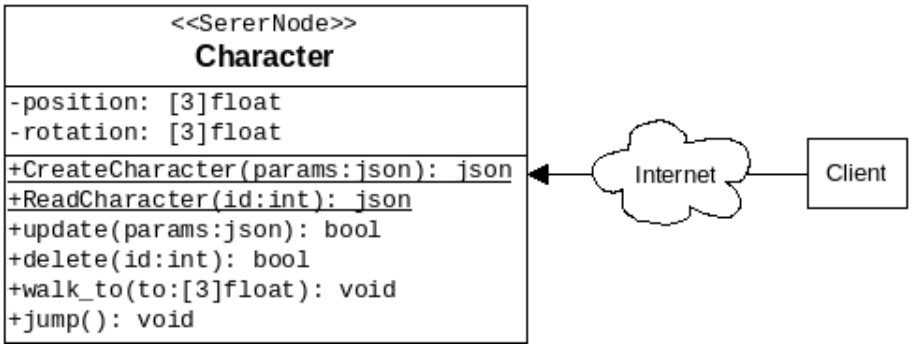
2. Ler: Permite consultar um dado no serviço (*e.g.,* ler os personagens que estão em sua região, ler os itens, etc).

3. Atualizar: Permite atualizar um dado no serviço (*e.g.,* efetuar a compra de um item de NPCs, andar para um ponto do mapa, etc).

4. Deletar: Permite deletar um dado no serviço (*e.g.,* usar um item do inventário, deletar uma mensagem lida de um amigo, etc).

Porém, além de realizar as operações CRUD, se faz necessário a chamada de métodos específicos a um objeto. Pode-se associar a técnica de RPC (THOMPSON, 2008). Pode-se analisar um exemplo da interface disponível na Figura 2.9 e o diagrama de requisições na Figura 2.10.

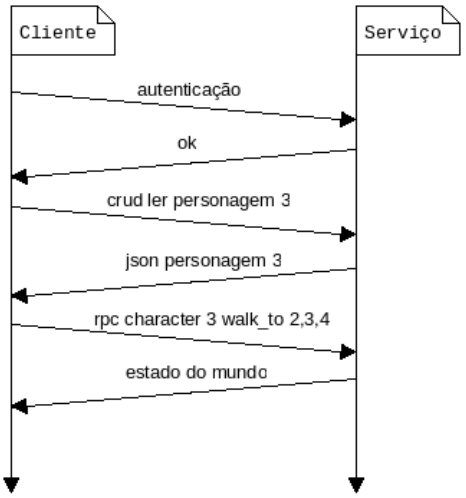
Figura 2.9: Cliente pode realizar requisições CRUD ou RPC



Fonte: Elaborado pelo autor

Uma técnica comum em jogos é a compressão de pacotes utilizando mapeamento hash de bytes (THOMPSON, 2008). Tanto o cliente quanto o serviço precisam ter a mesma estrutura de dados. Dessa forma, é possivel trocar o nome das funções requiridas em RPC por poucos bytes para transitar na rede. Já para operações CRUD, pode-se utilizar tanto requisições sobre o protocolo HTTP ou sobre um protocolo otimizado sobre TCP dependendo da necessidade de desempenho (THOMPSON, 2008).

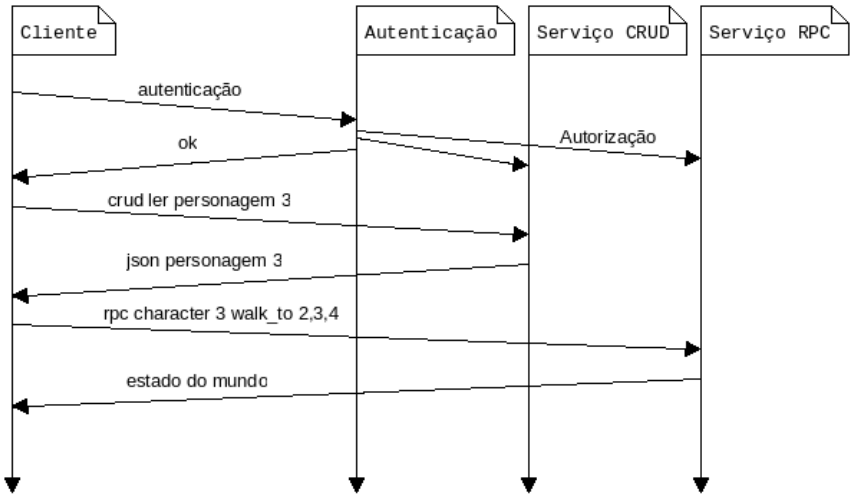
Figura 2.10: Diagrama de requisições entre serviço e cliente com operações CRUD e RPC em uma arquitetura monolítico



