Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Cloud.Jus: Arquitetura de Nuvem Comunitária para Provisionamento de Infraestrutura como Serviço no Poder Judiciário da União

Klayton Rodrigues de Castro

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo von Paumgartten

Brasília
2019

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Castro, Klayton Rodrigues de
CC355c Cloud.Jus: Arquitetura de Nuvem Comunitária para Provisionamento de Infraestrutura como Serviço no Poder Judiciário da União / Klayton Rodrigues de Castro; orientador Aletéia Patrícia Favacho de Araújo von Paumgartten. -- Brasília, 2019.
99 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2019.

1. Poder Judiciário da União. 2. Nuvem Comunitária. 3. Infraestrutura como Serviço. 4. Interoperabilidade. 5. IaaS. I. von Paumgartten, Aletéia Patrícia Favacho de Araújo, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Cloud.Jus: Arquitetura de Nuvem Comunitária para Provisionamento de Infraestrutura como Serviço no Poder Judiciário da União

Klayton Rodrigues de Castro

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo von Paumgartten (Orientadora)
CIC/UnB

Prof. Dr. Alexandre Solon Nery
FT/UnB

Prof. Dr. Eugene Francis Vinod Rebello
DCC/UFF

Prof.ª Dr.ª Aletéia Patrícia Favacho de Araújo von Paumgartten
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 08 de Julho de 2019

Dedicatória

Dedico este trabalho primeiramente a DEUS, que sempre esteve ao nosso lado, que nos deu a vida, que fez o mundo e tudo o que nele há, pois Nele vivemos, nos movemos e existimos (Atos 17:24-28).

E também à minha amada esposa, Gabriela, pelo apoio e compreensão durante este período empenhado por várias madrugadas, finais de semana comprometidos e, até mesmo, uma lua de mel curta demais.

Agradecimentos

Agradeço aos meus queridos pais, Wandercleiton e Léia, por me ensinarem a persistir sempre na luta, pelo suporte nas inscrições e viagens para apresentação dos trabalhos acadêmicos e por se preocuparem comigo e minha saúde durante estes dois anos tão árduos.

Agradeço também aos meus amigos de STF e UnB, pelo incentivo a ingressar nessa jornada e pela colaboração vital para que este trabalho fosse realizado. Muito obrigado Venicio Glébson, Polyane Wercelens, Gabriel Diógenes, Flávio Henrique, Ítalo Cavalcante, Rogério Moreira e demais colegas da STI e PPCA.

Agradeço à Prof.^{a} Dr.^{a} Maristela Holanda, pelo apoio na disciplina de Banco de Dados e pela oportunidade de participar das pesquisas do grupo de bioinformática e assim realizar minhas primeiras publicações e apresentações em conferências.

E, por fim, agradeço especialmente à Prof^a. Dr^a. Aletéia Patrícia, por ter me dado a honra de contar com seus tão valiosos ensinamentos e orientações, pela enorme paciência nos momentos de dificuldade e, por ter acreditado no potencial deste trabalho.

Resumo

A composição de uma nuvem comunitária por meio da federação de nuvens privadas é uma das alternativas para hospedar aplicações que exigem uma implantação distribuída, atendendo requisitos como confiabilidade, alta disponibilidade, economia em escala e conformidade aos níveis de serviço exigidos. Entretanto, apesar dos benefícios em potencial, ainda se observa uma baixa adesão ao modelo de nuvem no setor governamental brasileiro, possivelmente decorrente da falta de padronização para realizar a integração de sistemas, a interoperabilidade e a portabilidade para utilização de múltiplas infraestruturas. Atualmente não há uma abordagem transparente para migrar do modelo de infraestrutura tradicional para o modelo de nuvem. Nesse cenário, este trabalho propõe uma arquitetura capaz de integrar recursos computacionais existentes em órgãos públicos em uma plataforma de nuvem comunitária, capaz de abstrair a complexidade da infraestrutura subjacente, de modo que os órgãos possam encontrar uma forma de hospedagem em nuvem que ofereça mais rapidamente seus benefícios potenciais. É apresentada uma solução de baixo custo para iniciar a transição ao modelo de nuvem nas organizações, por meio de uma arquitetura que conta com um *middleware* de baixo acoplamento para interagir com diferentes hipervisores, o qual realiza a federação entre os provedores associados para composição da nuvem comunitária, e também com uma interface de gerenciamento para os potenciais usuários finais (desenvolvedores de software e administradores de infraestrutura), que oferecem a funcionalidade de provisionamento de recursos computacionais distribuídos entre os *datacenters* de maneira automatizada. Para demonstrar a viabilidade da proposta, utilizou-se dois tipos de sistemas de virtualização corporativos, sendo executados em três *datacenters* de órgãos do Poder Judiciário da União (PJU).

Palavras-chave: Poder Judiciário da União, Nuvem Comunitária, Infraestrutura como Serviço, Interoperabilidade, IaaS

Abstract

Building community clouds by federating private clouds is one of the lower-cost alternatives for hosting applications that require distributed deployment to meet scale-saving, high availability, reliability, and service level compliance. However, despite their potential benefits, there are many open issues related to a lack of standardization, system integration, interoperability and portability across multiple infrastructures, that contribute to the low adherence by organizations, particularly in the government sector, because they still struggle to adapt their legacy applications to a native cloud architecture in complex environments. Considering that currently there is no transparent approach to migrate from the traditional infrastructure model to a cloud computing model in some governmental agencies, this work proposes an architecture to integrate computational resources for building a community cloud platform that can abstract the complexity of underlying infrastructure. For this, we chose to develop a loosely coupled middleware to interface with different hypervisors, a GUI, and a CLI, that compose a cost-effective solution to start the transition to the cloud model in organizations. We evaluate the architecture on a set of infrastructures in the Courts of the Brazilian Judiciary to show our approach feasibility.

Keywords: Brazilian Judicial Branch of the Union, Community Cloud, Infrastructure as a Service, Interoperability, IaaS

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Boa parte das áreas de Tecnologia da Informação e Comunicação (TIC), nas empresas e também nas instituições governamentais, costuma exercer controles centrados em seus próprios processos de trabalho, resultando em uma cultura orientada a silos tecnológicos[17]. Como consequência, os ambientes tecnológicos se tornaram bastante complexos, exigindo times de administração cada vez mais numerosos e especializados, particularmente, na atribuição de infraestrutura. Nesse cenário, a construção de soluções integradas em função das necessidades de negócio é algo desafiador, sobretudo em virtude da dificuldade em tornar homogêneas plataformas que, de modo geral, foram estabelecidas de maneira disjunta para cada sistema[18].

Além disso, um paradoxo que ameaça a efetividade das organizações. Enquanto os profissionais de infraestrutura envidam esforços para a preservação do controle, estabilidade e segurança ao ambiente, fatores que tendem a reduzir a celeridade para o fornecimento de novas versões de software, os profissionais de desenvolvimento são demandados a apresentar novas funcionalidades para suas aplicações cada vez mais rapidamente[17]. Isso reforça a necessidade de provisionar recursos computacionais de maneira vez mais ágil e flexível, e amplia a relevância de abordagens capazes de lidar com os ambientes de infraestrutura de modo mais eficiente e eficaz.

Em resposta a esta lacuna, posiciona-se o modelo de computação em nuvem (*cloud computing*), que apresenta um crescimento impressionante nos últimos anos e está redefinindo não apenas a indústria de TIC, mas também a TIC corporativa em todos os setores da economia [19][20]. Este paradigma consiste na apresentação de recursos computacionais como um serviço, no qual a infraestrutura e os softwares envolvidos podem ser dinamicamente instanciados com esforço mínimo de gerenciamento, com base em acordos estabelecidos entre o provedor e o cliente [21][22]. Muitos fatores convergentes estão acelerando sua adoção e alavancando este crescimento: melhor conectividade com a Internet, intensa competição entre os provedores, aumento da maturidade das tecnologias ofereci-

das, redução das despesas operacionais (*Operational Expenditures* ou OpEx) e de capital (*Capital Expenditures* ou CapEx), maior conscientização dos usuários acerca dos benefícios e limitações existentes, bem como a facilidade para implementar dinamicamente novos serviços, levando a inovações que, de outro modo, seriam inacessíveis [23].

1.1 Contextualização

O tratamento do tema proposto para este trabalho possui especial relevância para o autor, uma vez que se relaciona com as atividades por ele desempenhadas no Supremo Tribunal Federal (STF), onde atua desde o ano de 2005 e envida esforços no sentido de aprimorar a arquitetura e implementação de serviços de TIC, de modo que sejam adotadas alternativas de custo/benefício otimizado, preservando assim o interesse público. É também o atual Gerente de Plataforma de Infraestrutura Tecnológica do STF e preside o Comitê Gestor de Arquitetura Tecnológica do órgão.

Conforme estabelecido pelo Art. 102 da Constituição Federal (CF/88), cabe ao STF o papel de guardião da Constituição. O Tribunal atua como corte constitucional e corte suprema, ou seja: julga processos em última instância, questões de constitucionalidade na criação ou aplicação de legislação, questões que envolvam potencial ameaça à Constituição Federal, infrações penais comuns das autoridades máximas da República (Foro Privilegiado), processos de extradição (convenção internacional por meio da qual os países acordantes se propõem a entregar indivíduos mutuamente para que o país em questão possa processá-lo e julgá-lo).

Os Ministros do STF são nomeados pelo Presidente da República e possuem mandato até sua retirada ou aposentadoria, devendo ser indicados dentre cidadãos com notável saber jurídico e reputação ilibada após sabatina do Senado Federal. O Ministro Presidente do STF é o quarto na linha sucessória presidencial (após o Vice-Presidente da República e os Presidentes da Câmara e do Senado Federal) e também o preside o Conselho Nacional de Justiça (CNJ). Cabe ao STF indicar 03 (três) de seus 11 (onze) Ministros para compor o Tribunal Superior Eleitoral (TSE).

Assim, é fundamental que o STF, na posição de órgão de cúpula do Poder Judiciário da União (PJU), ofereça uma prestação jurisdicional célere e se estabeleça como referência aos Tribunais de instâncias ordinárias por meio de estratégias de gestão efetivas com o emprego de TIC, atendendo aos princípios da Eficiência, Eficácia e Economicidade na Administração Pública Federal (APF). Para habilitar tal propósito, a Secretaria de TI do STF (STI) tem por finalidade prover o desenvolvimento de softwares, a prospecção e absorção de novas tecnologias, a administração da rede e bancos de dados, o suporte técnico de softwares e equipamentos e o atendimento especializado no âmbito do Tribunal.

Cabe à STI fornecer as ferramentas adequadas à avaliação proativa do cenário de processos em tramitação, a análise de estados de jurisprudências, a identificação de demandas de repercussão geral e se há melhorias que possam ser empregadas para o aprimoramento da efetividade da Corte, de modo que o órgão possa efetivar suas competências por meio das aplicações finalísticas, em especial os sistemas de tratamento processual e de inteligência para seleção e análise de ações mais relevantes para repercussão geral.

1.2 Motivação

A ausência de uma arquitetura tecnológica adequada repercute na degradação do desempenho da área de TIC perante a área de negócio e provoca uma desconfortável situação aos administradores de infraestruturas convencionais, que precisam dar suporte com grande celeridade às crescentes demandas e atualizações requeridas, ao passo que precisam manter a estabilidade do ambiente[24]. É considerável o impacto de arquiteturas inadequadas no que se refere à efetividade da implantação de aplicações finalísticas do STF, que precisam ser direcionadas a uma arquitetura mais moderna, mantendo a transparência e uma orquestração responsiva e otimizada em termos de custos. Este desafio é compartilhado por vários órgãos do PJU e instituições governamentais em geral.

Com intuito de aprimorar a assertividade dos investimentos, evitar aquisições inadequadas e o demasiado custo operacional ao lidar com ambientes cada vez mais complexos, a motivação deste trabalho é racionalizar a utilização das plataformas de infraestrutura de TIC nas organizações do PJU. Diante de tal necessidade, o modelo de arquitetura proposto neste trabalho é baseado em uma plataforma de computação em nuvem comunitária, cuja infraestrutura é compartilhada por instituições parceiras de maneira colaborativa, ou seja, suportando uma comunidade específica com os mesmos interesses[22].

Para aproveitar os benefícios característicos das tecnologias baseadas no modelo de computação em nuvem, é preciso redefinir a infraestrutura das aplicações, de modo que os softwares desenvolvidos pelo órgão, tais como os sistemas de Peticionamento Eletrônico, eJud (Processamento Inicial, Processamento Criminal), eSTF (Autuação de Processos, Digitalizador e Visualizador de Peças, Sistemas de Decisão dos Julgamentos, Repercussão Geral e Publicação) possam usufruir integralmente de uma arquitetura tecnológica baseada em serviços.

Esses sistemas, de arquitetura monolítica, estão sendo substituídos paulatinamente por microsserviços hospedados em *clusters* de contêineres *Docker*, dando origem à Plataforma STF-Digital, que requer um provisionamento de infraestrutura mais responsivo e elástico, ou seja, tanto para aumentar quanto para diminuir recursos tempestivamente. Por exemplo, provisionar uma máquina virtual simples costumava levar alguns dias no

STF, dada a necessidade de interação manual de várias unidades, e à falta de recursos para oferecer um provisionamento holístico às camadas de infraestrutura (serviços de rede, tais como resolução de nomes, conectividade, endereçamento e regras de segurança, sistemas de virtualização, armazenamento de dados e monitoramento).

Para atender aos requisitos da Plataforma STF-Digital, este provisionamento precisa ser automatizado, instantâneo e deve ser delegado aos administradores de infraestrutura e desenvolvedores de software de forma simples, por meio de uma interface web/API. Além disso, as aplicações finalísticas e serviços críticos precisam contar com uma alternativa de independência de localidade para hospedagem dos recursos computacionais, de modo os sistemas possam manter funcionamento contínuo mesmo em situações de interrupção do funcionamento do *datacenter*, assim como é característico aos provedores de computação em nuvem.

1.3 Justificativa

A adoção da computação em nuvem pressupõe a uso agnóstico e independente dos recursos de infraestrutura. Contudo, um dos maiores desafios ao adotar uma nuvem pública é a complexidade ao lidar com questões de segurança dos dados e gerenciamento de identidade dos usuários, uma vez que a delegação do armazenamento e do processamento de dados não alivia a organização de suas obrigações legais e regulatórias em torno destes. Os pontos críticos são, dentre outros, confiar no modelo de segurança do provedor, incapacidade do cliente de responder integralmente em auditorias, responsabilidade indireta do administrador, implementações proprietárias que não podem ser examinadas, e perda de controle físico sobre as instalações [25].

Além disso, em órgãos governamentais de cúpula, como é o caso do Supremo Tribunal Federal (STF), a ameaça de exposição indevida de dados pode prejudicar julgamentos, causar desconforto político e institucional, ou até mesmo impactar a harmonia da República, no caso de eventual vazamento de votos de Ministros incumbidos de casos polêmicos, ou de informações protegidas sob segredo de justiça em inquéritos resultantes de operações como a Lava Jato [26], a maior ação de combate à corrupção na história recente. Apesar da atribuição de mecanismos apropriados de controle de acesso, trata-se de uma questão política e do grau de aceitação de risco ao hospedar estes dados.

Embora os grandes provedores argumentem estarem melhor preparados para lidar com questões de segurança do que clientes potenciais em instalações *on-premisses*, o armazenamento de dados críticos em nuvem pública encontra certa resistência nestes órgãos pois este tipo de infraestrutura é geralmente oferecida de modo compartilhado ao público em geral [22], podendo encapsular vulnerabilidades em um ambiente externo,

cuja complexidade de controle e rastreamento é maior em relação às instalações locais (*on-premises*).

Deste modo, os Tribunais geralmente optam por hospedar internamente os sistemas de informação e plataformas de armazenamento de dados (*on-premises*). Também se encontra nestes órgãos um consistente parque de infraestrutura base: centros de dados seguros (*datacenters*), redes de comunicação confiáveis, grandes plataformas de armazenamento (*storages*) e máquinas servidoras modernas, dispondo dos principais sistemas de virtualização de mercado corporativo, baseados nos hipervisores VMware[27] e Hyper-V[28].

Considerando que o STF e o Conselho Nacional de Justiça (CNJ) são presididos pelo Ministro Chefe do PJU, e que o Tribunal Superior Eleitoral (TSE) conta com três Ministros que também são membros do STF, além de dois Ministros que são membros do Superior Tribunal de Justiça (STJ), é natural vislumbrar que a adoção de uma nuvem comunitária envolvendo uma parceria entre estes órgãos possa oferecer vantagens em relação às nuvens privadas, cuja infraestrutura é dedicada exclusivamente a determinada organização, sendo também preferível em relação às nuvens públicas, especialmente quando há trânsito de dados extremamente críticos.

Observa-se que os enormes custos iniciais para a implantação destes *datacenters* já foram consumados e, de modo geral, nota-se que há um baixo aproveitamento dos investimentos realizados em contrapartida ao potencial existente. Por outro lado, estas mesmas instituições possuem relação de confiança entre si, boa quantidade de profissionais qualificados atuando, e uma infraestrutura bastante robusta e geograficamente distribuída, porém, operando de modo isolado em cada órgão.

Assim, é comum encontrar infraestruturas preventivamente superdimensionadas em função de arquiteturas deficientes, algo difícil de justificar perante uma área de negócio cujo foco de atuação não é tecnológico, particularmente no segmento governamental. Ademais, a Emenda Constitucional 95/2016 (EC95) instituiu um novo regime fiscal aos órgãos da União, que irá vigorar por vinte anos, prevendo limites individualizados para os gastos públicos e vedando reajustes orçamentários acima da inflação[29]. Portanto, impõe-se aos órgãos maior cautela e responsabilização ao efetuar investimentos na área de TIC, de modo que, vislumbra-se um cenário em que será necessário extrair o máximo de cada infraestrutura por vários anos, até que seja possível realizar novamente grandes aquisições.

Dessa forma, a constituição de nuvens comunitárias, por meio da federação de nuvens privadas, torna-se potencialmente uma solução de atrativo custo-benefício. Esta abordagem é defendida por ampliar a dispersão geográfica dos recursos que cada provedor dispõe e os níveis de utilização que podem ser alcançados.

Por exemplo, os membros da comunidade que estiverem experimentando uma baixa

utilização de seus recursos podem atender cargas de trabalho mais intensas experimentadas por seus parceiros, permitindo que a capacidade ociosa e o Custo Total de Propriedade (*Total Cost of Ownership* ou TCO) sejam reduzidos [30].

Assim, entende-se que se ao menos uma porção da infraestrutura presente em cada Tribunal puder ser integrada em uma arquitetura comum, a cooperação entre as instituições poderá estabelecer uma robusta federação para compartilhamento de recursos de computação e de armazenamento.

1.4 Definição do Problema

Verifica-se que há forte presença de arquiteturas monolíticas nas aplicações do STF e nos demais órgãos do PJU, cuja característica é um alto acoplamento e baixa coesão no fluxo de informações entre Tribunais. Ademais, confiar em um único *datacenter*, mesmo que este venha a ser gerenciado segundo o modelo de nuvem, recai em riscos de impacto à resiliência, continuidade de serviço e capacidade de resposta inadequada aos serviços de prestação jurisdicional que são oferecidos aos usuários de todo o Brasil.

A literatura recomenda a utilização de nuvens de diferentes provedores para garantir a disponibilidade dos serviços de TIC, com intuito de evitar o efeito de bloqueio em um só fornecedor (*vendor lock-in*). Para isso, é desejável a adoção de interfaces padronizadas ou ferramentas de migração de recursos, que ainda se encontram em estágio pouco maduro, isto é, há bastante espaço para o aprimoramento da interoperabilidade em diferentes nuvens, tais como: métodos de acesso aos meios de armazenamento (*storage*), formatos de imagem de máquinas virtuais, desenvolvimento de aplicações e serviços para provisionamento de recursos de hardware e software por meio de *Application Programming Interfaces* (APIs), viabilizando interfaces com acesso à infraestrutura de nuvem [31][32].

Assim, embora a popularização do modelo tenha sido iniciada há cerca de uma década, as dificuldades inerentes aos problemas de *vendor lock-in* e suporte insatisfatório a múltiplas nuvens permanecem, impedindo que os usuários aproveitem ao máximo os recursos disponíveis [33]. Segundo Mezgár e Rauschecker [25], existem diferentes formas de abordar os problemas de interoperabilidade:

- Abordagem Integrada: quando um formato comum existe para todos os serviços utilizados na nuvem. Este formato deve ser acordado por todas as partes para construir e manter os sistemas hospedados.
- Abordagem Unificada: quando um formato comum existe em um nível mais alto de abstração e há equivalência semântica suficiente para permitir o mapeamento entre os serviços utilizados.

- Abordagem Federada: quando não existe ou não é possível empregar um formato comum e os parceiros precisam resolver as questões de interoperabilidade compartilhando uma ontologia capaz de mapear conceitualmente os serviços com uma semântica adequada.

Para superar estes obstáculos, os autores geralmente aconselham a virtualização de todas as dependências de serviços que os aplicativos necessitam, bem como o desenvolvimento de *middlewares* de nuvem [30][34][35], ou a utilização de interfaces existentes no mercado, por meio de ferramentas denominadas *Cloud Management Plataforms* (CMPs), especializadas no gerenciamento e a orquestração de nuvem [12] [33][10][36].