Fonte: Adaptado de (THOMPSON, 2008)

Como relatado na Seção 2.6, uma arquitetura de microsserviços permite multiplas tecnologias, pois a comunicação entre todos os elementos de um microsserviço será pela rede. Por esse motivo, é possível utilizar um serviço web para realizar operações CRUD e um serviço dedicado para realizar operações RPC. Essa arquitetura pode ser melhor compreendida pela Figura 2.11.

Figura 2.11: Diagrama de requisições entre serviço e cliente com operações CRUD e RPC em uma arquitetura de microsserviços



Fonte: Elaborado pelo Autor

Utilizando esse embasamento teórico sobre microsserviços e arquiteturas de jogos MMORPG, pode-se analisar os trabalhos (Seção 2.7) relacionados com o tema proposto no atual documento.

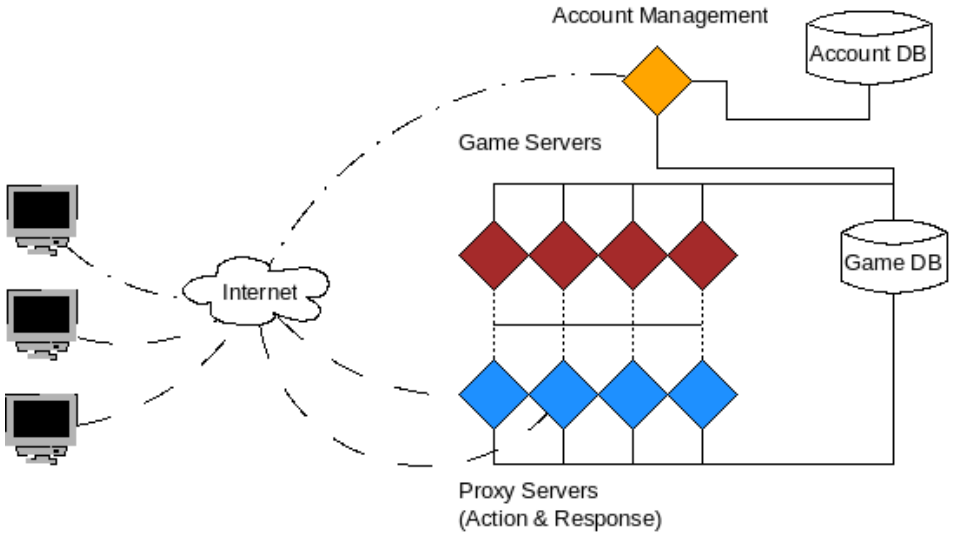
2.7 Trabalhos Relacionados

Para nortear o desenvolvimento da análise se microsserviços utilizados em jogos MMORPG proposto no atual trabalho, essa seção apresenta outros trabalhos que têm o escopo ou objetivo similar, no qual monitoraram e analisaram serviços de jogos MMORPG. Ao apresentar estes trabalhos, busca-se apresentar o contexto e objetivo, e então aprofundar em características, métricas e ferramentas que auxiliaram nas análises.

2.7.1 Huang et al. (2004)

O trabalho de (HUANG; YE; CHENG, 2004) investiga a relação entre os recursos utilizados e o número de conexões presentes em um serviço MMORPG distribuído. Neste trabalho é relatado que a infraestrutura utiliza três serviços: Um *Game Server* sobre protocolo TCP, um *Proxy Server* também sobre protocolo TCP, e um servidor web para autenticação que executa sobre uma interface HTTP. O foco de análise é o *Game Server* e o *Proxy Server*.

Figura 2.12: Arquitetura distribuída utilizando proxy

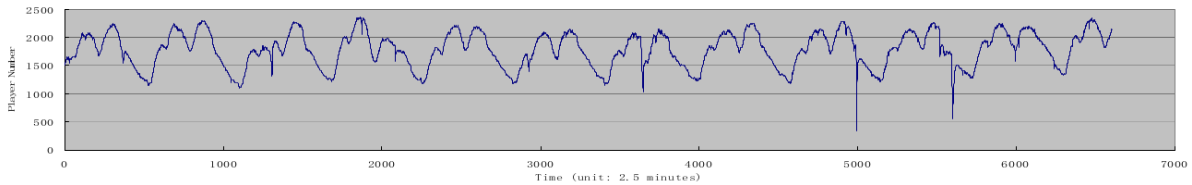


Adaptado de: (HUANG; YE; CHENG, 2004)

A infraestrutura do servidor de jogo contém um *Proxy Server Farm* utilizando o algoritmo *Round Robin* com pesos para balanceamento de carga entre cada cliente. Cada *Proxy Server* é responsável por comunicar com os demais microsserviços privados ao servidor, baseado com a área de interesse de sua conexão. O protocolo de comunicação utilizado entre o Cliente e *Proxy Server* é baseado em RPC (FARBER, 2002; BORELLA, 2000), porém não é relatado sobre o o protocolo de comunicação utilizado entre o *Proxy*

Server e o *Game Server*. A sua arquitetura pode ser observada na Figura 2.12, na qual obteve seus dados obtidos durante 100 dias para realizar as análises. A Figura 2.13 demonstra uma amostra do número de conexões pelo tempo no serviço obtido.

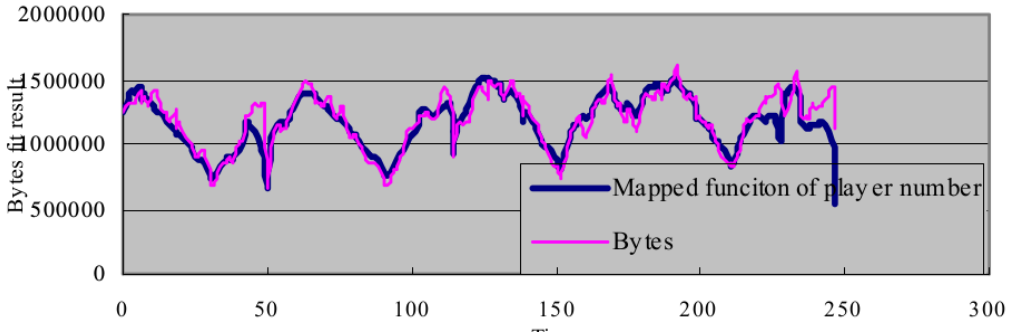
Figura 2.13: Número de conexões no serviço pelo tempo decorrido



Fonte: (HUANG; YE; CHENG, 2004)

Como análise, o autor correlacionou o número de conexões com número de pacotes, banda, memória e *Central Processing Unit* (CPU) consumidos utilizando uma função estatística linear. Esta função pode ser utilizada com regressão linear para prever consumo de recursos futuros e por fim realocar mais recursos ao serviço, contribuindo com escalabilidade vertical autônoma. Um exemplo de aplicação dessa regressão linear pode ser visualizada na Figura 2.14, onde o autor compara o consumo de banda real comparado a regressão linear.