Contudo, estas CMPs prometem capacidades e funcionalidades bastante próximas, tornando desafiadora a seleção da ferramenta mais adequada a cada organização, especialmente em um cenário que carece de adaptação de aplicações a uma arquitetura baseada em nuvem. Também há o problema de carregarem uma complexidade adicional de configuração, não oferecendo instantaneamente tantos recursos quanto desejado e necessário [37].

Assim, comumente, há necessidade de empregar esforços exaustivos para promover uma implantação satisfatória, além de exigir uma mudança de cultura dos administradores acerca da forma de gerenciar os recursos, adaptando a infraestrutura existente à ferramenta escolhida e não o contrário. Esta disruptura nem sempre é possível, rápida ou desejável.

Também estão disponíveis algumas bibliotecas de programação abertas, tais como libvirt [38], jclouds[39] e libcloud[40], capazes de trabalhar com alguns dos principais provedores de virtualização e ferramentas de orquestração [37][41]. Estas bibliotecas requerem a integração com interfaces pré-existentes para se tornarem efetivamente produtivas e, por vezes, ocasionam a depedência de uma comunidade de terceiros para evolução tecnológica, algo que nem sempre é desejável em órgãos públicos. Há ainda algumas iniciativas visando a compatibilização de APIs de diferentes fornecedores, a exemplo das especificações *Open Cloud Computing Interface* (OCCI) e *Cloud Infrastructure Management Interface* (CIMI), que evoluem de maneira um tanto tímida, razão pela qual os provedores ainda demonstram pouco interesse em oferecer pleno suporte a elas [42].

Portanto, apesar da computação em nuvem ter se tornado uma realidade para muitas organizações, ainda são grandes as dificuldades decorrentes da falta de padronização das ofertas dos provedores e das indefinições acerca da interoperabilidade, integração e portabilidade de sistemas entre múltiplas nuvens. Considerando que não há no momento uma abordagem transparente para realizar a transição ao modelo de nuvem no âmbito do PJU e, assim, tratar os problemas apresentados com o gerenciamento da infraestrutura e a heterogeneidade dos sistemas.

Diante do cenário apresentado, este trabalho situa-se no contexto de implantação de infraestrutura como serviço, seguindo uma abordagem de federação de nuvens privadas para estabelecimento de uma nuvem comunitária.

1.5 Questão de Pesquisa

Migrar uma aplicação existente ou construir uma nova, nativamente aderente a uma arquitetura de nuvem, requer bastante esforço de pesquisa e de desenvolvimento para aproveitar ao máximo as tecnologias disponíveis e mitigar os riscos de retrabalho. Assim sendo, surge a questão: é viável definir a arquitetura de uma plataforma de infraestrutura como serviço, pautada por critérios de padronização e modularidade, que seja capaz de abstrair a complexidade da infraestrutura subjacente e potencializar a percepção dos benefícios oferecidos pelo modelo de computação em nuvem comunitária para órgãos governamentais?

1.6 Hipótese

Neste projeto acredita-se que adotar uma abordagem de infraestrutura como serviço (*Infrastructure as a Service* - IaaS) para convergir os recursos existentes em uma arquitetura de nuvem comunitária, capaz de manter transparência e compatibilidade com as aplicações legadas, pode facilitar a transição para o modelo de nuvem. Assim, a hipótese é que outros órgãos do PJU e, eventualmente, outras instituições governamentais, também poderiam se beneficiar ao empregar implementações semelhantes no intuito de promover interesses comuns, tais como: manutenção da segurança na distribuição geográfica de seus dados, maior redundância e tolerância a falhas e catástrofes, implementação de mecanismos para garantir a estratégia de continuidade de serviços. E, além disso, futuramente alcançar uma economia de recursos em larga escala, resguardando o interesse público e garantindo a qualidade na implementação de novos serviços.

1.7 Objetivos

O objetivo geral deste trabalho é a proposição de uma arquitetura para provisionamento de recursos computacionais baseada em nuvem comunitária, capaz de endereçar os problemas elencados no escopo de infraestrutura base para o PJU, otimizando a utilização dos recursos existentes, oferecendo uma alternativa de atraente custo/benefício ao iniciar a transição para o modelo de nuvem nas organizações. Para que o objetivo geral deste trabalho seja alcançado, almejam-se os seguintes objetivos específicos:

- Propor um modelo resiliente, coeso e pouco acoplado para provisionamento de máquinas virtuais, visando a integração de soluções tecnológicas existentes nos Tribunais, capaz de atender as necessidades de hospedagem de recursos computacionais, abstraindo a complexidade ao lidar com elementos de infraestrutura base;
- Reduzir a complexidade das implementações de infraestrutura e facilitar o seu provisionamento de modo distribuído, subsidiando o aprimoramento da eficiência e o uso eficiente de recursos computacionais;
- Disponibilizar uma plataforma de gerenciamento de infraestrutura aderente ao modelo de computação em nuvem, apresentando um baixo risco de impactos negativos, maior visibilidade à entrega de soluções com menor custo global, engenharia adequada, compatibilidade e reuso;
- Demonstrar a viabilidade da arquitetura de nuvem comunitária por meio de uma solução de infraestrutura como serviço compatível com os principais sistemas de virtualização utilizados nos Tribunais;

1.8 Metodologia

Para o desenvolvimento deste trabalho de pesquisa, foram definidas as seguintes etapas:

- Etapa 1 - Levantamento de Estado da Arte: foi conduzida uma revisão narrativa da literatura existente, em um processo empírico e exploratório de busca, análise e descrição do conhecimento envolvendo as abordagens para implementação de uma nuvem computacional, e estabelecimento de uma plataforma de IaaS, com intuito de obter os dados necessários para subsidiar o desenvolvimento da pesquisa e viabilizar os futuros testes;
- Etapa 2 - Definição da Linha Base de Integração: considerando os dados obtidos no Levantamento de Estado da Arte, foi efetuada a modelagem da Arquitetura para Provisionamento da Infraestrutura. O intuito desta estapa foi criar uma linha de base para estabelecer a vinculação entre as funcionalidades existentes e os requisitos elencados;
- Etapa 3 - Desenvolvimento e Implementação Experimental: na terceira etapa serão implementadas as ferramentas necessárias ao desenvolvimento da arquitetura, envolvendo a prototipação e a utilização efetiva da solução proposta em um projeto piloto no STF;

- Etapa 4 - Análise Sistemática e Definição de Métricas: na quarta etapa foram estabelecidas as métricas a fim de suprir as lacunas que persistiam, com intuito de obter repetibilidade e reprodutibilidade, de modo a resumir os dados existentes, refinar hipóteses, selecionar amostras e definir os trabalhos futuros;
- Etapa 5 - Avaliação da Arquitetura: na quinta etapa foram realizados testes com base nos dados obtidos nas etapas anteriores, cujo propósito foi validar e avaliar o desempenho da arquitetura em um estudo de caso, prevendo a operação em múltiplos *datacenters* do PJU;
- Etapa 6 - Defesa da Dissertação e Publicação: a sexta e, última etapa, consistiu em discutir os resultados obtidos, apresentar artigos acadêmicos e elaborar a dissertação do mestrado.

1.9 Estrutura deste Trabalho

Além deste Capítulo introdutório, o trabalho está estruturado em outros 05 (cinco) capítulos, a saber:

- O Capítulo 2 descreve a fundamentação teórica necessária para conceituação do modelo de computação em nuvem e tecnologias envolvidas;
- O Capítulo 3 descreve a fundamentação teórica acerca das plataformas de gerenciamento de nuvem e mecanismos utilizados para lidar com a infraestrutura envolvida;
- O Capítulo 4 apresenta os trabalhos relacionados;
- O Capítulo 5 especifica, em detalhes, a arquitetura proposta;
- O Capítulo 6 apresenta as conclusões e descreve alguns trabalhos futuros.

Capítulo 2

Computação em Nuvem

Neste capítulo serão abordados os principais elementos da computação em nuvem. Na Seção 2.1 será apresentada a definição do paradigma. A seguir, a Seção 2.2 irá descrever suas funcionalidades e características essenciais. A Seção 2.3 discorre sobre as categorias de implantação e os tipos de serviços oferecidos nos ambientes hospedados em nuvem e a Seção 2.4 apresentará os seus principais provedores. Por sua vez, a Seção 2.5 irá descrever a abordagem de nuvens federadas, que surgiram da necessidade de ampliar o poder computacional e a dispersão de serviços em vários ambientes. Por fim, as Seções 2.6 e 2.7 apresentarão, respectivamente, duas tecnologias comuns para suportar as aplicações hospedadas em ambiente de nuvem, os sistemas de virtualização e os sistemas de mensageria com arquitetura distribuída.

2.1 Definição do Paradigma

Apesar de ser uma abordagem relativamente recente, o conceito elementar de computação em nuvem já havia sido enunciado pelo visionário cientista John McCarthy em meados da década de 1960 [43], quando ele afirma que "*chegará o tempo em que os sistemas serão fortemente orientados a serviços e a computação poderá ser definida como um serviço de utilidade pública*".

Assim, cerca de trinta anos depois, com a popularização da Internet e o início da era da computação orientada a serviços, a Amazon desempenha papel fundamental ao desenvolvimento do novo paradigma ao disponibilizar, no ano de 2006, o Amazon Web Services (AWS), cujo intuito foi aproveitar comercialmente a oportunidade de disponibilizar seus recursos computacionais ociosos de maneira abrangente ao público em geral [44]. Rapidamente, outras grandes corporações como Google, IBM, Red Hat e Microsoft iniciaram projetos de pesquisa na área [45] mas, como toda tecnologia emergente, ainda há uma quantidade razoavelmente grande de campos que não foram totalmente explorados.

Dentre os pesquisadores, Staten *et al.* [46] fornecem uma visão dos aspectos tecnológicos mais notáveis: uma infraestrutura resumida e prescrita, totalmente virtualizada, equipada com aplicações e sistemas operacionais independentes, dotada de ferramentas de gerenciamento dinâmico, que abstraem a instalação de hardware e software. Armbrust *et al.* [47] definem nuvem computacional como a união de aplicações oferecidas na forma de um serviço, cujo o hardware e o software utilizado localizam-se em *datacenters* acessíveis pela Internet. Foster *et al.* [48] afirmam que se trata de um paradigma altamente distribuído, orientado à economia em larga escala, cujo poder computacional, armazenamento, plataformas e serviços são virtualizados e gerenciados de forma dinâmica, escalável e sob demanda.

A definição mais popular é estabelecida pelo NIST (*National Institute of Standards and Technology*) [22], o qual define nuvem como sendo um modelo para permitir acesso conveniente e sob demanda a um conjunto compartilhado de recursos de computação configuráveis (redes, servidores, armazenamento, aplicações e serviços), que podem ser rapidamente provisionados e liberados com esforço mínimo de gerenciamento ou interação com o provedor de serviços.

Para Sosinsky [49], a computação em nuvem representa a convergência de vários paradigmas de computação, em especial os sistemas distribuídos (SDs), orientação a serviços, virtualização, alta disponibilidade e tolerância a falha, cuja associação encontra um paralelo com outros serviços sob demanda, nos quais os clientes podem consumir determinados recursos mediante acordo estabelecido com o fornecedor de uma maneira bem mais ampla e efetiva se comparada às abordagens anteriores, que se caracterizavam por uma administração isolada, baseada em silos de infraestrutura, cuja coordenação ágil e automatizada é difícil de ser alcançada.

2.2 Funcionalidades e Características Essenciais

Um ambiente de nuvem é dotado de algumas funcionalidades típicas dos sistemas distribuídos, tal como a escalabilidade [21], que pode ser definida como a capacidade de operar de maneira eficiente, independentemente do seu tamanho, de modo que os softwares utilizados não necessitem de alterações quando a escala aumentar, e que o hardware permita a expansão do sistema, tornando possível a adoção de técnicas como a comunicação assíncrona e o particionamento de componentes [50].

A elasticidade é outra funcionalidade essencial, podendo ser definida como a capacidade, proativa ou reativa, de aumentar ou diminuir os recursos de um serviço em tempo de execução, de maneira que os serviços possam ser alocados e desalocados rapidamente, no decorrer da requisição do usuário [47].

Deste modo, a noção de tempo é crucial, envolvendo tanto o atraso para a percepção da necessidade de reconfiguração, quanto a duração desse procedimento. Assim, para o consumidor, as capacidades disponíveis para provisionamento muitas vezes parecem ser ilimitadas, uma vez que podem ser adquiridas a qualquer momento [51].

Segundo Bermbach [52], existem duas formas básicas para implementar a elasticidade. Na forma vertical, ela se caracteriza ao atuar no aumento ou diminuição de recursos envolvendo um mesmo componente. Na forma horizontal, ela permite a implementação de novos componentes em paralelo ou mesmo a remoção destes de forma rápida e, em alguns casos, automática.

Considerando a combinação das funcionalidades exigidas e o desenvolvimento das ofertas por parte da indústria de nuvem, o NIST [22] apresentou uma proposta consolidada de arquitetura, baseada em cinco características essenciais, três modelos de serviço e quatro tipos de implantação, cujas definições são geralmente as mais difundidas no meio acadêmico e corporativo. Assim, segundo o NIST [22], as principais características de nuvem são:

- Serviço sob Demanda: os clientes poderão consumir recursos de computação unilateralmente, tais como processamento, armazenamento e rede de forma automática, ou seja, sem necessidade de interação humana com cada provedor de serviços (equipe especialista), respeitados os limites previamente acordados;
- Facilidade de Acesso: as capacidades da nuvem estarão disponíveis por meio da rede de dados e poderão ser acessadas através de mecanismos padronizados via interface web ou plataforma dedicada;
- Agrupamento de Recursos: os recursos de computação da nuvem serão agrupados para atender a múltiplos consumidores, cujos recursos físicos e virtuais atribuídos e reatribuídos dinamicamente de acordo com a demanda. Os clientes poderão especificar a localização em um nível de abstração superior (por exemplo: cidade, país ou continente);
- Rápida Elasticidade: as capacidades da nuvem poderão ser provisionadas e liberadas elasticamente, para se escalar rapidamente em conformidade com a demanda, respeitados os limites acordados com o provedor de serviços;
- Medição de Serviço: a plataforma em nuvem terá mecanismos de controle e otimização automatizada do uso de recursos, entregando uma capacidade de medição em nível de abstração apropriado para o tipo de serviço (por exemplo: armazenamento, processamento, largura de banda). O uso de recursos poderá ser monitorado, con-

trolado, auditado e relatado, proporcionando transparência tanto para o provedor como para o consumidor do serviço utilizado.

2.3 Tipos de Implantação e Modelos de Serviço

Para Loeffler [1], os tipos de implantação podem ser definidos como: nuvem comunitária, nuvem pública, nuvem privada e nuvem privada hospedada. Estas categorias dispõem sobre a forma de hospedagem e uso dos ambientes de computação em nuvem e respectivos mecanismos de acesso à infraestrutura. O autor defende que a distinção se dá pela relação entre o nível de acesso e controle da infraestrutura apresentada ao cliente, que pode ser dedicada (exclusiva) ou de uso compartilhado, bem como da localização das instalações, que podem ser privativas ao consumidor (*on-premisses*) ou mantidas por um terceiro (provedor de serviços), conforme apresentado na Figura 2.1:

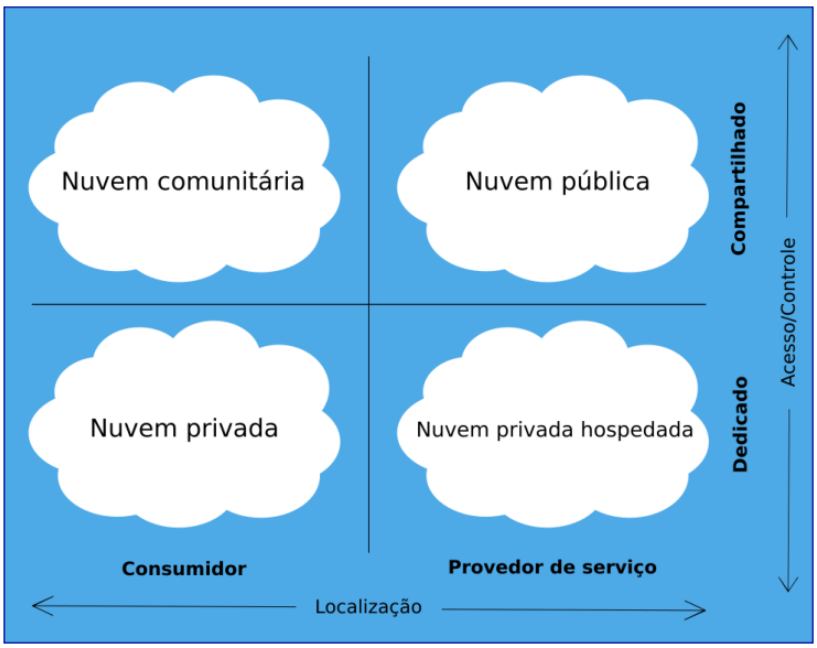

Figura 2.1: Modelos de Implantação de Nuvem, adaptado de Loeffler [1].

Para Buyya [53] e Sotomayor [54], os tipos de implantação podem ser definidos como nuvem pública, nuvem privada, nuvem comunitária ou nuvem híbrida. Estas categorias foram estabelecidas de maneira semelhante pelo NIST [22], e são apresentadas a seguir:

- Nuvem Pública: os recursos são hospedados em uma infraestrutura compartilhada pelo público em geral, disponibilizados aos clientes com mecanismos apropriados de controle de acesso e localizados em instalações operadas por uma organização empresarial, acadêmica, governamental ou alguma combinação entre estas [21]. Segundo

o relatório "Quadrante Mágico 2018" do Instituto Gartner para Infraestrutura de Nuvem como um Serviço, os principais concorrentes deste mercado são a Amazon com a plataforma AWS (Amazon Web Services) [55], Microsoft Azure [56] e GCP (Google Cloud Platform) [57], conforme apresentado na Figura 2.4;

- Nuvem Privada: a infraestrutura de nuvem é utilizada exclusivamente por uma organização, podendo ser gerenciada e operada pela própria instituição, por terceiros ou uma combinação destes. O ambiente está localizado dentro das instalações da empresa (*on-premises*). Quando é utilizada uma infraestrutura de nuvem situada remotamente, esta é chamada de Nuvem Privada Hospedada [1]. As principais soluções que atendem a este segmento podem ser implementadas com uso de plataformas proprietárias [36], tais como o VMware vCloud/NSX [58] ou, ainda, com ferramentas de código aberto, como OpenStack [10], OpenNebula [12], CloudStack [13] e Eucalyptus [14];
- Nuvem Comunitária: a infraestrutura de nuvem é compartilhada por várias organizações de maneira colaborativa, oferecendo suporte a uma comunidade específica que possui os mesmos interesses como, por exemplo, missão, requisitos de segurança e políticas de conformidade. Este modelo pode ser implantado de maneira local ou remota e ser gerenciado ou operado por uma ou mais organizações pertencentes à comunidade, por terceiros ou alguma combinação destes.
- Nuvem Híbrida: existe uma composição de duas ou mais nuvens – que podem ser privadas, comunitárias ou públicas – as quais permanecem como entidades exclusivas, podendo ser agrupadas por meio de tecnologias proprietárias, ou com esforços para o desenvolvimento de softwares e ferramentas de interoperabilidade [59], viabilizando assim a portabilidade de dados e aplicações entre os ambientes envolvidos [33].

Além dos tipos de implantação, existem os modelos de serviço, que definem um padrão arquitetural para o funcionamento das soluções baseadas em computação em nuvem. Estes modelos são definidos como IaaS, PaaS e SaaS [22], e apresentados na Figura 2.2:

- Infraestrutura como Serviço (IaaS - *Infrastructure as a Service*): são oferecidos ao consumidor os recursos, tais como servidores, rede, armazenamento e outros recursos de computação essenciais para construir um ambiente de aplicação sob demanda, que podem incluir sistemas operacionais e aplicativos. A IaaS possui algumas características, tais como uma interface única para administração da infraestrutura, API - *Application Programming Interface* – para interação com dispositivos de computação, armazenamento e comunicação, com suporte à adição de novos componentes

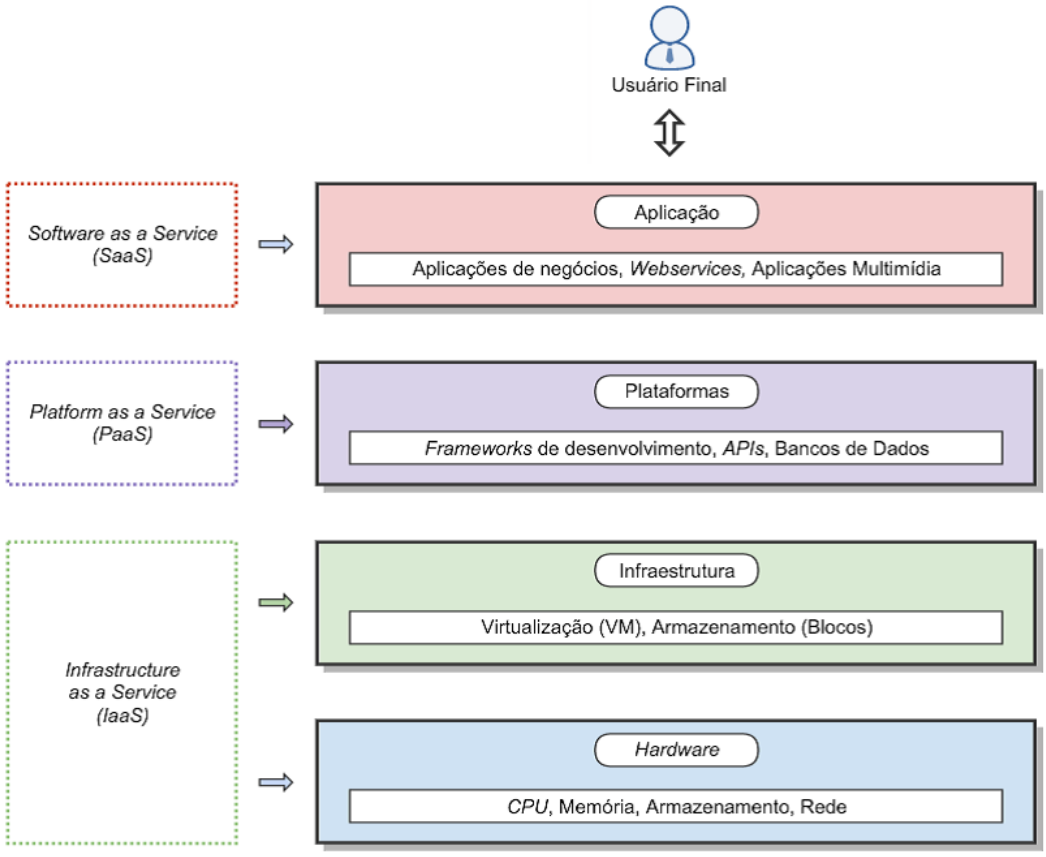

Figura 2.2: Arquitetura de Nuvem segundo Vaquero *et al.* [2].

de forma simples e transparente. Em geral, o usuário não administra ou controla a infraestrutura da nuvem, mas tem controle sobre os sistemas operacionais, recursos de armazenamento e aplicativos implantados. Eventualmente, pode selecionar componentes de rede, tais como balanceadores de carga e *firewalls*;

- Plataforma como Serviço (PaaS - *Platform as a Service*): oferece uma infraestrutura de alto nível de integração para implementar e testar aplicações na nuvem. O usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistemas operacionais ou armazenamento, mas tem controle sobre as aplicações implantadas e, possivelmente, sobre as configurações de aplicações hospedadas nesta infraestrutura. A PaaS fornece um sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de softwares, já que contém as ferramentas de desenvolvimento e colaboração necessárias para tal finalidade. Em geral, os desenvolvedores dispõem de ambientes escaláveis, mas precisam lidar com algumas restrições acerca do tipo de software que pode ser manipulado, tais como, linguagens de programação, bibliotecas, serviços e

ferramentas suportadas pelo provedor;

- Software como Serviço (SaaS - Software as a Service): disponibiliza softwares com propósito específico que estão acessíveis para os usuários através da Internet por meio de uma interface *thin client*, a exemplo de um navegador web. O usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento, ou mesmo as características individuais da aplicação, exceto configurações bastante específicas.

Conforme apresentado na Figura 2.3, pode ser uma observada uma relação entre os modelos segundo uma visão em camadas, na qual o Provedor fornece recursos de IaaS, os quais suportam os recursos das camadas de PaaS que, por sua vez, suportam a camada de SaaS. Enquanto o usuário final consome geralmente recursos no modelo de SaaS, os desenvolvedores de software constroem aplicações que são fornecidas como na forma de SaaS e consomem recursos de PaaS e IaaS.

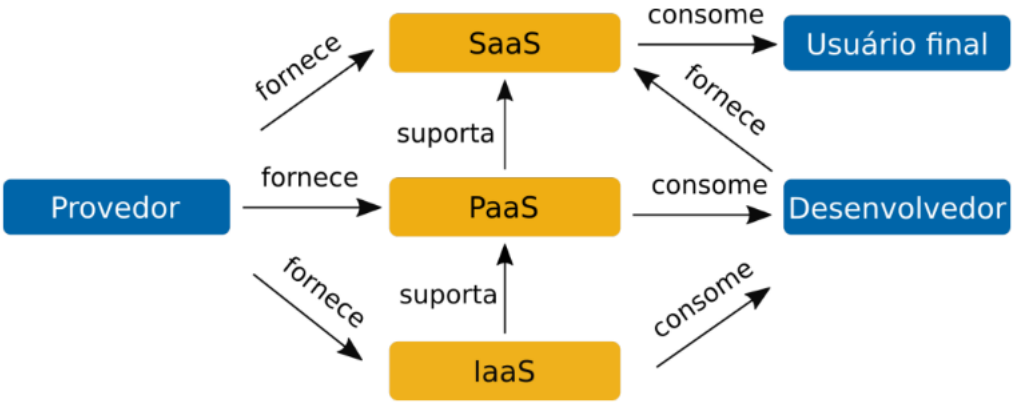

Figura 2.3: Relação entre os modelos serviço, adaptado de Vaquero [2].

2.4 Provedores de Serviço

Existem diversos CSPs operacionalizando o modelo de IaaS em nuvens públicas, a exemplo de Amazon, Google, Microsoft, Rackspace, Digital Ocean, dentre outros. Estes provedores disponibilizam uma enorme capacidade de processamento e de armazenamento a custos acessíveis, uma vez que cada usuário paga por aquilo que usa, enquanto o custo de operação e suporte é dividido entre os vários clientes, descreve Bittman [60]. Segundo o autor, a concorrência entre os CSPs impulsionou um crescimento de 100% no número total de máquinas virtuais (*virtual machines - VMs*) e contêineres instanciados entre 2013 e 2014.

O relatório "Quadrante Mágico 2018"do Instituto Gartner é uma representação usada para relacionar as principais empresas de tecnologia e sua posição de acordo com sua área de atuação [61]. No quadrante, o setor "Líderes"aponta quais são as empresas com o nível mais avançado de desenvolvimento tecnológico. O setor "Desafiantes"apresenta empresas que já têm capacidade plena, mas ainda não conseguiram atingir uma grande parcela do mercado, mas estão crescendo rapidamente. O setor "Visionárias"ressalta as empresas que possuem capacidade de apresentar produtos disruptivos e realizar investimento em pesquisa e desenvolvimento, porém ainda não conseguem executar o que prometem, seja por falta de recursos financeiros, seja por ausência de mão de obra. Por fim, o setor "Concorrentes de Nicho"apresenta empresas que focam em apenas algumas características do mercado e não dispõem de um portfólio de serviços muito abrangente, direcionando suas atividades apenas para um produto específico.

Figura 2.4: Relatório "Quadrante Gartner 2018"para ofertas de IaaS.

Assim sendo, como pode ser observado na Figura 2.4, as principais ofertas de nuvem IaaS estão representadas. O provedor que recebe o maior destaque é a *Amazon Web Services* (AWS), que representa aproximadamente 70% deste mercado e torna-se referência para os demais [62]. A empresa apresenta o *Amazon Elastic Compute Cloud* (EC2)[63] como sua oferta de IaaS, existindo três maneiras de pagar por VMs: (1) instância sob demanda, que fornece uma VM sempre que necessário, podendo ser finalizada após o uso;

(2) instância reservada, que permite ao usuário pagar antecipadamente por uma VM ao longo do tempo; e (3) instância pontual (*spot*), que pode ser adquirida por meio de lances e utilizada temporariamente, apenas enquanto o preço da oferta estiver maior do que outros [62]. Os hipervisores da AWS são baseados nos sistemas de virtualização Xen[64] e KVM[65]. Há ofertas consolidadas de armazenamento de bloco (Amazon EBS) [66], de arquivos/objetos (Amazon S3)[67], e de contêineres *Docker* (Amazon Elastic Container - ECS) [68]. O provedor agrupa seus *datacenters* em regiões, cada uma delas contendo pelo menos duas zonas de disponibilidade [55].

Por outro lado, a Microsoft entrou no mercado de nuvem IaaS com o lançamento do Azure Virtual Machines [69] em meados de 2012, utilizando o hipervisor Hyper-V e instâncias especializadas para operar com muitos recursos, visando atender aplicações como ERP (*Enterprise Resource Planning*). Há armazenamento de bloco e arquivo, juntamente com muitos recursos IaaS e PaaS adicionais, incluindo um serviço de contêiner baseado no *Docker* (Azure Container Service) [56].

A Google fornece a Google Cloud Platform - GCP [57], apresentando o (Google Compute Engine) [70] como oferta de IaaS, combinada com uma oferta de PaaS (Google App Engine) [71] e uma variedade de recursos complementares, incluindo o armazenamento de objetos[72], e um serviço de contêineres *Docker* (Google Kubernetes Engine)[73]. Como diferencial, o Google se posiciona como um provedor aberto, com ênfase na portabilidade centrada em ecossistemas colaborativos. Porém, a exemplo de seus principais rivais (Amazon e Microsoft), também agrega valor por meio da automação de operações em grande escala, não abrindo mão de aprimoramentos proprietários.

2.5 Federação de Nuvens

Alguns autores sustentam que a computação em nuvem, por definição, exige uma infraestrutura mais flexível, interconectada e onipresente, a exemplo de utilidades como eletricidade e telefonia [3][74]. Assim, os pesquisadores atuam no sentido de projetar a computação entre nuvens e obter aprimoramentos imediatos na qualidade de serviço, confiabilidade e economia com integrações. Dessa forma, demonstra-se que as questões relacionadas à interoperabilidade e portabilidade são importantes, tanto para a proteção dos investimentos dos usuários, quanto para a realização da computação como uma utilidade.

Celesti [75] explica que a evolução do mercado da computação em nuvem pode ser dividida em três fases: (1) monolítica, cujos serviços de computação em nuvem são baseados em arquiteturas proprietárias ou oferecidos grandes por provedores; (2) cadeia vertical de fornecimento, na qual alguns provedores tiram proveito de serviços oferecidos por ter-

ceiros, a exemplo das fabricantes de software movendo suas aplicações para o modelo SaaS em nuvens públicas, cujos ambientes ainda são proprietários mas com a presença de iniciativas de integração entre nuvens; e (3) federação horizontal, o estágio atual, no qual pequenos e médios provedores atuam de maneira colaborativa para atingir maior escalabilidade e eficiência no uso de seus recursos.

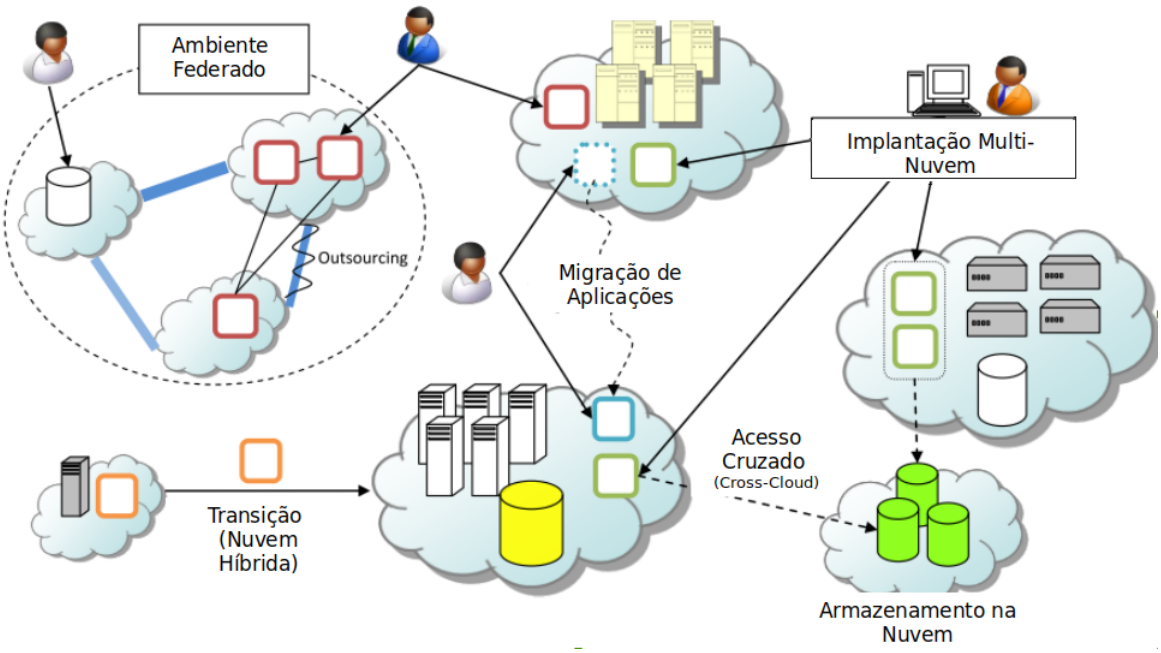

Figura 2.5: Relações de Interoperabilidade entre Nuvens [3].

Conforme apresentado na Figura 2.5, os provedores federados oferecem a possibilidade de migração cruzada de recursos para implementação de redundância e complementação de funcionalidades de maneira cooperativa em uma rede colaborativa, cujo objetivo é promover o compartilhamento de recursos em diferentes camadas, garantindo a distribuição dos serviços com uma maior amplitude e combinação de poder computacional. Deste modo, a federação de nuvens computacionais, também chamada de *inter-cloud* ou *cross-cloud*, é uma área de pesquisa particular em computação em nuvem e pode ser definida como um conjunto de provedores de nuvens computacionais, públicos e privados, conectados por meio da Internet[76]. Grozev e Buyya [74] explicam que os mecanismos de federação *inter-cloud* podem ser gerenciados de modo centralizado, ou seja, quando há um facilitador que assume a responsabilidade por controlar os aspectos de integração, ou ainda do tipo *peer-to-peer*, quando as entidades envolvidas estabelecem colaboração direta umas com as outras negociadas caso a caso.

Toosi *et al.*[3] explicam que as abordagens que podem ser empregadas nestes cenários de federação podem ser baseadas em padronização de interfaces, uso de ferramentas para intermediação (*broker* ou *middleware*) e, também, abordagens híbridas. Os auto-

res defendem que estes cenários podem ser categorizados como centrados no cliente, ou centrados no provedor, em termos de interoperabilidade. Ou seja, as abordagens cujo aspecto central é a vontade de provedor, recaem em nuvens federadas e nuvens híbridas, nas quais os mecanismos de interoperabilidade são mantidos pelos provedores envolvidos. Já as abordagens cujo aspecto central é a vontade do cliente em manejar os recursos entre os provedores, recaem no uso de ferramentas *multi-cloud*, ou seja, aquelas capazes de lidar com múltiplos provedores, e fornecedores de serviços de nuvem agregados por um *broker*.

2.6 Sistemas de Virtualização

Ao investir tempo e recursos para estabelecer operações em uma plataforma de nuvem, uma empresa pode ter dificuldade de alternar o fornecedor, de modo que não é adequado que uma corporação seja bloqueada em um provedor específico (efeito conhecido como *vendor lock-in*), uma vez que esta dependência limitaria o controle sobre os custos e as opções de hospedagem [77]. Isso ocorre pois, embora haja um nível de portabilidade que pode ser alcançado por meio de escolhas estratégicas ao desenvolver uma aplicação, a computação em nuvem ainda está em evolução e experimenta ausência de institucionalização em muitos aspectos [78].

Além disso, se somente fossem utilizadas nuvens públicas, haveria desperdício dos recursos eventualmente existentes em organizações públicas como as do PJU, implicando também em uma possível forte dependência tecnológica ao CSP contratado, tendo em vista que os CSPs costumam impor soluções e interfaces proprietárias para acesso a recursos e serviços. Considerando que administrar um novo ambiente de nuvem é semelhante ao aprendizado de uma nova tecnologia, ou seja, requer tempo e esforço, e que não é incomum a alteração de preços dos serviços e ofertas de nuvem, estas situações elevam as barreiras de transição ao modelo de nuvem em algumas organizações [62].

Assim sendo, a composição de nuvens híbridas ou privadas se apresenta como um modelo popularmente aceito em cenários corporativos [79]. Deste modo, a virtualização de servidores se impõe como elemento chave, ao viabilizar o provisionamento de ambientes flexíveis e robustos para hospedagem de aplicações. Sahoo *et al.* [80] explicam que a tecnologia de virtualização de servidores consiste na criação de instâncias de máquinas virtuais (*virtual machines* ou VMs) que abstraem o hardware subjacente, de modo que cada VM possui um sistema operacional (SO) convidado e as tarefas são gerenciadas por um sistema denominado hipervisor. Isso permite o desacoplamento dos mecanismos de acesso aos recursos do equipamento servidor e aprimora sua eficiência de utilização, pois viabilizar a agregação de sistemas heterogêneos e autônomos em um mesmo ambiente, conforme apresentado na Figura 2.6.

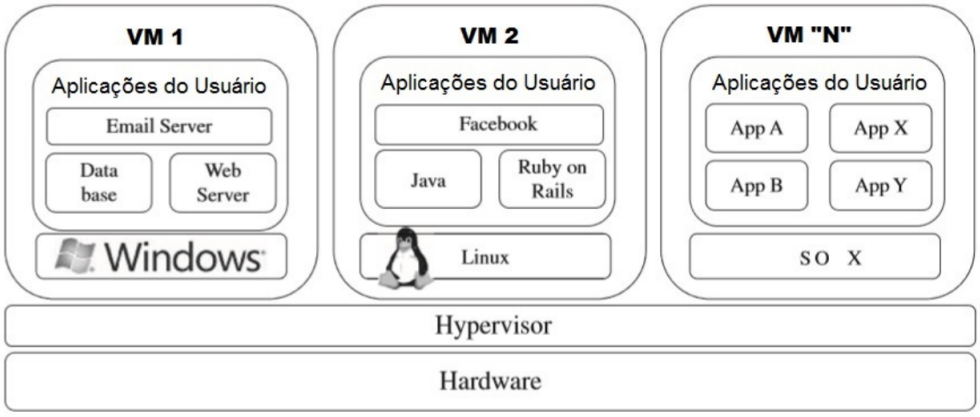

Figura 2.6: Virtualização de Servidores, adaptado de Buyya [4].

O mercado corporativo de virtualização é liderado pelos hipervisores das empresas VMware e Microsoft [60], sendo acompanhadas pela Oracle, conforme "Relatório Quadrante Gartner 2016", apresentado na Figura 2.7.

Segundo Bittman [60], a VMware apresenta uma participação de mercado dominante e capacidades elevadas para o desenvolvimento de produtos e a prestação de suporte técnico. No entanto, o autor explica que a fabricante tem sido a mais afetada pelo rápido crescimento de alternativas hospedadas em nuvens públicas. Além disso, a empresa tem sido desafiada pelo fato de alguns clientes estarem considerando outras alternativas em implantações para atender times de desenvolvimento e suportar filiais corporativas em locais remotos, adotando outros tipos de sistemas de virtualização de menor custo nestes cenários, especialmente o Hyper-V da Microsoft.

Por outro lado, o autor ressalta que, apesar da inferioridade de recursos em relação à líder VMware, as últimas atualizações da Microsoft para o Hyper-V oferecem uma versão mais sólida do produto, mantendo um menor custo que a rival. Isso tem ajudado a fabricante a ampliar sua participação de mercado, tornando-a a segunda opção mais comum em virtualização corporativa. O autor descreve que os esforços da Microsoft em habilitar capacidades de gerenciamento similares às de sua nuvem pública Azure têm atraído empresas interessadas em administrar serviços locais do Hyper-V, da mesma maneira análoga à Azure *Stack* [56]. Esta solução oferece integração a toda a pilha de infraestrutura de maneira associada à estratégia de suporte a contêineres no Windows Server 2016, sendo bastante atrativa em implantações que visam suportar times de desenvolvimento e usuários de filiais em locais remotos.

Bittman [60] também descreve que estas empresas passam cada vez mais a concorrer com a alternativa de virtualização da Oracle, o Oracle Virtual Machine (OVM), cuja estratégia de virtualização se diferencia por otimizar custos ao hospedar os aplicativos portfólio de banco de dados da empresa, que é líder de mercado corporativo neste setor.

Figura 2.7: Relatório "Quadrante Gartner 2016"para ofertas de virtualização de plataformas x86.

Assim, há economia nos custos de licenciamento para os clientes que optam por instalar seus bancos de dados Oracle em ambientes com o OVM. Esta solução é baseada em aprimoramentos proprietários do hipervisor Xen [64] e está impulsionando um crescimento da empresa no setor "Concorrentes de Nicho".

Além da virtualização de servidores, outras técnicas de virtualização têm sido estabelecidas mais recentemente em outros níveis de infraestrutura (sistemas, armazenamento e rede) [80]. Uma das técnicas que se tornou bastante popular é a virtualização baseada em contêineres, a qual oferece mais uma camada de abstração em relação a virtualização de servidores, criando assim um ambiente de execução que encapsula aplicações e bibliotecas necessárias para seu funcionamento. Isso amplia o alcance de características como a independência de localidade e facilita a portabilidade das aplicações entre diferentes infraestruturas [81].

Enquanto um contêiner se concentra em fornecer o ambiente de execução para as aplicações, o provisionamento e gerenciamento de infraestrutura fica à cargo da camada subjacente, operada pelo provedor de computação em nuvem [5]. Assim, desde que o *kernel* (núcleo) do SO utilizado seja compatível e a abordagem de desenvolvimento suporte uma

atuação desacoplada, a virtualização baseada em contêiner permite que uma aplicação seja instanciada diretamente para execução em outro ambiente, suportando inclusive SOs distintos [82], conforme apresentado na Figura 2.8.