Figura 2.14: Regressão linear comparado ao consumo de banda real do servidor



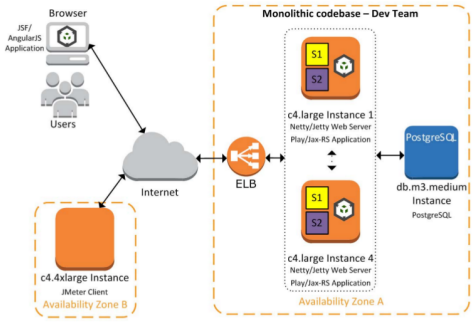
Fonte: (HUANG; YE; CHENG, 2004)

Entretanto, a escalabilidade horizontal não pode ser prevista, visto que não é analisado o posicionamento de cada personagem a fim de dividir os ambientes em pedaços menores com outros serviços. Como trabalhos futuros é relatado a análise do posicionamento de personagens para escalabilidade horizontal, a análise de outras arquiteturas e timos de jogos diferentes e qual o impácto de utilizar balanço de carga e provisionamento de recursos de forma dinâmica.

2.7.2 Villamizar et al. (2016)

O trabalho de (VILLAMIZAR et al., 2016) investiga o custo de arquiteturas de microsserviços, arquiteturas PaaS orientedas a eventos e aplicações monolíticas para aplicações web. A sua principal motivação é a comparação de custos para a tradução de sistemas legados para arquiteturas distribuídas. Para isso, o autor preparou 3 instâncias de testes com suas configurações desenhadas a fim de ter o maior número de requisições por minuto com o mesmo custo financeiro.

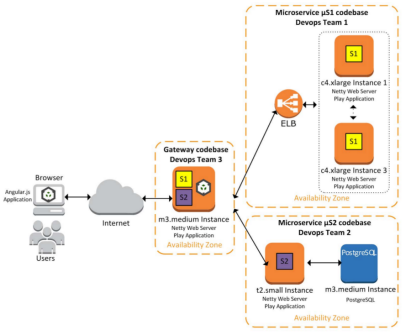
Figura 2.15: Arquitetura monolítica web implementada na AWS



Fonte: (VILLAMIZAR et al., 2016)

Instância I. Utilizando quatro instâncias AWS *c4.large*, uma instância AWS *c4.xlarge* e uma instância AWS *db.m3.medium*.. A Figura 2.15 exibe a implantação de uma aplicação web monolítica. Essa arquitetura foi implementada utilizando *Jax-RS* e *Play Framework*.

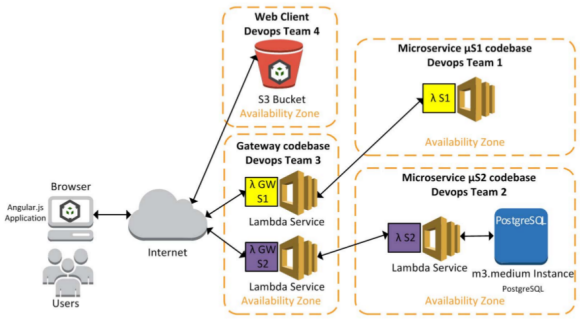
Figura 2.16: Arquitetura de microsserviços web implementada na AWS



Fonte: (VILLAMIZAR et al., 2016)

Instância II. Utilizando três instâncias AWS *c4.xlarge*, uma instância AWS *t2.small* e uma instância AWS *db.m3.medium*. A Figura 2.16 exibe a implantação de uma aplicação de microsserviços web. Essa arquitetura foi implementada utilizando *Play Framework*.

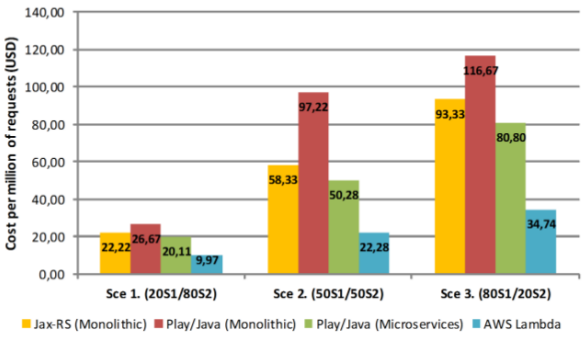
Figura 2.17: Arquitetura de microsserviços web implementada na AWS utilizando a tecnologia *lambda*



Fonte: (VILLAMIZAR et al., 2016)

Instância III. Utilizando duas instâncias AWS *lambda S1*, duas instâncias AWS *lambda S2*, uma instância AWS *S3 Bucket* e uma instância AWS *db.m3.medium*. A Figura 2.17 exibe a implantação de uma aplicação de microsserviços web utilizando a tecnologia AWS *lambda*. Essa arquitetura foi implementada em *Node.js*, onde as funções de *gateway* foram implementadas em quatro funções independentes do tipo *microservice-http-endpoint*.