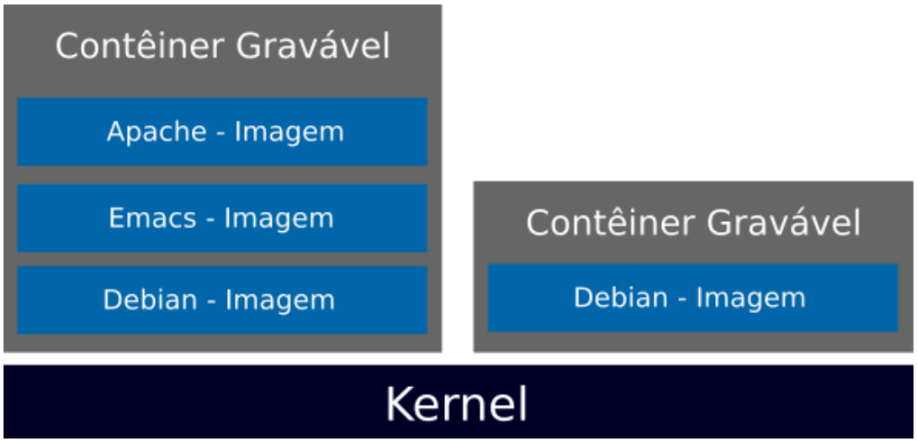

Figura 2.8: Contêineres Formados por Imagens de Softwares Pré-Construídas [5].

Atualmente, o maior viabilizador deste tipo de abordagem é a ferramenta *Docker*, que tornou-se a tecnologia padrão de mercado ao introduzir um ecossistema completo para o gerenciamento de aplicações e implantação de serviços baseados em contêineres [5]. Deste modo, aplicações diversas podem ser migradas para *datacenters* geo-distribuídos, mantendo consistência e independência do sistema hospedeiro, inclusive da infraestrutura de servidores virtuais em diferentes nuvens [82].

A exemplo da virtualização de servidores, a popularização do uso de contêineres deu origem a plataformas especializadas na orquestração e automação deste tipo de abordagem de virtualização. Tais ferramentas são capazes de eliminar grande parte dos processos manuais necessários para implantar e escalar as aplicações em nuvens públicas, privadas ou híbridas, tais como o Openshift [6] e o Kubernetes [73].

Conforme apresentado na Figura 2.9, o orquestrador Kubernetes interage com *pods* (conjunto de contêineres) executados em nodos de um *cluster*. A máquina mestre do Kubernetes aceita os comandos de um administrador (ou equipe de *DevOps*[17]) e retransmite as instruções aos nodos do *cluster*. Esta retransmissão é realizada automaticamente por vários serviços operando em conjunto, de modo a decidir qual nó é o mais adequado para atender a solicitação. Em seguida, são alocados os recursos e atribuídos os *pods* para cumprir a tarefa.

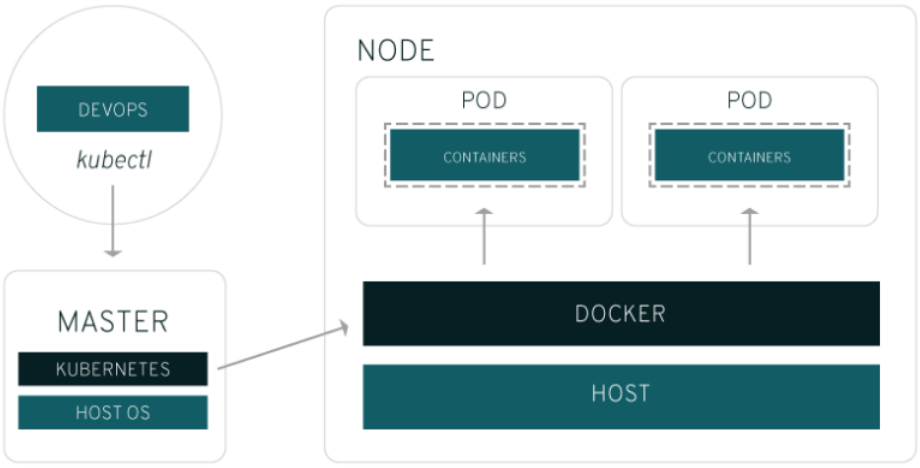

Figura 2.9: Infraestrutura de Orquestração de Contêineres baseada no Kubernetes [6].

2.7 Sistemas de Mensageria Distribuída

O uso de APIs de nuvem torna possível que serviços e aplicações trabalhem de uma forma unificada, integrando ambientes e componentes totalmente distintos, viabilizando um dos principais requisitos para realizar a computação em nuvem: sua capacidade de empregar múltiplos recursos de maneira agnóstica ao ambiente de infraestrutura, mantendo a independência de localidade [49].

A comunicação entre serviços e a criação de mecanismos de troca de dados com maior uniformidade contribui sobremaneira para o desenvolvimento de aplicações eficientes para execução no paradigma de nuvem, permitindo o desacoplamento dos componentes envolvidos, de forma que os clientes de determinado sistema podem se concentrar na comunicação, sem ter que lidar diretamente com os terminais (*end-points*) onde os serviços estão localizados, delegando tal responsabilidade ao sistema de mensageria [7].

Essa abordagem materializa os conceitos de heterogeneidade e transparência em um sistema distribuído [2], no qual os componentes de hardware ou software, interligados em rede, coordenam ações por meio da troca de mensagens, quer estejam em uma única sala ou em continentes separados, mantendo a coesão na experiência do usuário final, o qual deve perceber a atuação conjunta dos múltiplos componentes como se fosse a de um único sistema [83].

Assim, sistemas de missão crítica que não contam com arquitetura totalmente distribuída podem ter seu desempenho, escalabilidade, confiabilidade e resiliência limitados. Embora as APIs REST sejam muito populares atualmente, sua utilização pode ser aprimorada ao adotar sistemas PubSub (Publicação/Subscrição) [7], que já estão bem adaptados à implantação de sistemas distribuídos fracamente acoplados, preparados para serem escaláveis e aderentes ao modelo de nuvem, a qual é facilitada por três aspectos [84]:

- Dissociação de entidades: produtores e consumidores não precisam estar cientes um do outro. O sistema de mensageria determina o início e fim das transações.
- Desacoplamento do tempo: as partes que se comunicam não precisam participando de modo ativo e simultâneo ao longo do tempo.
- Desacoplamento de Sincronização: a interação entre produtor/consumidor e a infraestrutura do sistema não precisa sincronizar *threads* de execução do consumidor, permitindo uso máximo de recursos de processador em produtores e consumidores.

Deste modo, o processamento de *logs* e a troca mensagens emergem como componentes críticos a serem tratados pelo *pipeline* de dados, dando origem a sistemas de mensageria com arquitetura distribuída, a exemplo do RabbitMQ [8], do Kafka [9] e do NATS [85]. As filas de mensagens geradas podem ser categorizadas como intermediadas (broker-based), quando possuem algum tipo de servidor entre os *end-points*, ou não intermediadas (*peer-to-peer*), quando não é requerida uma infraestrutura central atuando na transmissão de mensagens. Existem dois padrões básicos adotados nestes sistemas: (i) solicitação/-resposta (*request/reply*) ou RPC (*Remote Procedure Call*) para serviços e (ii) fluxos de eventos e dados [85].

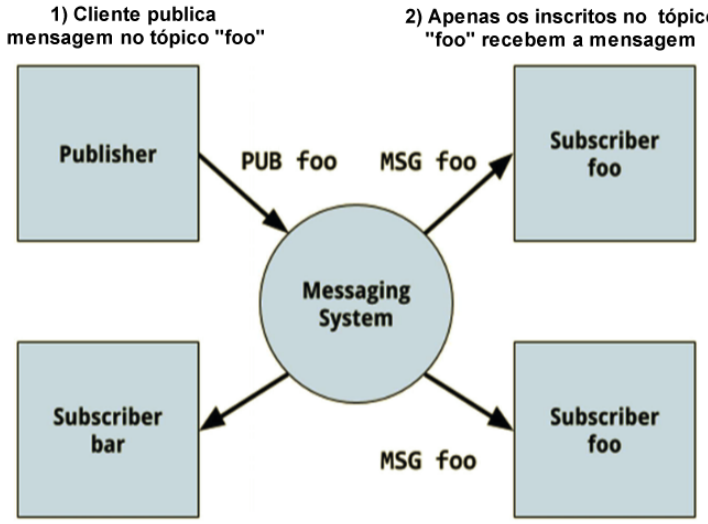

Figura 2.10: Transações de Publicação/Subscrição no Sistema de Mensageria NATS [7].

Assim, a implementação de mecanismos de tolerância a falha e redundância passa a ser responsabilidade da arquitetura do sistema de mensageria empregado, facilitando o desenvolvimento da aplicação pronta pra rodar em ambientes sujeitos à variações de latência de rede em *datacenters* geo-distribuídos. Já os clientes tornam-se apenas consumidores (*subscribers*), registrando seu interesse em um determinado tópico. Sempre que

um produtor (*publisher*) emite uma mensagem sobre esse assunto, o sistema de mensagens a entregará aos consumidores interessados neste tópico, conforme Figura 2.10.

2.7.1 RabbitMQ

O RabbitMQ [8] é uma implementação eficiente e escalonável do *Advanced Message Queuing Protocol* (AMQP), que nasceu da necessidade de interoperar diferentes *middlewares* de mensagens assíncronas. Este protocolo foi projetado para atender requisitos rigorosos de desempenho, escalabilidade e confiabilidade [84].

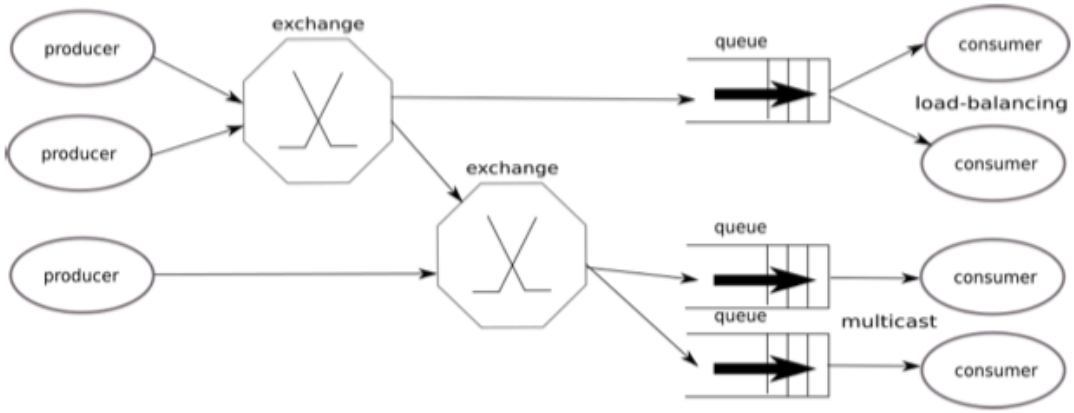

Figura 2.11: Transações de Publicação/Subscrição no RabbitMQ[8].

O RabbitMQ implementa o AMQP, mas possui um mecanismo transacional mais eficiente ao lidar com o controle de fluxo. Sua arquitetura foi estendida para suportar *plug-ins* de outros protocolos de mensageria, tais como o *Streaming Text Oriented Messaging Protocol* (STOMP), que é voltado a transmissão de texto, e o *Message Queuing Telemetry Transpor* (MQTT), cuja comunicação é otimizada para ambientes sujeitos a alta latência de rede, tais como sensores e outros pequenos dispositivos IoT (*Internet of Things*) . Conforme mostrado na Figura 2.11, adota uma abordagem modular, dividindo a intermediação de mensagens entre trocas e filas:

- Uma troca é essencialmente um roteador que aceita mensagens recebidas de aplicações e, com base em um conjunto de regras ou critérios, decide para quais filas encaminhar as mensagens;
- Uma fila de mensagens as armazena e envia para os consumidores de mensagens. A durabilidade do armazenamento depende da configuração desejada pelo administrador do recursos. Geralmente persistem em disco, mas podem ser criadas filas que operam exclusivamente em memória.

2.7.2 Kafka

O Kafka foi desenvolvido no LinkedIn para atuar na integração do *pipeline* de eventos de aplicações da empresa, vindo a substituir um conjunto complexo de sistemas de integração [9]. A solução é projetada para lidar com alto *throughput*, na ordem de bilhões de mensagens, resultando em um sistema PubSub escalável e com recursos de confirmação distribuídos [84].

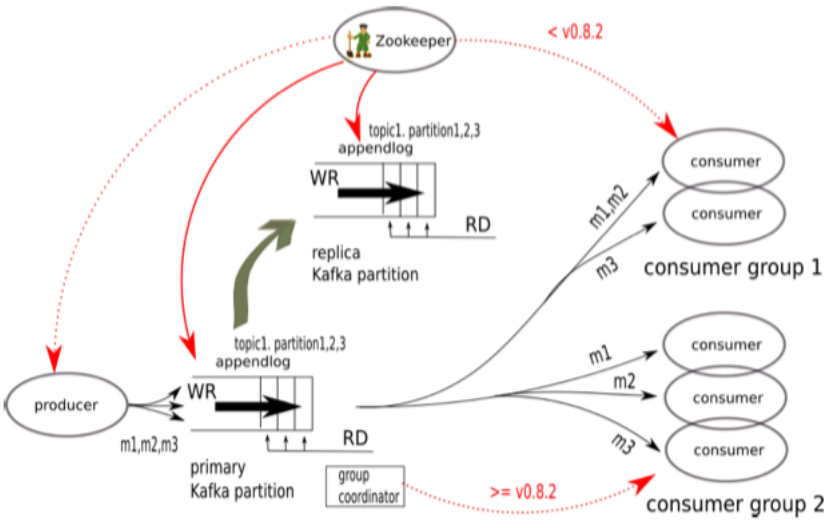

Figura 2.12: Transações de Publicação/Subscrição no Sistema de Mensageria Kafka [9].

A Figura 2.12 mostra a arquitetura de alto nível da plataforma, na qual os produtores enviam mensagens para um tópico do que contém um *feed* de todas as mensagens relacionadas. Cada tópico é distribuído por um *cluster* de *brokers*, no qual cada nodo hospedando zero ou mais partições de cada tópico. Cada partição é um *log* persistido no disco. Todos os tópicos estão disponíveis para leitura por qualquer número de consumidores, e os consumidores adicionais têm uma sobrecarga muito baixa [9].

O Kafka confia no Apache Zookeeper [86] para a implementação do seu plano de controle, adotando técnicas de otimização muito eficazes, tais como o uso processamento em lote em todos os estágios do *pipeline* (produção, intermediação e consumo), estruturas de dados persistentes armazenadas em *cache* de memória RAM, e uso da estratégia de leitura antecipada do sistema operacional, que apresenta uma alta taxa de acertos no *cache* de leitura, especialmente as sequenciais, aliviando assim a necessidade de acesso a disco [84].

2.7.3 NATS

O NATS foi originalmente construído para servir como o barramento de mensagens do Pivotal Cloud Foundry, solução corporativa de orquestração de contêineres da Dell/VMware e com a ascensão de microsserviços e paradigmas nativos da nuvem vem aumentando rapidamente sua popularidade, dando origem a um crescente ecossistema de ferramentas e projetos que o integram como em sua arquitetura, sendo bastante útil para abstrair da aplicação preocupações como [7]:

- Descoberta de serviço;
- Comunicação de baixa latência;
- Balanceamento de carga;
- Notificações e manipulação de eventos;

Ao contrário de outros sistemas de mensageria, não é necessário depender de uma série de componentes que adicionam complexidade ao ambiente, tais como o Zookeeper e a JVM utilizados para instanciar o Kafka e seu algoritmo de consenso do cluster [9]. Uma outra vantagem é que o protocolo NATS é baseado em texto, podendo ser acionado via conexão Telnet por exemplo. Isso é particularmente útil para facilitar a comunicação em cenários que contam com aplicações integradas rodando em sistemas operacionais distintos.

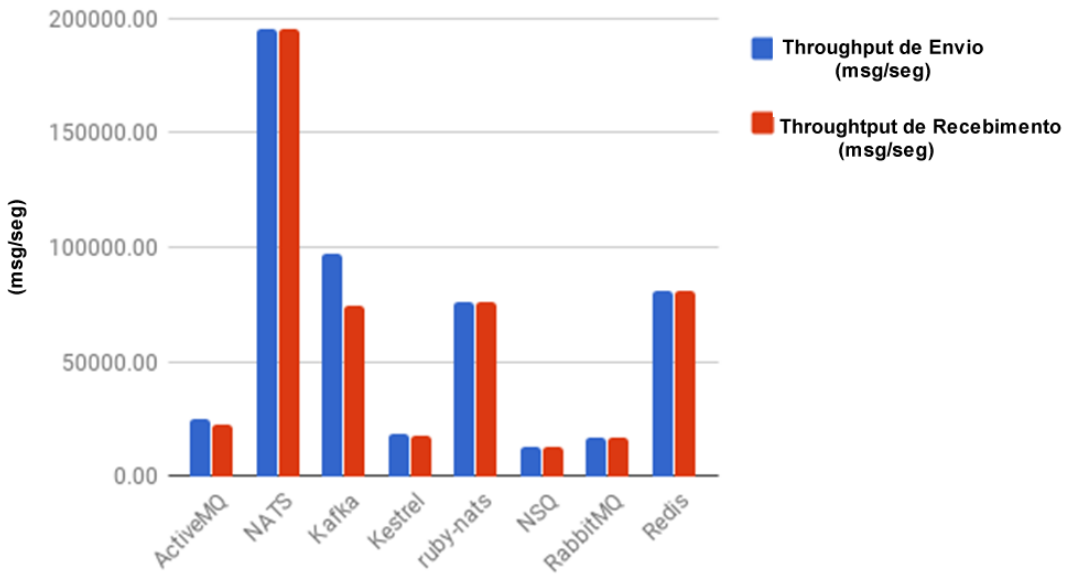

Figura 2.13: *Benchmarking - Throughput* de Sistemas de Mensageria Distribuída[7].

Apesar da arquitetura e implantação mais simples do NATS em relação a outros sistemas de mensageria como o RabbitMQ e o Kafka, os *benchmarks* de solicitação/resposta (*request/reply*) demonstram desempenho superior ao lidar com mensagens menores do que 1MB de carga útil (*payload*)[7][85], conforme apresentado na Figura 2.13.

Capítulo 3

Plataformas de Gerenciamento de Nuvem

Um ambiente de computação em nuvem requer mecanismos eficientes para gerenciar os recursos e orquestrar a infraestrutura oferecida aos clientes. Nesse cenário, encontram-se disponíveis diversas plataformas de gerenciamento de nuvem (CMPs - *Cloud Management Platforms*), que se propõem a gerenciar nuvens privadas, comunitárias, híbridas ou públicas. A Seção 3.1 descreverá a arquitetura das principais plataformas desenvolvidas para esta finalidade e uma comparação entre elas. A Seção 3.2 apresentará as abordagens geralmente utilizadas para padronizar a integração e a interoperabilidade entre ambientes de nuvem distintos e, por fim, a Seção 3.3 discorrerá sobre os principais desafios ao implantar IaaS no modelo de nuvem comunitária, bem como os fatores críticos para o êxito deste projeto.

3.1 Arquitetura das Principais CMPs

A necessidade de obter o compartilhamento de recursos em nuvem e o suporte a *datacenters* elásticos em larga escala, orquestrados segundo o modelo de infraestrutura ágil, proporcionou enormes ganhos de eficiência, em geral decorrentes da arquitetura baseada em serviços [17]. Assim sendo, a comunicação entre as camadas de abstração existentes se tornou cada vez mais eficaz, não exigindo do usuário o conhecimento exato acerca da localidade física e de outros detalhes de configuração [48][49].

Para habilitar a associação destes recursos computacionais, foram desenvolvidas algumas plataformas de gerenciamento de nuvem (CMPs) baseadas em software livre, que se tornaram bastante populares, a exemplo do OpenNebula [54], que se propõe a gerenciar nuvens privadas, híbridas ou públicas, e do OpenStack [87], que proporciona cobertura completa à pilha de recursos das camadas de infraestrutura por meio de uma arquite-

tura modular, baseada em uma série de projetos interoperáveis. Além do OpenStack e do OpenNebula, há outras CMPs que se mostram efetivas neste campo de pesquisa [88], tais como o CloudStack [13] e o Eucalyptus [14], demonstrando que é possível estruturar tais soluções de maneira econômica, tornando-as alternativas viáveis em relação aos provedores comerciais [89][11][90].

Há vários artigos que examinam o assunto e, além de contrapor as alternativas existentes, fornecem recomendações específicas e promovem ajustes de acordo com o uso pretendido [91][92][93]. É o caso do trabalho de Couto *et al.* [94], que propõe a solução de nuvem PID (*Platform for IaaS Distribution*), na qual usuários de instituições de pesquisa podem se beneficiar dos serviços de IaaS com um esforço mínimo de gerenciamento, e sem taxas de assinatura, desde que compartilhem com a comunidade ao menos um servidor capaz de executar máquinas virtuais. Para construir a PID, foi utilizado o OpenStack e realizadas algumas modificações para acomodar a infraestrutura geo-distribuída. É apresentado um protótipo funcional aplicado em três universidades do estado do Rio de Janeiro, e efetuada uma avaliação experimental da infraestrutura de nuvem comunitária. Os resultados demonstram que a latência adicional potencialmente esperada em um ambiente geograficamente disperso teve impacto praticamente insignificante no componente de controle de rede do OpenStack, o Neutron.

Por sua vez, o trabalho de Sharma *et al.* [95] apresenta um modelo de regressão logística para individualizar o provisionamento e a reconfiguração dinâmica de serviços de planos de controle em uma nuvem privada baseada em OpenStack e métodos reativos e proativos para implementar o controle de elasticidade, simulados com a carga de trabalho de uma nuvem pública. Os resultados demonstram que a abordagem é capaz de se adaptar a ambientes de pequeno e grande porte, incluindo a federação de nuvens.

No artigo de Bhardwaj *et al.* [89], os autores descrevem um ambiente de nuvem privada denominado Megh, que utiliza o OpenNebula para provisionamento de infraestrutura em um cenário de teste que envolve computação de alto desempenho (*High Performance Computing* - HPC). De maneira análoga, Razavi *et al.* [96] experimentam cargas de trabalho de HPC interativas e aproveitam a elasticidade da nuvem para provisionar recursos de infraestrutura com base na demanda do usuário, fornecendo dinamicamente máquinas virtuais em tempo de execução. Os resultados demonstram que o ambiente é altamente responsivo e eficiente ao lidar com escalabilidade e elasticidade.

Os autores mencionados demonstraram que as CMPs devem ser projetadas para funcionar com dezenas ou centenas de servidores associados em *cluster* e capazes de acomodar centenas ou milhares de máquinas virtuais (VMs). Assim sendo, uma operação bem-sucedida e eficiente depende de um dimensionamento correto de recursos hospedados, cuja implementação envolve várias opções de configuração, ficando a cargo do adminis-

trador manipular as ferramentas escolhidas conforme as necessidades locais e realizar os ajustes necessários para adequação da capacidade e suporte à carga de trabalho produzida. Assim, ao passo que a complexidade de administração é ampliada, os cuidados com monitoramento e o controle de SLA tornam-se cada vez mais críticos, devido à heterogeneidade, incerteza e dispersão geográfica dos recursos [97].

Nesse sentido, o trabalho de Cunha Rodrigues *et al.* [98] defende que os sistemas de monitoramento devem ser aperfeiçoados, tornando-os capazes de adaptar-se à dinamicidade e complexidade do ambiente de nuvem. Assim, o funcionamento torna-se cada vez mais autônomo e abrangente, observando critérios como facilidade de configuração, capacidade de automatização, precisão para obtenção das métricas em um ambiente distribuído, inclusive a detecção dos impactos da migração de VMs para o SLA estabelecido.

Diante do exposto, conclui-se que o uso de CMPs orientadas ao uso interno de IaaS em diversas instituições de médio e de grande porte começa a ganhar cada vez mais adeptos, visando principalmente a abstração de parte da complexidade da infraestrutura subjacente [21].

Contudo, para se beneficiar da utilização estas ferramentas, é importante trabalhar com uma visão integrada do ecossistema de virtualização, isto é, almeja-se um gerenciamento dos recursos de forma homogênea, de modo que a combinação das diferentes tecnologias implementadas seja apresentada de forma consolidada, independentemente das plataformas de infraestrutura subjacente [54].

Assim, com o objetivo de adequar o planejamento da infraestrutura e a seleção das soluções de software corretas, os usuários e administradores de nuvem precisam avaliar e comparar os recursos oferecidos pelas plataformas concorrentes, cujas características estão descritas em detalhes nas próximas seções.

3.1.1 OpenStack

O OpenStack [10] é uma combinação de softwares desenvolvidos originalmente pela NASA e pelo CSP Rackspace, oferecendo robustez e confiabilidade para gerenciar nuvens públicas e privadas no modelo IaaS. Sua arquitetura modular e altamente configurável foi concebida para se adequar aos recursos de hardware disponíveis, em ambientes grandes ou de menor porte, orientado a cada caso em particular. Conforme apresentado na Figura 3.1, podem ser observados os principais componentes de nuvem controlados segundo uma arquitetura orientada a serviços, cujos mecanismos de comunicação são mantidos por APIs e baseiam-se em uma rede comum compartilhada, suportando servidores físicos, VMs e Contêineres, bem como sua infraestrutura de armazenamento. A visão do usuário é apresentada por meio dos *Dashboards* de Gerenciamento e Monitoração e as aplicações são provisionadas sem que haja dependência de localidade [49].

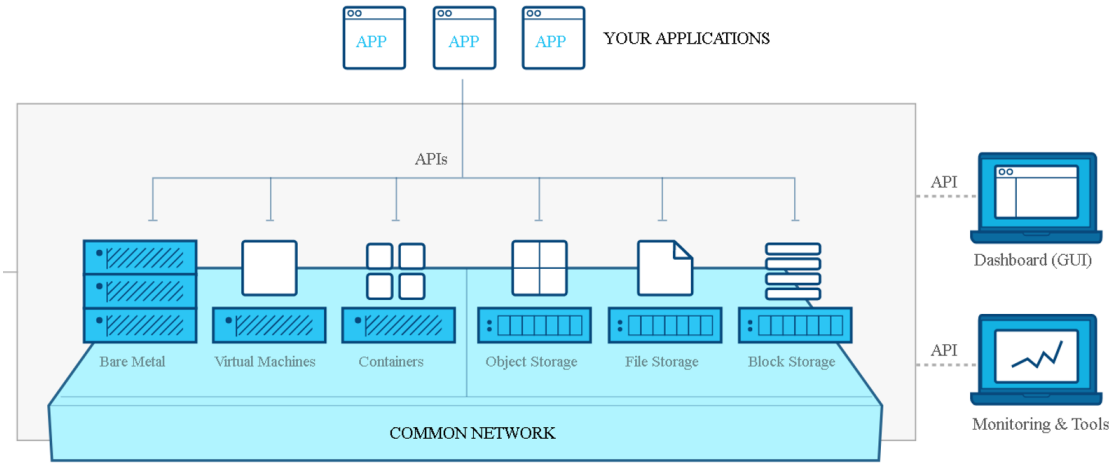

Figura 3.1: Arquitetura da Infraestrutura de Nuvem do OpenStack [10].

Isso permite que as organizações escolham entre uma variedade de serviços complementares para atender diferentes necessidades relacionadas à computação, rede e armazenamento, implementados pelos componentes designados a seguir:

Computação

- Nova: fornece servidores virtuais sob demanda interagindo com hipervisores como KVM [65], Xen[64], além de VMware[27] e Hyper-V [28], mas com algumas limitações, como a exigência de *plugins* proprietários. Disponibiliza serviços para gerenciamento de recursos em nuvem por meio de suas APIs, capazes de orquestrar instâncias em execução, redes e controle de acesso. Seu processo *Scheduler* determina os *hosts* físicos nos quais uma VM pode ser instanciada com filtros em duas etapas. A primeira é baseada na avaliação do *cluster*, selecionando os nodos candidatos com capacidade suficiente de memória RAM e CPU para receber a instância. A segunda é determinada por função de custo, realizada por meio de um sistema gerenciador de filas de mensagens, que utiliza o protocolo AMPQ do RabbitMQ [8];
- Glance: é o serviço de imagens para disponibilização de *templates* de máquinas virtuais por meio de uma API que permite a consulta de metadados de imagem de VM, o catálogo e o gerenciamento de bibliotecas de imagens de servidor.

Rede

- Neutron: fornece conectividade entre dispositivos gerenciados por meio de protocolos dinâmicos (DHCP), estáticos (IPs) ou redes virtuais (VLANs) para aplicação de políticas e topologias avançadas. Devido à sua arquitetura, permite que os usuários

aproveitem os componentes de segurança disponíveis na infraestrutura do *datacenter*, tais como *firewalls*, sistemas de detecção e prevenção de intrusão, balanceadores de carga, dentre outros. Ele também oferece o serviço de virtualização de rede para viabilizar a criação, o compartilhamento e o isolamento da topologia de rede.

Armazenamento

- Cinder: fornece armazenamento de bloco persistente para máquinas virtuais, interagindo principalmente com o Nova. Ele disponibiliza volumes para suas instâncias e permite a manipulação de volumes e instantâneos (*snapshots*) através de sua API;
- Swift: projeto para fornecimento de *object-storage*, oferece armazenamento distribuído e com alta disponibilidade. Além disso, ele pode ser usado pelo componente Cinder para fazer *backup* dos volumes das VMs.

Serviços Compartilhados

- Keystone: responsável pelo gerenciamento de identidades, é o serviço que centraliza a autenticação no OpenStack, disponibilizando as interações via API entre usuários e componentes. Fornece um mecanismo de controle de acesso e *logon* único (*single sign-on*) baseado em *tokens*;
- Horizon: trata-se de uma aplicação web que fornece uma interface de usuário para o gerenciamento da infraestrutura em nuvem, interagindo com todas as outras APIs públicas de serviços, que são apresentadas e controladas neste Painel de Gerenciamento;
- Ceilometer: é o componente de monitoramento, que fornece uma coleção configurável de dados de medição de custos de CPU, memória e rede dos serviços da plataforma, oferecendo um ponto de contato exclusivo para sistemas de tarifação (*billing*).

De acordo com Lima[11], a implantação da arquitetura básica do OpenStack consiste na instalação de três nodos independentes. O primeiro nó é responsável pelo componente de computação e gerencia os hipervisores. O segundo nó é responsável pela componente de rede. O terceiro nó é o controlador, o qual estabelece um serviço de enfileiramento de mensagens. Cada componente trabalha independentemente um do outro, porém os dados são sincronizados entre eles.

Mesmo ao instanciar os serviços em uma só VM, os componentes irão se comportar de modo independente. A comunicação entre eles é intermediada pelo nó controlador, usando APIs RESTful por meio do sistema de mensageria RabbitMQ. Assim, os componentes

Nova, Cinder e Neutron se comunicam via AMQP (*Advanced Message Queue Protocol*), enquanto nova-compute, nova-scheduler e nova-api se comunicam via RPCs, também enfileirados pelo RabbitMQ. Na Figura 3.2 podemos ver o nó controlador interagindo com os demais componentes.

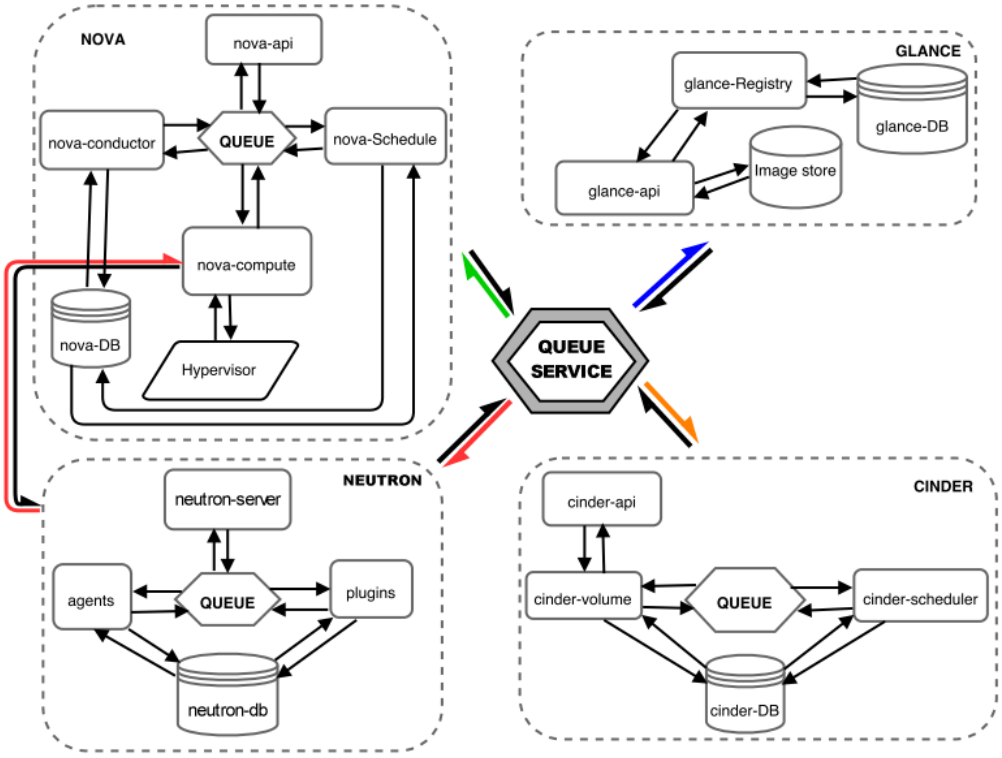

Figura 3.2: Interação dos Componentes do OpenStack com o Sistema de Mensageria RabbitMQ [11].

3.1.2 OpenNebula

A plataforma OpenNebula [59] permite o gerenciamento de infraestrutura de nuvens públicas, privadas e hibridas, com um foco maior nas necessidades do cliente, especialmente em ambientes de nuvem privada [12]. Por ser fracamente acoplada, possui uma instalação relativamente simples, apesar de não possuir tantos recursos em relação ao OpenStack, pois foi desenvolvida para suprir outros tipos de necessidades.

Também é possível empregar ferramentas complementares para efetuar determinadas tarefas, tais como o OpenVSwitch para criar redes, *firewalls* e roteadores virtuais, ou o OpenStack Swift para armazenamento de objetos. O OpenNebula é compatível com os hipervisores KVM, VMware e Xen. Assim sendo, apesar da arquitetura simples, a solução é rica em recursos e pronta para uso corporativo em infraestruturas homogêneas [99].

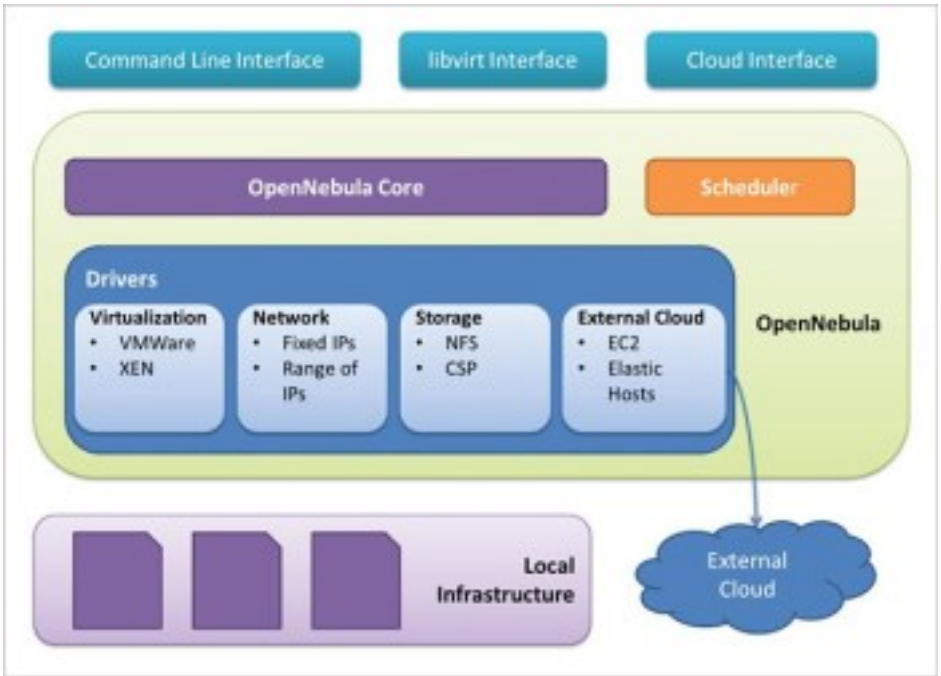

Figura 3.3: Arquitetura da Infraestrutura de Nuvem do OpenNebula [12].

Conforme apresentado na Figura 3.3 o OpenNebula pode executar *plugins*, para realizar operações com imagens de VMs, manipulação do armazenamento, criação de ambientes de rede e interações com os hipervisores. As decisões de posicionamento de VMs são executadas por um componente *scheduler*, que logicamente funciona de modo similar ao OpenStack, ou seja, de acordo com a classificação de recursos e função de custo do escalonador.

A solução é capaz de interagir com nuvens externas e oferece mecanismos para implementar escalabilidade das máquinas virtuais em *clusters* e para elasticidade em nuvem híbrida. Contudo, estes mecanismos são implantados por meio de comandos SSH (*Secure Shell*), ao invés de utilizar um sistema de mensageria. Assim, é provável que surja sobrecarga ao lidar com uma grande quantidade de nodos de computação, o que pode representar gargalos para os usuários da nuvem altamente distribuída [15].

3.1.3 Cloudstack

O CloudStack [13] é uma plataforma aberta para gerenciamento de nuvens IaaS mantido pela empresa Citrix, com suporte para os hipervisores KVM, Xen, VMware e OVM. A solução trabalha com uma arquitetura de implantação que divide a nuvem de acordo com os seguintes conceitos:

- Zona: normalmente é equivalente a um único *datacenter*. Uma zona consiste em um ou mais *pods* e armazenamento secundário.

- *Pod*: geralmente é um *rack* de hardware que inclui um *switch* de camada 2 e um ou mais *clusters*.
- *Cluster*: consiste em um ou mais *hosts* e armazenamento primário.
- *Host*: nó de computação dentro de um *cluster*, onde de fato os serviços de nuvem são executados na forma de máquinas virtuais.
- Armazenamento primário: é associado a um *cluster* e armazena os volumes de disco de todas as VMs em execução nos *hosts* do grupo.
- Armazenamento secundário: está associado a uma zona e armazena modelos, imagens ISO e instantâneos de volume de disco (*snapshots*)

Assim, um *datacenter* pode ter uma ou mais zonas. Um *pod* é tipicamente um *rack* de máquinas, onde *cluster* representa um grupo de máquinas dentro de um *pod*, conforme apresentado na Figura 3.4.

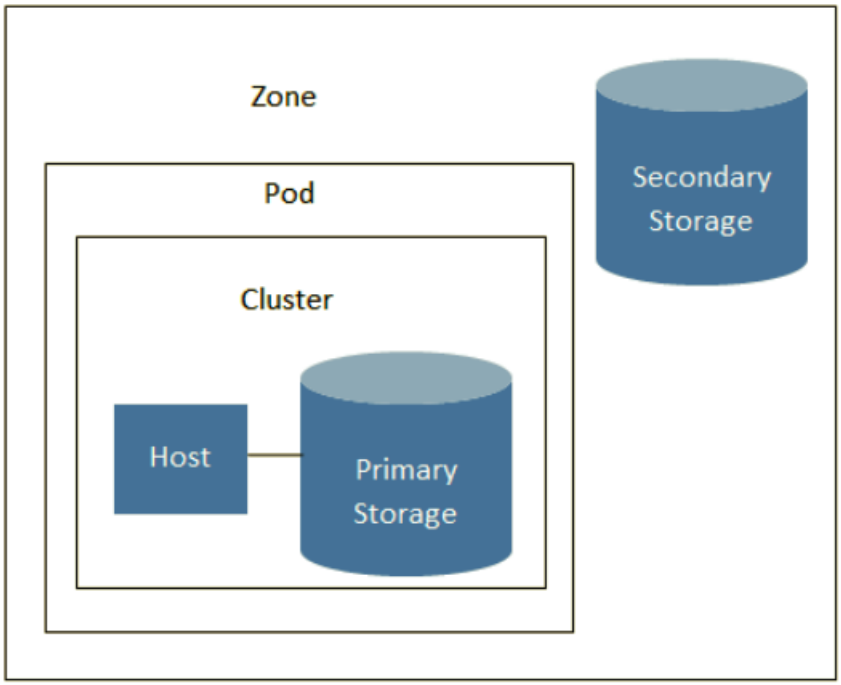

Figura 3.4: Arquitetura da Infraestrutura de Nuvem do CloudStack[13].

No CloudStack, um evento pode ser uma alteração de estado de recursos virtuais ou físicos, uma ação executada por um usuário ou baseada em políticas (alertas). Todos os recursos gerenciados, tais como VMs do usuário, volumes, interfaces de rede, IPs associados, operações com *snapshots* e imagens de VMs, estão associados a uma máquina de estado, e geram eventos como parte da mudança de estado. Para lidar com estes *logs* de eventos síncronos e assíncronos, a plataforma implementa a abstração de barramento

de eventos no Servidor de Gerenciamento. Uma alteração no estado de um recurso resulta em um evento de mudança de estado e o evento é publicado na máquina de estado correspondente no barramento de eventos.

Assim sendo, o barramento de eventos apresentado pelo Servidor de Gerenciamento, o CloudStack permite a implementação de *plugins* para integração com sistemas de mensageria como o RabbitMQ, utilizando um cliente AMQP (*Advanced Message Queuing Protocol*). Além disso, tanto uma implementação em memória quanto uma implementação do Apache Kafka estão disponíveis [13].