Figura 2.18: Custo por por um milhão de requisições em dólares utilizando diferentes arquiteturas sobre a AWS



Fonte: (VILLAMIZAR et al., 2016)

Foi concluído que a arquitetura de microsserviços, nas condições desta aplicação, podem reduzir até 13.42% em gastos com a infraestrutura. Essa redução pode ser observada na Figura 2.18. O autor alerta sobre tolerância a falhas, transações distribuidas, distribuição de dados e versionamento de serviço.

2.7.3 Suznjevic e Matijasevic (2012)

O trabalho de (SUZNJEVIC; MATIJASEVIC, 2012)

2.7.4 Análise dos trabalhos relacionados

3 Proposta para análise de consumo de recursos computacionais

CAP 3

4 Considerações & Próximos passos

CONCLUSÃO

Referências

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.

ADAMS, A. R. E. *Fundamentals of Game Design (Game Design and Development Series)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 0131687476.

ADAMS, E. *Fundamentals of Game Design*. New Riders Publishing, 2014. ISBN 978-032192967-9. Disponível em: <https://www.amazon.com.br/Fundamentals-Game-Design-Ernest-Adams/dp/0321929675>.

BECK, K.; ANDRES, C. *Extreme Programming Explained: Embrace Change, 2nd Edition (The XP Series)*. Addison-Wesley, 2004. ISBN 978-032127865-4. Disponível em: <https://www.amazon.com/Extreme-Programming-Explained-Embrace-Change/dp-/0321278658>.

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/>.

BORELLA, M. Research note: Source models of network game traffic. v. 23, p. 403–410, 02 2000.

BUCHINGER, D. *Sherlock Dengue 8: The Neighborhood - Um jogo sério colaborativo-cooperativo para combate à dengue*. 2014. Online; accessed 17. Apr. 2018. Disponível em: <http://www.udesc.br/arquivos/cct/id_cpmenu/1024-/diego_buchinger_1_15167055468902_1024.pdf>.

CHADWICK, J.; SNYDER, T.; PANDA, H. *Programming ASP.NET MVC 4: Developing Real-World Web Applications with ASP.NET MVC*. O'Reilly Media, 2012. ISBN 978-144932031-7. Disponível em: <https://www.amazon.com/Programming-ASP-NET-MVC-Developing-Applications/dp/1449320317>.

CLARKE, R. I.; LEE, J. H.; CLARK, N. Why Video Game Genres Fail: A Classificatory Analysis. *SURFACE*, 2015.

DEZA, E. D. M. M. *Encyclopedia of Distances*. Springer, 2009. ISBN 978-364200233-5. Disponível em: <https://www.amazon.com/Encyclopedia-Distances-Michel-Marie-Deza-/dp/3642002331>.

FARBER, J. Network game traffic modelling. In: *Proceedings of the 1st Workshop on Network and System Support for Games*. New York, NY, USA: ACM, 2002. (NetGames '02), p. 53–57. ISBN 1-58113-493-2. Disponível em: <http://doi.acm.org/10.1145-/566500.566508>.

FREEMAN, S.; PRYCE, N. *Growing Object-Oriented Software, Guided by Tests*. Addison-Wesley Professional, 2009. ISBN 978-032150362-6. Disponível em: <https://www.amazon.com.br/Growing-Object-Oriented-Software-Guided-Tests/dp-/0321503627>.

GOLDSMITH, T. *"Cathode-ray tube amusement device"*. 1947. "Online; accessed 15. Apr. 2018". Disponível em: <https://patents.google.com/patent/US2455992>.