3.1.4 Eucalyptus

A plataforma Eucalyptus (*Elastic Utility Computing Architecture*)[14] é capaz de implantar soluções de nuvem privada e híbrida no modelo de IaaS. A solução se notabilizou por ser a primeira a apresentar compatibilidade com a API de nuvem da Amazon, oferecendo suporte à movimentação de instâncias entre um uma nuvem privada e a AWS, estabelecendo assim uma nuvem híbrida. O uso da ferramenta se baseia na configuração dos seguintes componentes:

- Imagem: uma imagem é uma coleção fixa de informações de configuração iniciadas a partir de uma linha de base empacotada e pronta para se tornar uma VM;
- Instância: quando uma imagem é colocada em uso, ela é chamada de instância. A configuração é executada em tempo de execução e o *Cloud Controller* decide onde a imagem será executada, o armazenamento e a rede estarão conectados para atender às necessidades requeridas;
- Endereçamento: as instâncias podem ter endereços IP públicos e privados. Um endereço IP é atribuído a uma instância quando a instância é criada a partir de uma imagem. Para instâncias que exigem um endereço IP persistente, o Eucalyptus fornece endereços elásticos, que podem ser reatribuídos a uma instância em execução;
- Segurança: grupos de segurança TCP/IP compartilham um conjunto comum de regras de *firewall*. As instâncias da camada 2 são isoladas, de maneira que um usuário não pode manipular instâncias vizinhas, pois isso violaria o princípio de separação de inquilinos (*tenants*);
- Rede: existem três modos de rede, (1) Gerenciado, (2) Sistema e (3) Estático. No modo gerenciado, o Eucalyptus gerencia uma rede local de instâncias, incluindo grupos de segurança e endereços IP. No modo Sistema, a ferramenta atribui um endereço MAC e anexa a interface de rede da instância à rede física por meio da *bridge* do controlador de nodos e não oferece endereços IP elásticos, grupos de

segurança ou isolamento de VM. No modo estático, o Eucalyptus atribui endereços IP às instâncias, não oferece IPs elásticos, grupos de segurança ou isolamento de VM;

- Controle de acesso: um usuário do Eucalyptus recebe uma identidade, as quais podem ser agrupadas para centralizar as atribuições de gerenciamento.

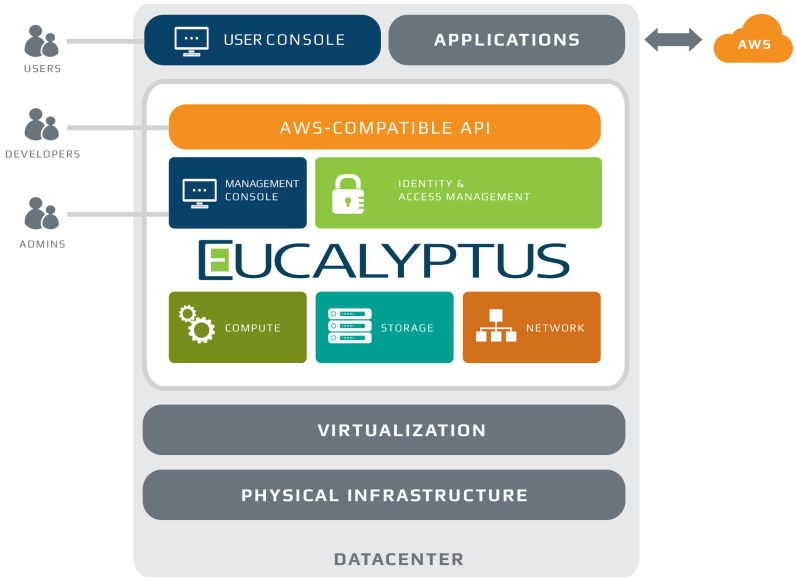

Figura 3.5: Arquitetura da Infraestrutura de Nuvem do Eucalyptus[14].

O principal benefício da Eucalytus é sua praticidade e agilidade. Contudo, apesar ser compatível com o sistema de mensageria da Amazon, o Simple Queue Service (SQS) [100], a ferramenta não oferece suporte a eventos de ciclo de vida das VMs e recursos de automação para elasticidade. Assim, sua escalabilidade é fraca ao manipular fila de mensagens e fluxos de trabalho entre componentes e demonstra limitações para realizar customizações, uma vez que apresenta alguns módulos fechados. A arquitetura de alto nível da plataforma é apresentada na Figura 3.5, cuja implantação é baseada nos componentes descritos a seguir:

- *Cloud Controller* (CLC): é o módulo de gerenciamento central para acesso dos administradores e usuários aos recursos virtualizados;
- *Cluster Controller* (CC): é o módulo executado em uma VM controladora, sendo responsável pelo gerenciamento de rede;
- *Storage Controller* (SC): é o módulo que fornece o armazenamento de rede em nível de bloco;
- *Node Control* (NC): é o módulo usado para monitorar atividades como execução e encerramento de instâncias das VMs.

3.1.5 Comparação entre as Plataformas

Os estudos de Lynn *et al.* [88] e Parmar e Champaneria [36] apresentam uma revisão qualitativa entre as mais proeminentes CMPs de código aberto no âmbito de IaaS (OpenStack, OpenNebula, CloudStack e Eucalyptus). Estes trabalho tem o intuito de apoiar potenciais usuários a compreender suas funcionalidades e auxiliar no processo de tomada de decisão para adoção em caráter inicial, mediante critérios de escalabilidade, compatibilidade com hipervisores, mecanismos de abstração de armazenamento, rede e autenticação, cujas principais distinções estão descritas na Tabela 3.1.

Pode-se observar que as principais distinções se dão no tipo de acoplamento da arquitetura ao ambiente de virtualização, que pode ser fraca ou forte. O acoplamento significa o quanto um componente depende do outro para funcionar, segundo Martin [101]. Quanto maior for esta dependência, mais fortemente acoplados os componentes estarão, tornando mais complexa a realização de mudanças e a evolução de um componente arquitetural. Por sua vez, o baixo acoplamento contribui para independência da solução arquitetural adotada em relação à tecnologia que a suporta, viabilizando uma manutenção mais segura e maior confiabilidade ao realizar mudanças e evoluções nos componentes.

Assim sendo, o OpenStack se destaca pela aceitação corporativa, apesar de ser mais fragmentado e complexo. Mas é robusto e bastante completo, sendo baseado em vários componentes que podem, inclusive, ser utilizados por outras ferramentas como, por exemplo, o armazenamento de objetos via Swift [10]. Por outro lado, o OpenNebula oferece maior simplicidade de instalação e eficiência da interface de gerenciamento disponibilizada aos usuários, segundo Roveda *et al.*[102]. Estas CMPs costumam empregar um sistema de mensageria para apoio à realização das tarefas de gerenciamento, sendo o RabbitMQ [8] o mais comum. O OpenNebula não adota este tipo de ferramenta.

Llorente [59] explica que as soluções apresentadas podem se vincular, em maior ou menor grau, a duas filosofias de *design*, postas de acordo com a proximidade funcional das implementações comerciais consideradas como estado da arte: Virtualização de *Datacenter* (OpenNebula e CloudStack, conceitualmente mais próximas das características do VMware vCloud), e Provisionamento de Infraestrutura (OpenStack e Eucalyptus, conceitualmente mais próximas das características da Amazon AWS), conforme descrito na Tabela 3.2.

Assim, pode-se observar que as ferramentas voltadas à Virtualização do *Datacenter* são mais apropriadas para lidar com ambientes de Nuvem Privada e aplicações definidas em multicamadas (*front-end*, *back-end* e persistência), geralmente dependem de conhecimento mais detalhado acerca da infraestrutura configurada e requerem interfaces mais robustas em funcionalidades. Já as ferramentas voltadas ao Provisionamento de Infraestrutura estão mais associadas aos CSPs de Nuvem Pública e são mais apropriadas para suportar

Tabela 3.1: Distinções entre as Principais CMPs.

	OpenStack	**OpenNebula**	**CloudStack**	**Eucalyptus**
APIs suportadas	AWS/OCCI	AWS/OCCI	AWS	AWS
Hipervisores Recomendados	Xen, KVM, VMware	Xen, KVM, VMware	KVM, Xen, VMware, Oracle VM	Xen, KVM, VMware
Arquitetura	Baseada em Componentes	Fracamente Acoplada	Fortemente Acoplada	Fortemente Acoplada
Nuvem Híbrida	Não	Sim	Sim	Sim
Sistema de Mensageria	RabbitMQ	Não	RabbitMQ e Kafka	AWS SQS

aplicações de arquiteturas modernas *cloud-native*, possibilitando opções de gerenciamento mais simplificado, já que estas aplicações pouco dependem de informações de configuração inerentes aos ativos instanciados.

Tabela 3.2: Diferentes filosofias para implantação de IaaS.

	Virtualização de *Datacenter*	**Provisionamento de Infraestrutura**
Tipo de Aplicação	Multicamadas, definida de modo convencional	Projetada de modo aderente ao paradigma de nuvem
Interfaces	API e Portal de Administração ricos em funcionalidades	API simples e portal de auto-serviço
Tipo de Gerenciamento	Gerenciamento completo do ciclo de vida de recursos virtuais e físicos	Gerenciamento simplificado do ciclo de vida de recursos virtuais e abstração da infraestrutura subjacente
Modelo de Implantação	Privada, em maior parte	Pública, em maior parte
Abordagem de *Design*	*Bottom-up*, definido pelo gerenciamento da complexidade do *datacenter*	*Top-down*, definido pela implementação eficiente de interfaces de nuvem
Requisitos de Negócio	Alta Disponibilidade, Tolerância a falha, replicação e agendamento promovida pela nuvem	*Design* depende da aplicação
Integração ao *Datacenter*	Fácil de adaptar a infraestrutura existente, visando o reaproveitamento dos investimentos	Construído sobre uma infraestrutura de mercado homogênea.

O OpenStack oferece suporte limitado ao VMware e a integração com Hyper-V depende de *plugins* de terceiros. Para funcionar em sua última versão, a integração com VMware traz a exigência do módulo de SDN (*Software-Defined Network*) da fabricante, o NSX [58], que ainda não é tão popular e não está disponível na maioria dos órgãos. A interface GUI Horizon é visualmente pobre. Não é simples implementar os diversos módulos e componentes da solução, requerendo imersão profunda dos administradores, algo que nem sempre será possível no PJU. Mesmo que seja implantada em larga escala, a estratégia tende a incorrer novamente em *vendor lock-in*, algo que se deseja justamente evitar.

Já o OpenNebula, apesar de disponibilizar a visualmente atraente interface SunStone e contar com um vantajoso *workflow* para liberação de recursos, ainda deixa a desejar no quesito integração. É oferecido suporte limitado ao VMware, requerendo a abertura

de portas para estabelecer a abertura de console via VNC, na qual o *token* é o nome da porta e da máquina. Isso representa uma potencial vulnerabilidade quando se trata de um ambiente estendido para comunicação via Internet. Além disso, não oferece suporte para Hyper-V, nem com o uso de *plugins* de terceiros.

Por outro lado, o OpenStack não possui alguns dos recursos que tornam o OpenNebula mais amigável para uso, a exemplo do suporte nativo ao escalonamento automático de VMs (*auto-scaling*) e interface web para disparar a implantação de VMs em lote [15]. Contudo, esta funcionalidade pode ser implementada manualmente no OpenStack, sendo acionada por meio de uma interface de linha de comando (CLI).

O CloudStack, apesar de suportar os hipervisores OVM e VMware, não suporta o Hyper-V. O Eucalyptus tem uma proposta diferente, baseada na possibilidade de portar o ambiente para nuvem pública por meio de sua grande compatibilidade com a API da AWS. Nenhuma das ferramentas oferece de modo simples a integração com ferramentas de monitoramento robustas, tal como o Zabbix [103] ou suporte adequado ao legado, exigindo uma ruptura na forma de administrar o provisionamento de VMs.

É importante destacar que as soluções estudadas empregam diferentes abordagens ao tratar da manipulação de tarefas e eventos. Enquanto o OpenStack adota nativamente o RabbitMQ como sistema de mensageria, o CloudStack pode empregar tanto o RabbitMQ[8] quanto o Kafka[9], por meio de *plugins*. O OpenNebula e o Eucalyptus apresentam abordagens limitadas nesse aspecto, respectivamente baseadas em protocolo SSH (*Secure Sockets Layer*), o qual não gerencia filas, e no AWS SQS [100] (que conta com suporte limitado e não escalável).

Ao estudar o desempenho das abordagens e os efeitos do uso de gerenciadores de filas para lidar com as requisições, o trabalho de Kostantos *et al.*[15] apresenta uma comparação entre OpenStack e OpenNebula, avaliando os tempos de inicialização de 10 VMs, com uma imagem de SO de 10GB, simultaneamente, bem como a sobrecarga de rede introduzida na interação entre os serviços de gerenciamento das plataformas.

É demonstrado um comportamento mais equilibrado na implementação do OpenStack, que demonstra-se menos suscetível a gargalos de desempenho quando exposto a requisições em rajada. Possivelmente esse comportamento ocorre em virtude da plataforma adotar um sistema de mensageria para lidar com tarefas e eventos. Já o OpenNebula, apesar de contar com recursos de API capazes de implementar amigavelmente a elasticidade de VMs, apresentou desempenho insatisfatório lidar com múltiplas requisições, conforme mostrado na Figura 3.6.

Assim, pode ser observado que o OpenStack não é impactado por um alto desvio de tempo para efetivar a implantação simultânea de VMs e que o OpenNebula apresenta um desvio de tempo maior, de comportamento linear, à medida que o número de VMs sendo

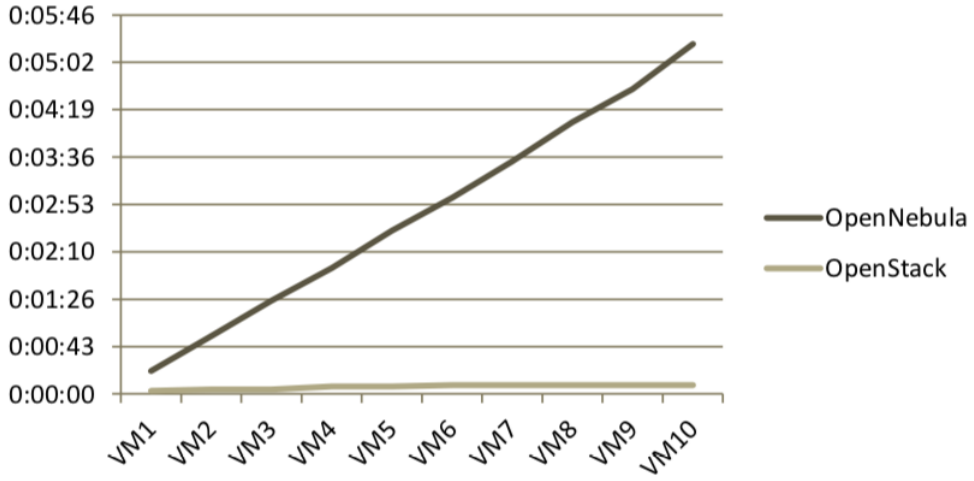

Figura 3.6: Tempos para estabelecer a criação de VMs[15].

implantadas aumenta. Nos experimentos conduzidos, a implantação escalonada de novas VMs geralmente é 75% mais rápida no OpenStack, se comparada ao OpenNebula.

Por outro lado, os autores descrevem que a sobrecarga de rede introduzida no OpenNebula durante os testes é de aproximadamente 20%. Já no OpenStack, a sobrecarga de rede é significativamente maior, atingindo mais de 200%. Contudo, o OpenNebula possui um desempenho superior ao OpenStack para obter o estado de execução das VMs, e poderia ser significativamente beneficiado se um algoritmo ou sistema de mensageria fosse implementado para reduzir o tempo na implantação em rajada.

De maneira análoga, o trabalho de Caron *et al.* [104], aponta que o OpenStack apresenta um melhor tempo de instanciação de VMs em relação ao OpenNebula, também com imagens de SO menores. Foi utilizado um *middleware multi-cloud* para realizar a implantação de VMs com cerca de 3GB em um ambiente de HPC (*High Performance Computing*) para execução de *workflows* de simulação cosmológica. Os componentes de infraestrutura do *middleware* foram estabelecidos em uma máquina física com 4 núcleos de processamento e 16GB de memória RAM. O provisionamento de 10 máquinas alcançou um *throughput* de implantação de 1,8 VMs/min na infraestrutura do OpenNebula e de 6,6 VMs/min no OpenStack.

O trabalho de [105] analisa o comportamento do OpenStack, CloudStack e OpenNebula. Os autores demonstram que as abordagens que fazem uso de sistemas para controlar a fila de mensagens com requisições, OpenStack e CloudStack, são as mais equilibradas ao lidar com requisições em rajada. A implantação de 15 VMs com 1GB de imagem levou em torno de 38 segundos no OpenStack (23,8 VMs/min) e 32 segundos no CloudStack (28,3 VMs/min).

Já o OpenNebula sofreu drasticamente, apresentando uma taxa de implantação de 1,8

VMs/min, repetindo o resultado obtido por Caron *et al.*. Este resultado possivelmente ocorreu em virtude da ferramenta não implementar um sistema robusto para lidar com múltiplas requisições, sendo mais suscetível a problemas de desempenho ao manipular solicitações em rajada, especialmente em ambientes altamente distribuídos, os quais são mais sujeitos à latência de rede.

Assim, conclui-se que essas ferramentas, apesar de flexíveis em nível de customização, envolvem um elevado nível de complexidade e desafios a serem superados, decorrentes da diversidade de funcionalidades e alternativas de implantação, as quais resultam em numerosas relações de interdependência e subordinação de componentes de arquitetura. Isso pode ser um problema no cenário altamente heterogêneo do PJU, especialmente ao lidar com um legado que nem sempre será capaz de suportar uma ruptura imediata com o paradigma tradicional de administração de infraestrutura. Todavia, a compreensão da arquitetura das ferramentas mais proeminentes é essencial para subsidiar a construção da solução proposta neste trabalho.

3.2 Estratégias de Interoperabilidade

O trabalho de Petcu *et al.* [106] define a interoperabilidade como necessária para viabilizar o desenvolvimento de um ecossistema mais amplo de nuvem, capaz de explorar todas as vantagens dos conceitos de elasticidade sob demanda de maneira comum ao mercado. Para os autores, a interoperabilidade possui três aspectos dimensionais: (1) políticas de federação e comunicação entre provedores, (2) ambiente de execução e suporte de migração e (3) projeto de arquitetura com abstração das diferenças programáticas entre provedores. Os requisitos mínimos são distribuídos em subcamadas dentro de cada dimensão, conforme pode ser visto na Figura 3.7.

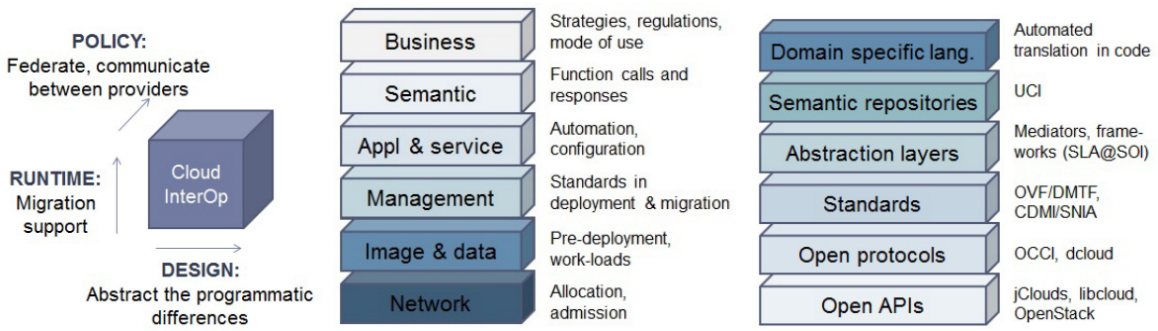

Figura 3.7: Níveis de Interoperabilidade de Nuvem.

Ao provisionar recursos sob demanda por meio de serviços na forma de nuvem computacional, é necessário adotar interfaces de comunicação entre as camadas de abstração

existentes e tratar cada segmento de infraestrutura maneira simples e uniforme [107]. Assim sendo, o número cada vez maior de diferentes serviços e componentes de software empregados em ambientes de nuvem leva ao risco de efeitos inesperados quando tecnologias de diferentes fornecedores são dispostas [108]. Considerando a importância de flexibilizar a operação em nuvem adotando tecnologias de múltiplos fornecedores, surgem alguns trabalhos conceituais, que definem metamodelos e arquiteturas de referência, assim como propostas de APIs interoperáveis e *middlewares* para soluções de código aberto. Embora ainda não tenham sido definidos processos universais para o gerenciamento de plataformas de IaaS, a busca por um padrão uniforme auxilia a proposição mecanismos comuns de interoperabilidade, atendendo à necessidade de combinar recursos e serviços expostos por diferentes fornecedores.

Um dos padrões mais adotados em trabalhos recentes é o OCCI (*Open Cloud Computing Interface*) [32] que define, além de uma API REST, um metamodelo para ambientes de IaaS. A capacidade de simplificar o trabalho de integração de sistemas em nuvem é um dos principais benefícios do OCCI, que oferece um protocolo para abstrair as diferenças entre as implementações de serviços de maneira independente do fornecedor e auxilia o desenvolvimento de extensões para integração com CMPs populares, tais como o OpenStack, o OpenNebula e o CloudStack.

A especificação do OCCI foi projetada para ser modular e customizável. Os recursos virtuais em IaaS são categorizados como Computação, Armazenamento e Rede. Há um modelo central que descreve como o padrão pode ser estendido em eventuais necessidades, aliado a componentes de infraestrutura, que são o escopo deste trabalho, conforme descrito na Figura 3.8.

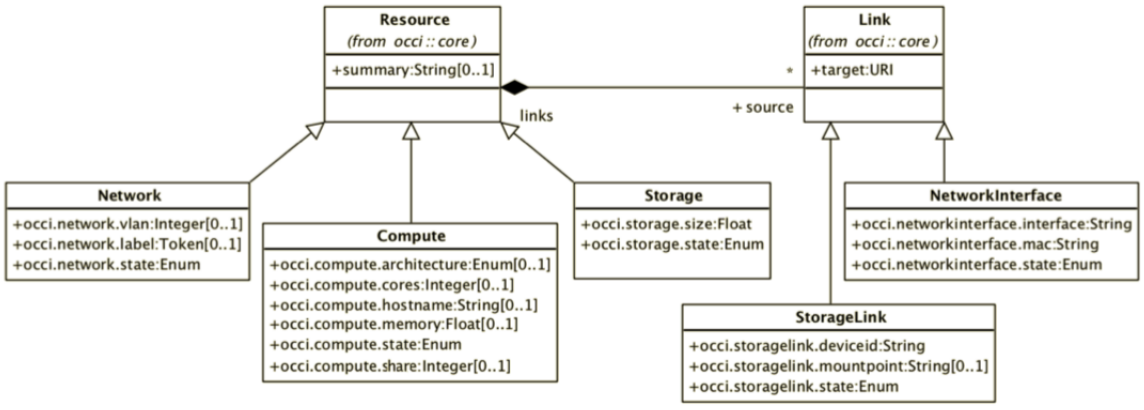

Figura 3.8: Diagrama de Tipos de Infraestrutura do OCCI.

A terceira vertente trata da renderização utilizando HTTP para realizar a interação por meio da API OCCI RESTful. Além disso, detalha os requisitos gerais para a trans-

missão de informações via HTTP, utilizando mecanismos de segurança para autenticação. Há também uma abstração para realizar as associações entre os recursos de Rede ou Armazenamento aos recursos de Computação, bem como as operações de gerenciamento permitidas, tais como criar, modificar e excluir recursos.

3.3 Principais Desafios ao Implementar Nuvem Comunitária no Modelo IaaS

Para as grandes organizações, boa parte das dificuldades de integração reside no software corporativo legado, que adiciona complexidade ao problema e engessa as possibilidades de adaptação ao modelo de nuvem [109]. A imposição de políticas mais assertivas para operação de infraestrutura, desenvolvimento e manutenção de software contribui efetivamente para a construção de aplicações mais previsíveis, reduzindo também a magnitude e impacto negativo na migração do legado para uma infraestrutura aderente ao paradigma de nuvem [108].

Outros autores defendem que o principal desafio não é fazer com que todos os envolvidos adotem uma solução comum, mas que os provedores sejam capazes de oferecer acesso uniforme a serviços essenciais por meio de interfaces padronizadas [110]. Contudo, há resistência na migração de controle do ambiente virtual para CMPs.

Assim sendo, apesar das vantagens que podem ser obtidas ao federar nuvens privadas para constituir uma nuvem comunitária, a implementação não pode ser considerada trivial [78]. Cada nuvem possui características específicas de gerenciamento, configuração, administração, cujas limitações devem ser compreendidas para estabelecer um acordo entre os entes da comunidade. No estudo de caso proposto por este trabalho, os órgãos do PJU apresentam problemas particularmente relacionados aos softwares legados e à variedade de plataformas de virtualização e mecanismos de controle empregados, que ainda não são aderentes ao modelo de nuvem.

Desta maneira, entende-se que os modelos tradicionais apresentados na revisão da literatura, geralmente, consideram recursos estáticos e pouco heterogêneos, viabilizando a adoção de alternativas que exigem menor grau de customização. No cenário do PJU, onde há diferentes hipervisores e administração de infraestrutura altamente dinâmica, é necessário encontrar uma abordagem integrativa que permita a transição ao modelo de nuvem IaaS, primeiramente para cada tecnologia e, em seguida, estabelecendo a comunicação entre os *datacenters*, considerando fatores como o monitoramento, a segurança e o gerenciamento de identidades.

Para endereçar estes requisitos, alguns autores propõem o estabelecimento das seguintes características [4][75][78]:

- Automatização: uma nuvem membro da comunidade, usando mecanismos de descoberta, deve ser capaz de identificar as demais nuvens da federação e quais são os seus recursos, reagindo a mudanças de maneira transparente e automática;
- Previsão de carga de aplicações: o sistema que implementa a federação deve possuir alguma forma de prever as demandas e comportamentos dos serviços oferecidos de maneira eficiente e dinâmica, escalonando a execução das tarefas adequadamente entre os membros;
- Mapeamento de serviços a recursos: os serviços oferecidos pela federação devem ser mapeados aos recursos disponíveis de maneira flexível, atingido o melhor custo-benefício e a garantia da QoS e SLA.
- Modelo de segurança interoperável: a federação deve permitir a integração de diferentes tecnologias de segurança e políticas de gerenciamento de identidades, de modo que os membros da comunidade não necessitem alterar suas políticas internas ao aderir à nuvem comunitária;
- Escalabilidade no monitoramento de componentes: de acordo com a quantidade de participantes, a federação deve ser capaz de lidar com as várias filas de requisições de maneira consistente, sem perda de escalabilidade e desempenho.

Capítulo 4

Trabalhos Relacionados

Para abstrair a complexidade da infraestrutura subjacente no cenário de interoperabilidade de nuvens IaaS, a literatura estabelece como principais enfoques a aplicação padrões e metamodelos capazes de operar em diferentes provedores e camadas de serviço, desenvolvimento de mecanismos de integração que permitam um gerenciamento mais uniforme por meio APIs, extensões (*plugins*) compatíveis com CMPs populares e *middlewares* de provisionamento e federação [34][3][110][106][111][112][113].

Também é comumente apresentada a elaboração de arquiteturas personalizadas, estabelecidas por meio de serviços autônomos, que automatizam as tarefas de gerenciamento e aumentam o valor agregado para os usuários [37][41][42][114][115][16]. Nestes trabalhos, os autores detalham situações que exigem interoperabilidade em nuvem para aplicações científicas/corporativas e conduzem experimentos em ambientes de simulação ou em sistemas empíricos personalizados internamente.

Neste capítulo serão descritas as arquiteturas utilizadas e a comparação entre os principais trabalhos relacionados.

4.1 Arquitetura FogBow

Os trabalhos de Brasileiro *et al.* [30][34][116][117] apresentam o FogBow, um *middleware* utilizado para federar nuvens privadas hospedadas por instituições de pesquisa brasileiras e européias, executando sobre as plataformas de IaaS CloudStack, OpenStack e OpenNebula. A solução também é empregada na Nuvem Acadêmica Federada (NAF), um serviço gerenciado pela Universidade Federal de Campina Grande (UFCG) para federar infraestruturas de instituições que são clientes da Rede Brasileira de Pesquisa e Ensino (RNP). A solução é implantada no topo dos orquestradores de nuvem IaaS de cada membro da federação, apresentando grande flexibilidade ao implementar *plugins* dedicados e

pontos de interação precisos entre o *middleware* de federação e o orquestrador de nuvem subjacente, conforme disposto nas Figuras 4.1 e 4.2.

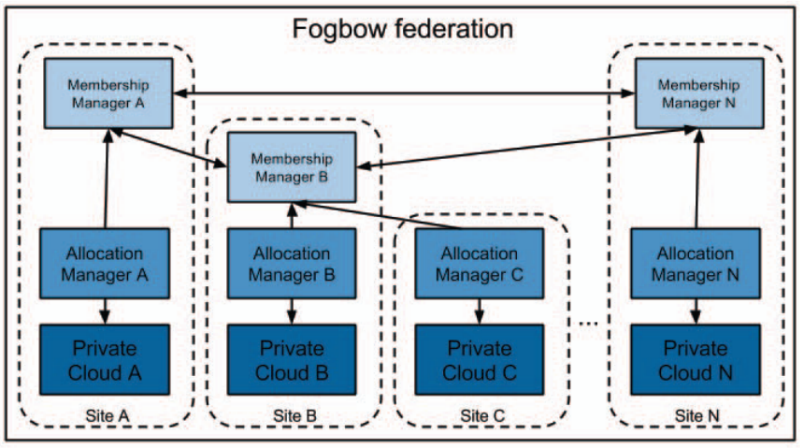

Figura 4.1: Federação de Nuvem com o FogBow.

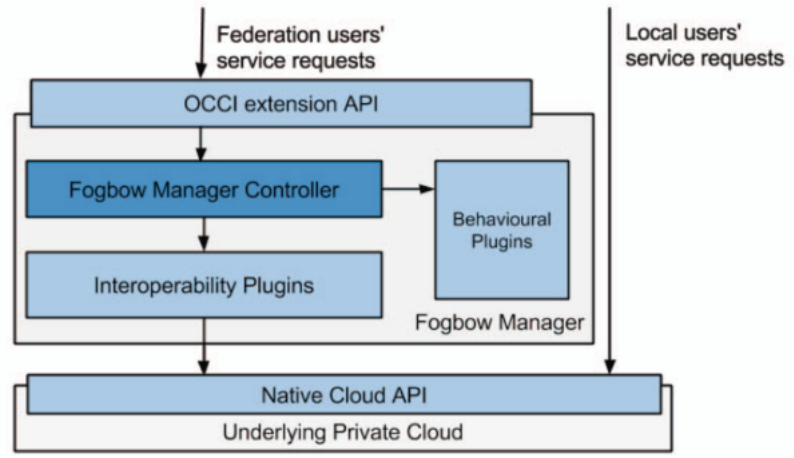

Figura 4.2: Arquitetura do Gerenciador de Alocação de Recursos do Fogbow.

O Fogbow usa tecnologias desenvolvidas para federação via Internet, como o protocolo XMPP[118], flexível para lidar com implementações específicas para autenticação e autorização de usuários e membros. O FogBow fornece uma implementação padrão da interface OCCI, viabilizando a interação dos clientes com gerenciadores compatíveis. A opção pelo XMPP é justificada por este estabelecer um conjunto de padrões que viabilizam a comunicação entre diferentes sistemas, contemplando um indicador de presença que informa aos servidores XMPP a situação (*online/offline/busy*) e a capacidade de envio de mensagens em tempo real com um mecanismo de baixa sobrecarga de rede. Assim os nodos gerenciadores podem obter conhecimento da situação de cada ambiente e realizar o gerenciamento da plataforma, escalonando as tarefas de armazenamento de imagens e serviços de computação com uma abordagem aberta, projetada para ser extensível.

4.2 Arquitetura DIRAC

O trabalho de Casaj *et al.* [90] descreve uma arquitetura de federação denominada DIRAC (*Distributed Infrastructure with Remote Agent Control*), capaz de atuar como *broker* para oferecer IaaS de modo compatível com CMPs como OpenNebula e OpenStack. Conforme descrito na Figura 4.3, o *middleware* do DIRAC constrói uma camada de abstração entre os usuários e os recursos, oferecendo uma interface comum para acionar os vários fornecedores via API, GUI e CLI. O uso de um conector OCCI é implementado em um *middleware*, de maneira que o usuário final pode interagir com recursos alocados em diferentes provedores de forma transparente.

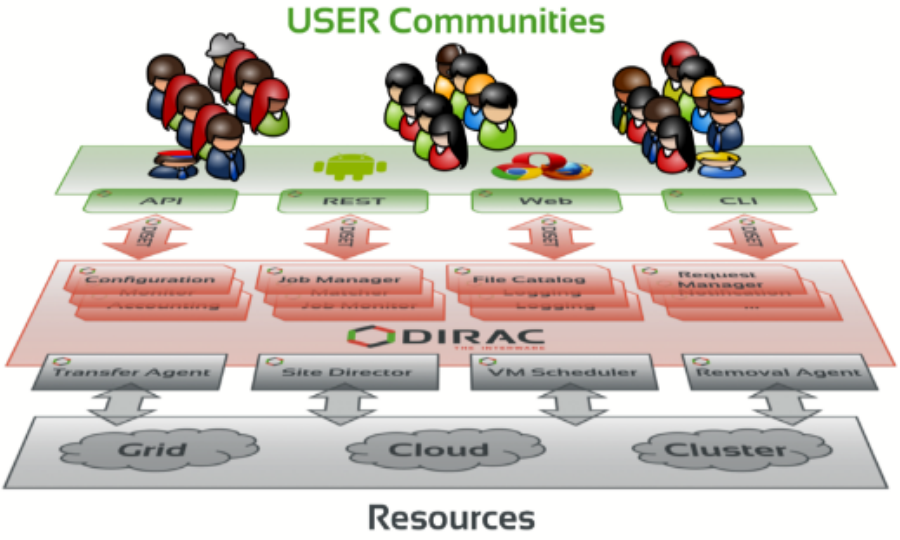

Figura 4.3: Arquitetura do DIRAC.

4.3 Arquitetura Aurora Cloud Manager

Ao enfrentar dificuldades relacionadas às estratégias de alocação de recursos de rede, Schneider *et al.* [119] defendem que necessidades individuais de ambientes e aplicações ainda são limitadas quando se trata de implantar aplicações altamente distribuídas com requisitos rígidos de rede, tais como garantias de baixa latência e QoS para largura de banda, em virtude das CMPs atuais serem projetadas para lidar majoritariamente com recursos de computação e armazenamento.

Em continuidade à proposta de Schneider *et al.*, os trabalhos de Wickboldt *et al.* [37] e Heimovski *et al.* [41] apresentam o Aurora Cloud Manager (Aurora CM), cuja arquitetura possibilita o controle de alguns aspectos da infraestrutura, como o tráfego de fluxo da rede em tempo real, através da programabilidade obtida por meio de interação

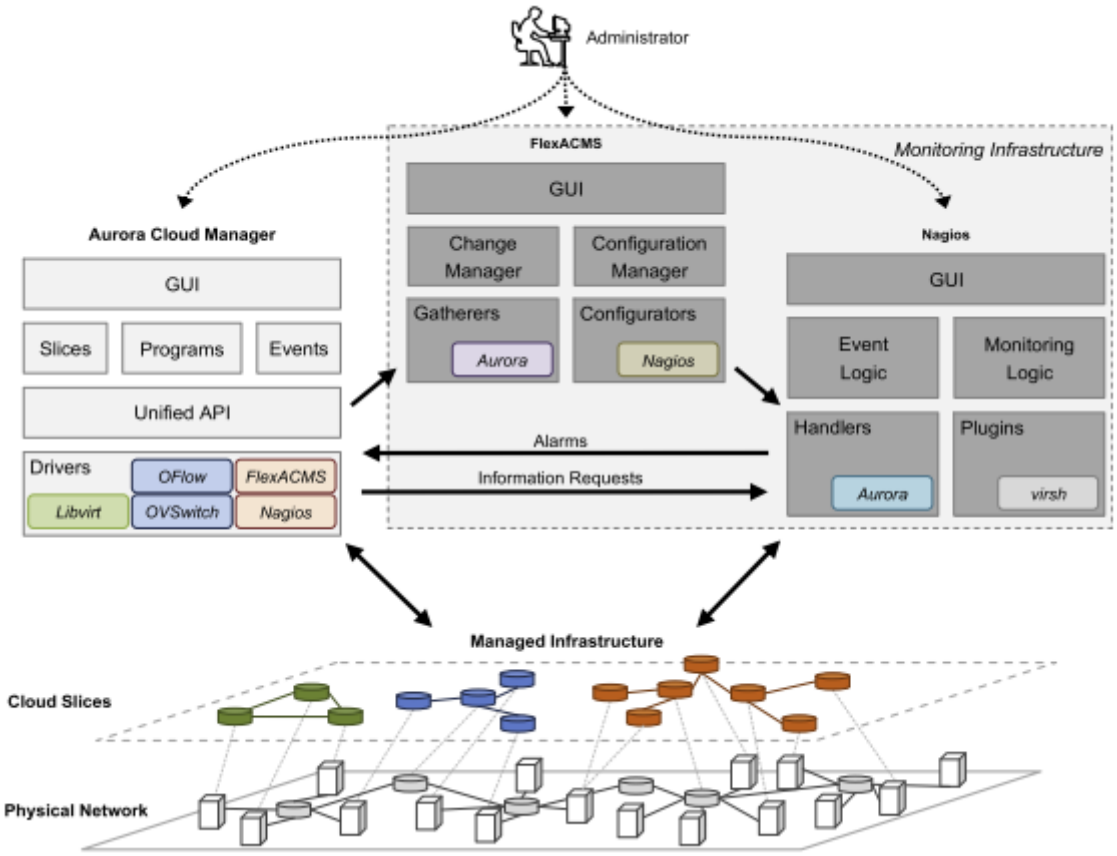

Figura 4.4: Arquitetura do Aurora Cloud Manager.

com a API desenvolvida para tornar o provisionamento e o gerenciamento de recursos de IaaS mais flexível.

Sistemas externos fáceis de usar (Nagios e FlexACMS) foram adotados pela Aurora CM para oferecer o monitoramento da nuvem [98]. Conforme descrito na Figura 4.4, a arquitetura da plataforma oferece GUI, CLI e API que abrangem o gerenciamento e provisionamento de IaaS compatível com federação e estabelecimento de nuvem comunitária. Para tal, a Aurora CM confia na biblioteca libvirt para operar a camada de virtualização.

4.4 Arquitetura CLOUDIATOR

Ao apontar que a abstração de nuvem oferecida por bibliotecas não é suficiente e que, além disso, vários CMPs ainda falham ao fornecer recursos fundamentais, Baur e Domaschka [42] apresentam o CLOUDIATOR, uma abordagem alternativa para tratar problemas existentes na orquestração entre nuvens.

A solução foi projetada para permitir que o usuário defina a configuração da máquina virtual com base em requisitos de negócios, abstraindo a necessidade de obter conheci-

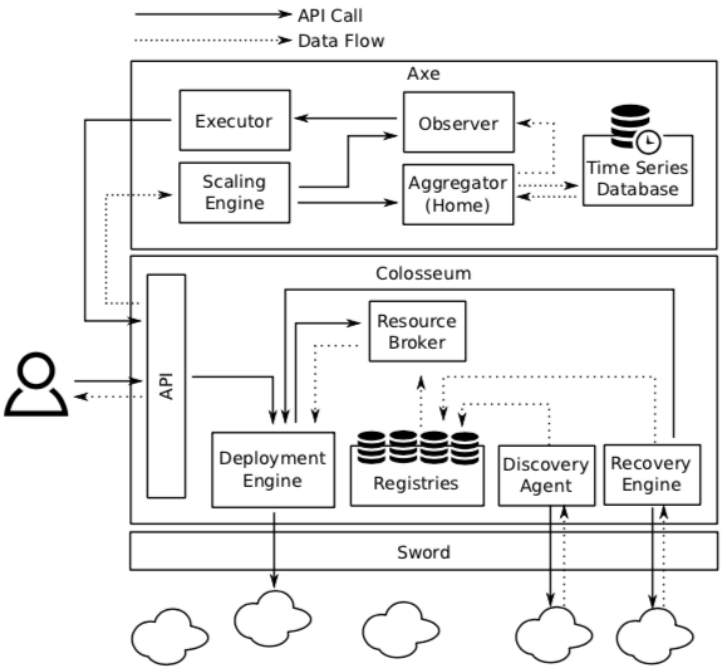

Figura 4.5: Arquitetura do CLOUDIATOR.

mento detalhado sobre os provedores de nuvem empregados, oferecendo ainda um sistema de monitoramento de recursos (AXE), conforme apresentado na Figura 4.5.

4.5 Arquitetura MCOF

A orquestração em ambiente multi-nuvem geo-distribuído encontra desafios ao lidar com plataformas de nuvem e APIs diversas, além de catálogos de serviços distintos. A latência de rede entre *datacenters* e seus diferentes níveis de segurança também dificultam o gerenciamento da orquestração com ferramentas tradicionais como Ansible e Puppet.

Nesse contexto, o trabalho de Lu *et al.* apresenta um modelo de orquestração multi-nuvem global denominado MCOF (Multi-Cloud Orchestration Framework) [16]. O MCOF converte as instruções de orquestração iniciadas por um nó mestre MCOF utilizando padrões pré-estabelecidos, tipicamente escritos em *Shell Script* para ambientes Linux, Powershell para ambientes Windows e API do OpenStack Heat para componentes de infraestrutura distribuídos no hipervisor KVM.

As requisições são repassadas os nodos em execução (workers) dentro de cada *datacenter*, por meio de uma fila de mensagens, de modo que somente serão efetivadas no ambiente provedor de serviços de nuvem correspondente, ou seja, atrás do *firewall*, adaptando-se a ambientes operacionais complexos, conforme apresentado nas Figuras 4.6 e 4.7. Como

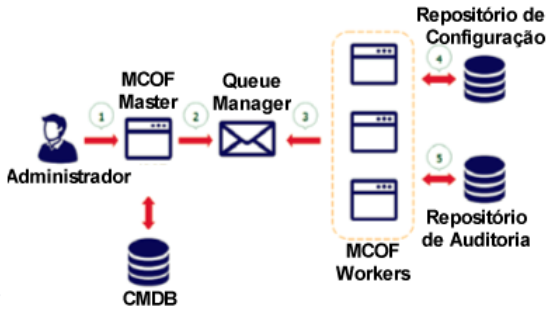

Figura 4.6: Arquitetura Lógica do MCOF[16]

.

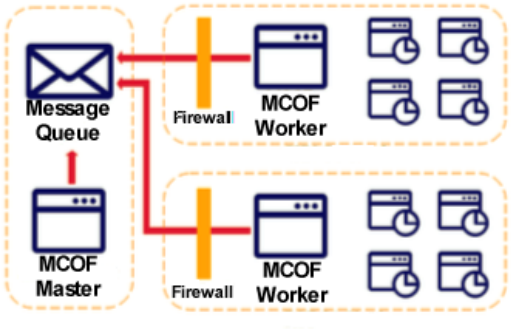

Figura 4.7: Diagrama de Implantação do MCOF.

o sistema de mensageria são adotados o Celery e o RabbitMQ e, para monitoramento, o Zabbix.