GUINNESS. *Greatest aggregate time playing an MMO or MMORPG videogame (all players)*. 2013. [Online; accessed 23. Apr. 2018]. Disponível em: <http://www-.guinnessworldrecords.com/world-records/most-popular-free-mmorpg>.

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

HOWARD, E. et al. Cascading impact of lag on quality of experience in cooperative multiplayer games. In: *2014 13th Annual Workshop on Network and Systems Support for Games*. [S.l.: s.n.], 2014. p. 1–6. ISSN 2156-8138.

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.

IKEM, O. V. How We Solved Authentication and Authorization in Our Microservice Architecture. *Initiate*, Initiate, May 2018. Disponível em: <https://initiate.andela-.com/how-we-solved-authentication-and-authorization-in-our-microservice-architecture-994539d1b6e6>.

JON, A. A. The development of mmorpg culture and the guild. v. 25, p. 97–112, 01 2010.

JONES, M. et al. *JSON Web Token (JWT)*. 2015. [Online; accessed 1. May 2018]. Disponível em: <https://tools.ietf.org/html/rfc7519>.

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, 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.

KLEINA, N. *8 dos maiores mundos virtuais que já conhecemos*. 2018. [Online; accessed 17. Apr. 2018]. Disponível em: <https://www.tecmundo.com.br/internet/129103-habbo-second-life-8-maiores-mundos-virtuais-conhecemos.htm>.

LENGYEL, E. *Mathematics for 3D Game Programming and Computer Graphics, Third Edition*. Cengage Learning PTR, 2011. ISBN 978-143545886-4. Disponível em: <https://www.amazon.com/Mathematics-Programming-Computer-Graphics-Third/dp-/1435458869>.

NADAREISHVILI, I. et al. *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media, 2016. ISBN 978-149195625-0. Disponível em: <https://www-.amazon.com/Microservice-Architecture-Aligning-Principles-Practices/dp/1491956259>.

NEWMAN, S. *Building Microservices*. O'Reilly Media, 2015. ISBN 978-149195035-7. Disponível em: <https://www.amazon.com.br/Building-Microservices-Sam-Newman-/dp/1491950358>.

ROLLINGS, A.; ADAMS, E. *Andrew Rollings and Ernest Adams on Game Design*. New Riders, 2003. (NRG Series). ISBN 9781592730018. Disponível em: <https://books.google.com.br/books?id=Qc19ChiOUI4C>.

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, 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, 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>.

SPORTV. *League of Legends ganha torneio de fim de ano organizado pela ABCDE*. 2018. [Online; accessed 17. Apr. 2018]. Disponível em: <https://sportv.globo.com/site-/e-sportv/noticia/league-of-legends-ganha-torneio-de-fim-de-ano-organizado-pela-abcde-.ghtml>.

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/>.

STATISTA. *Global internet gaming traffic 2021 | Statistic*. 2018. [Online; accessed 19. Apr. 2018]. Disponível em: <https://www.statista.com/statistics/267190/traffic-forecast-for-internet-gaming>.

STATISTA. *Global PC/MMO revenue 2015*. 2018. [Online; accessed 1. May 2018]. Disponível em: <https://www.statista.com/statistics/412555/global-pc-mmo-revenues>.

STATISTA. *LoL player share by region 2017*. 2018. Online; accessed 17. Apr. 2018. Disponível em: <https://www.statista.com/statistics/711469/league-of-legends-lol-player-distribution-by-region>.

SUZNJEVIC, M.; MATIJASEVIC, M. Towards reinterpretation of interaction complexity for load prediction in cloud-based mmorpgs. In: *2012 IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE 2012) Proceedings*. [S.l.: s.n.], 2012. p. 148–149.

THOMPSON, G. W. L. *Fundamentals of Network Game Development*. Cengage Learning, 2008. ISBN 978-158450557-0. Disponível em: <https://www.amazon.com-/Fundamentals-Network-Game-Development-Lecky-Thompson/dp/1584505575>.

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, 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>.

YARUSSO, A. *2600 Consoles and Clones*. 2006. Disponível em: <http://www.atariage-.com/2600/archives/consoles.html>.