4.6 Arquitetura SDCon

Son e Buyya [115] propuseram o SDCon (*Software-Defined Cloud Controller*), uma plataforma desenvolvida para fornecer gerenciamento integrado aos recursos de computação e rede na infraestrutura de nuvem.

Conforme mostrado na Figura 4.8, a plataforma pode executar o posicionamento e migração de VMs, escalonamento do fluxo de rede, alocação de largura de banda, monitoramento em tempo real dos recursos, além medição do uso de energia. Os autores afirmam que, no futuro, o SDCon poderá ser estendido para suportar CMPs e controladores de rede alternativos, além da popular CMP OpenStack[12] e do OpenDayLight[120], uma plataforma aberta e modular para personalizar e automatizar redes definidas por software. O monitoramento é baseado no componente Ceilometer do OpenStack.

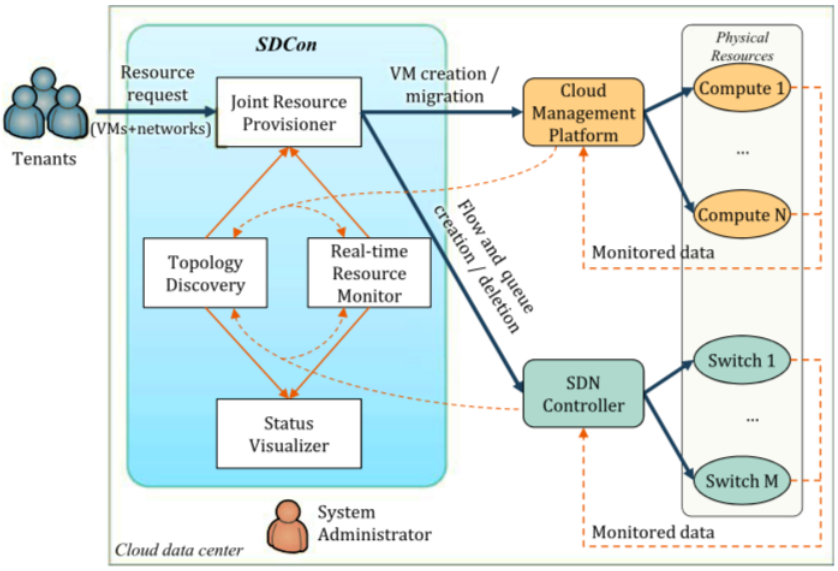

Figura 4.8: Arquitetura e Princípios de *Design* da SDCon.

4.7 Comparação entre os Trabalhos Relacionados

Após revisão da literatura e testes com CMPs de padrão aberto, não foi identificada uma alternativa de código aberto capaz de atender os requisitos deste trabalho integralmente pois, no cenário apresentado nos Tribunais, é essencial que os principais hipervisores corporativos (VMware, Hyper-V) sejam suportados integralmente.

Um dos fatores que se deseja eliminar com a implementação de interoperabilidade na nuvem comunitária proposta é o efeito de bloqueio em um só fornecedor (*vendor lock-in*). Abordagens como as propostas de *middlewares* compatíveis com CMPs populares, a exemplo do FogBow, são particularmente interessantes para viabilizar operações de integração, mas requerem a associação a outras ferramentas para obtenção de um modelo mais robusto de nuvem comunitária. Nesse cenário, SDCon e MCOF oferecem compatibilidade apenas com o OpenStack.

Outras alternativas como o Aurora e o DIRAC ainda não oferecem uma solução suficientemente desacoplada para atender infraestruturas mais heterogêneas, como é o caso do PJU. As abordagens consideradas pelos principais trabalhos relacionados estão descritas na Tabela 4.1, que apresenta as distinções acerca da características de integração e monitoramento.

De modo geral, observa-se que as soluções propostas evitam realizar múltiplas integrações, adaptando-se a algum tipo de CMP ou componente central para intermediar com os hipervisores. Contudo, isso potencialmente inviabiliza ou atrasa a realização de algumas tarefas, tais como a atualização de versões dos hipervisores, que passam a depender de evoluções nas bibliotecas ou APIs genéricas para interação.

Tabela 4.1: Avaliação das Características dos Trabalhos Relacionados.

	Acoplamento	**Provedores Suportados**	**APIs Suportadas**	**Integração**	**Monitoramento**
SDCon: Son e Buyya (2019)	Baseado em Componentes	OpenStack	OpenStack API, OpenDayLight API	Centrada no Provedor	Ceilometer/ Telemetry
CLOUDIATOR: Baur e Domaschka (2016)	Baseado em Componentes	OpenStack	API própria	Centrada no Provedor	Componente próprio (AXE)
MCOF: Lu *et al.* (2017)	Baseado em Componentes	OpenStack	OpenStack API	Centrada no Provedor	Zabbix
FogBow: Brasileiro (2016)	Fortemente Acoplado	OpenStack, OpenNebula	OCCI	Centrada no Provedor	Descentralizado
DIRAC: Casaj *et al.* (2014)	Fortemente Acoplado	OpenStack, OpenNebula, CloudStack	OCCI, AWS	Centrada no Provedor	Descentralizado
Aurora: Wickboldt *et al.* (2014)	Fortemente Acoplado	Libvirt	API própria	Centrada no Cliente	Nagios/ FlexACMS

Por exemplo, o suporte aos *clusters* Microsoft Hyper-V e ao VMware vCenter pela biblioteca libvirt ainda é precário, e a última versão suportada do VMware ESX é a 5.5[38]. Deste modo, boa parte dos Tribunais não poderia ser atendido, por já estarem rodando a versão 6.5 deste hipervisor.

Além disso, as soluções propostas são mais capazes de lidar com infraestruturas homogêneas, cuja integração é centrada no provedor, com exceção da Aurora Cloud Manager, que é centrada no cliente. Isso significa que para fazer uso destas ferramentas, o gerenciamento da infraestrutura em operação nos órgãos teria de ser alterado em função de plataformas ainda experimentais, sem suporte corporativo.

Os ambientes de virtualização no PJU não são homogêneos, as topologias, instalações e equipes de operação são diferentes, com necessidades distintas de integração. Embora os serviços de TIC mantidos pelos órgãos do PJU geralmente atendam usuários de todo o Brasil, observa-se um forte acoplamento das aplicações e dados, que utilizam apenas infraestruturas locais.

Esta prática ameaça a disponibilidade dos serviços, não só em nível de aplicação, mas também em relação à segurança dos dados em caso de catástrofes. Sendo assim, observa-se um efeito de *vendor lock-in*, decorrente do acoplamento das soluções oferecidas. Conclui-se que as abordagens que requerem infraestruturas de gerenciamento totalmente homogêneo para construção de uma plataforma de nuvem comunitária não são apropriadas nesse cenário. Assim, o capítulo seguinte irá apresentar a proposta de arquitetura desenvolvida neste trabalho.

Capítulo 5

Arquitetura Cloud.Jus

Este capítulo apresenta a arquitetura proposta, denominada Cloud.Jus. A Seção 5.1 apresenta uma Visão Geral acerca da proposta. A Seção 5.2 define os conceitos e a abordagem da arquitetura em alto nível, enquanto a Seção 5.3 especifica os detalhes de implementação dos componentes da plataforma. A Seção 5.4 apresenta uma avaliação da arquitetura por meio de experimentos relacionados à elasticidade para implantação de máquinas virtuais em rajada. Por fim, a Seção 5.5 descreve a contribuição da pesquisa, considerando os resultados obtidos.

5.1 Visão Geral

Uma das consequências de desenvolver arquiteturas extremamente centralizadas são os conflitos para adoção de padrões homogêneos restritos. A experiência mostra que esta abordagem é limitada em ambientes complexos [109][121], isto é, com vários tipos de serviços e tecnologias em operação. Assim, ao desenvolver uma arquitetura de TIC, é necessário particionar os componentes em uma granularidade adequada para a atuação dos administradores, uma vez que enxergar componentes menores irá permitir uma previsibilidade maior do comportamento da plataforma.

Sessions [122] afirma que, para elaborar arquiteturas em ambientes segmentados e complexos, não é aconselhável criar projetos universais que pretendam sanar todo tipo de situação imediatamente. Pelo contrário, deverão ser projetados componentes intercambiáveis, desenvolvidos um a um, até que se passe para o próximo, considerando que o particionamento irá diminuir a complexidade geral e a iteração irá aumentar a probabilidade de construir sistemas de alta coesão e baixo acoplamento [123].

Para Edwards [17], a complexidade deve ser abordada com foco na velocidade, não na completude, de modo que as organizações devem pensar em ferramentas segundo uma abordagem em cadeia, pois este é o único padrão de *design* que todos poderão concordar

em padronizar. Este tipo de abordagem é essencial para garantir o baixo acoplamento, buscado como boa prática de arquitetura:

"Ter ferramentas individualmente boas é a maneira mais simples e fácil de se realizar algo, sendo mais relevante a preocupação sobre como se dará a integração entre elas: qual o conjunto de ferramentas necessário para conectar as aplicações ao provisionamento de infraestrutura? Como gerenciar esta integração de maneira lógica e consistente? Assim, é mais fácil compreender o uso de ferramentas como componentes intercambiáveis e o seu papel no processo, ao invés de alimentar batalhas entre framework X ou Y, pois isso não ajuda ninguém nem é importante para as organizações."

Considerando que grandes empresas investem no desenvolvimento de novos produtos e, como resultado, estes funcionam melhor utilizando uma combinação de recursos nativos [59], não faz sentido atuar no desenvolvimento de ferramentas que já estão muito maduras e amplamente disponíveis em hipervisores corporativos. Os serviços oferecidos pelos hipervisores corporativos são uma excelente maneira de operar o ambiente de virtualização e interagir com a infraestrutura base. São fáceis de configurar e oferecem APIs e consoles de gerenciamento.

Ao aproveitar os recursos já conhecidos destas ferramentas, não exigindo mudanças na forma em que os administradores de virtualização de cada órgão atuam com o gerenciamento nas tarefas rotineiras, tais como, o uso das consoles do VMware vCenter ou Hyper-V para migração de VMs, monitoramento e ajuste de posicionamento da carga de trabalho, alocação de recursos subjacentes como, por exemplo, o fornecimento de volumes de bloco para servir aos *datastores*, dentre outros, cada provedor da nuvem comunitária continua como responsável por manter os componentes do cluster de virtualização, reiniciando os serviços com falha e realizando atualizações. Essa abordagem reduz a carga administrativa de um provedor centralizado.

Assim, a arquitetura Cloud.Jus trabalha em estreita colaboração com esses hipervisores e aprimora o que eles oferecem, já na forma de infraestrutura como um serviço em nuvem, gerenciando tarefas de provisionamento para qualquer número de instalações hospedadas em execução em qualquer provedor. A plataforma irá orquestrar o manuseio dessas infraestruturas em um nível mais alto de abstração, oferecendo gerenciamento unificado para o provisionamento de máquinas virtuais e interações com serviços de rede para, efetivamente, facilitar a utilização de uma infraestrutura distribuída.

Assim, o usuário será capaz de gerenciar recursos alocados em vários *datacenters* por meio de uma única interface, mas seguindo uma política de provisionamento adaptável a cada ambiente. Por exemplo, se o órgão participante da nuvem comunitária possuir um provedor de identidade interno, como o Active Directory, este poderá ser adotado para empregar seus usuários e grupos para controlar o acesso ao ambiente.

Dessa maneira, almeja-se que a opção por um projeto de IaaS de baixo acoplamento viabilize a confirmação da hipótese do trabalho, isto é: "que as instituições possam se beneficiar ao empregar implementações semelhantes, no intuito de promover interesses comuns, tais como manutenção da segurança na distribuição geográfica de seus dados, maior redundância e tolerância a falhas e catástrofes, implementação de mecanismos para garantir a estratégia de continuidade de serviços, resguardando o interesse público e a qualidade na implementação de novos serviços".

5.2 Descrição Conceitual

Para uma abordagem de *design* bem-sucedida, a arquitetura de nuvem deve ser tolerante à troca de fornecedor em seus componentes, serviços ou processos [62][3]. Assim, devem ser contemplados os seguintes aspectos [124]:

- Foco no Público Alvo: os resultados devem ser planejados de acordo com as escolhas e ações observadas no usuário;
- Encontrar boas soluções: desenvolver soluções e serviços implica em escolher os *trade-offs*, de acordo com a necessidade do público alvo;
- Prototipação ágil: devem ser construídos modelos rápidos, de modo que ideias inadequadas sejam descartadas rapidamente, logo após os primeiros testes;
- Trabalho Colaborativo: os atores do processo não devem trabalhar de forma isolada, pois isso ocasiona perda de contexto e prejudica o desempenho da solução proposta;
- Soluções Personalizadas: cada organização tem suas especificidades, então cada solução desenhada deverá se aplicar ao contexto em que o objeto ou serviço está inserido.

A Figura 5.1 descreve a arquitetura em uma abordagem *top-down*, e organiza os componentes conceitualmente semelhantes em 4 camadas, a saber: Camada de Interface do Usuário (UI - *User Interface*), Camada de Abstração (AL - *Abstraction Layer*), Camada de Integração de Serviços (SI - *Service Integration*) e Camada de Infraestrutura Gerenciada (MI - *Manager Infrastructure*).

A Camada UI é responsável por lidar com as ações dos usuários finais e administradores, requisições de provisionamento, modificação e remoção de recursos em diversas infraestruturas, bem como fornecer acesso fácil e compreensivo ao ambiente instalado e monitorado. Oferece opções de interação via GUI, por meio da implementação de um

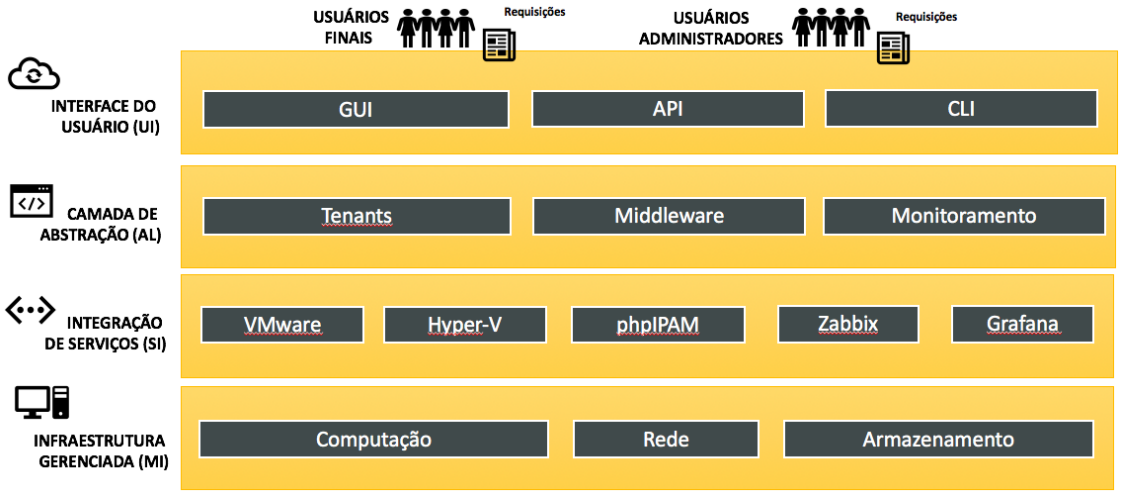

Figura 5.1: Arquitetura de Alto Nível da Cloud.Jus.

Painel de Gerenciamento, denominado *Dashboard*, que é utilizado para acionar os mecanismos de controle e provisionamento de recursos da nuvem, além de consolidar a exibição de alocação e consumo. Também oferece a opção de interação via CLI e API.

A Camada AL é responsável por estabelecer o conceito de *tenant*, o *middleware* que orquestra os componentes da plataforma, e também agrupa o conjunto de recursos avançados de monitoramento, fornecidos por uma infraestrutura especializada, que será descrita nas próximas Seções. Cada *tenant* corresponde a um conjunto de recursos pré-determinado, os quais podem ser instanciados por um determinado grupo de usuários na infraestrutura virtual compartilhada. O *middleware* é o componente central da arquitetura, e estabelece a orquestração da plataforma e os mecanismos de interligação com as camadas subjacentes.

A Camada SI agrupa o conjunto de componentes que gerenciam cada infraestrutura especializada, tais como os *clusters* dos hipervisores, endereçamento de rede, regras de monitoramento. A opção de *design* indicada pela arquitetura proposta é realizar o aproveitamento das soluções nativas oferecidas pelas plataformas já instaladas, construindo serviços de integração. Enquanto as interfaces simples da Camada UI traduzem as requisições obtidas via *front-end*, um *middleware* da Camada AL trata de comunicar os componentes e serviços de *back-end* para lidar com as plataformas de gerenciamento de infraestrutura já oferecidas pelos órgãos envolvidos na nuvem comunitária.

Ao contrário de outros autores, que propuseram trabalhos com características mais centradas no provedor, ou seja, adequam-se melhor a tecnologias homogêneas e exigem mudança de cultura dos clientes, a arquitetura da Cloud.Jus possui foco no cliente e administradores da nuvem, isto é, preserva a compatibilidade entre os componentes mesmo em ambientes heterogêneos e disponibiliza uma interface amigável, que adota ferramentas

oficialmente suportadas por seus fabricantes como, por exemplo, serviços desenvolvidos para acionar módulos Powershell de gerenciamento do VMware e Hyper-V em *back-ground*.

A Camada MI lida diretamente com a operação dos ativos e recursos gerenciados, tais como servidores físicos, armazenamento de dados, ativos de rede, *firewalls*, *links* de Internet, dentre outros. O micro-gerenciamento desta infraestrutura não deve ser afetado pela plataforma, ficando a cargo de cada instituição prover o suporte necessário. Acerca da capacidade do conjunto de recursos empregado, cada órgão terá direito a utilizar a mesma capacidade computacional cedida em suas instalações de maneira distribuída nos *datacenters* que estarão operando na nuvem comunitária.

5.3 Funcionalidades da Nuvem

A seguir estão descritas as principais funções da nuvem, e como são desempenhadas pelos componentes especificados durante o fluxo de gerenciamento da infraestrutura:

- Painel de Controle - *Dashboard* (GUI): interface gráfica que centraliza as operações de criação, alteração e exclusão de recursos, controlados de modo unificado. Inclui a exibição dos recursos alocados e consumidos.
- Interface de linha de comando (CLI): possibilita ao administrador gerenciar componentes e recursos computacionais da nuvem, tais como: máquinas, redes e volumes;
- *Application Programming Interface* (API): permite a integração de aplicações e serviços, via protocolo HTTP e comandos RESTful;
- Orquestração (*middleware*): possibilita o trabalho unificado e coordenado entre componentes e ambientes distintos, automatiza tarefas e procedimentos na utilização do ambiente, coordena as requisições via GUI, CLI ou API, de forma que as interações com a nuvem sejam transparentes ao usuário. O *middleware* mantém comunicação com um repositório de metadados, um sistema gerenciador de filas (NATS) e um ou mais nodos de trabalho (*workers*), os quais processam as tarefas encaminhadas utilizando as APIs nativas dos hipervisores.
- Gerenciamento de Federação: o orquestrador (*middleware*) é o responsável por envolve a integração entre ambientes diversos de nuvem para operar como se fossem uma só infraestrutura, de modo transparente e coordenado;
- Gerenciamento de Recursos: o suporte ao funcionamento da infraestrutura computacional e das cotas de recursos disponibilizados é de responsabilidade de cada membro da comunidade, que continuará a atuar nas ferramentas já instaladas para operar cada solução de virtualização empregada e suas camadas subjacentes, tais

como servidores físicos, rede e armazenamento. O endereçamento de rede é automatizado via API pelo phpIPAM[125], que se integra ao conjunto de IPs e VLANs instanciadas pelo provedor local. Cada hipervisor gerencia os volumes de bloco e o catálogo de imagens das VMs apresentadas aos clientes. Todos os recursos oferecidos pelo provedor local (computação, armazenamento e rede) podem ser provisionados de maneira automática pela plataforma Cloud.Jus.

- Monitoramento: Permite que o administrador e usuários acompanhem de modo geral a utilização dos recursos computacionais empregados e disponíveis. A funcionalidade é provida pelo Zabbix, de forma integrada com a nuvem, cada VM criada já dispõe de um conjunto básico de métricas implantadas. A exibição consolidade é criada automaticamente pelo Grafana[126], por meio de expressões regulares. O Grafana é uma plataforma que permite consultar, visualizar, alertar e coletar métricas de monitoramento por diversos métodos, APIs, scripts, servidor Zabbix[103], dentre outros, apresentando os dados em painéis personalizados.
- Gerenciamento de Eventos: Detecta incidentes e problemas na infraestrutura, norteando as ações que irão viabilizar o cumprimento dos SLAs estabelecidos. É implantado via Zabbix[103], plataforma de monitoramento capaz de coletar dezenas de milhares de métricas de monitoramento de servidores físicos, máquinas virtuais, dispositivos de rede e várias aplicações simultaneamente.

A Figura 5.2 apresenta os componentes das camadas sendo orquestrados pelo *middleware*, o qual traduz os requisitos de implantação do usuário, neste caso solicitados via GUI, e os descreve para execução na infraestrutura virtual via Powershell, por meio de serviços de integração executando em nodos de trabalho (função *worker*). Estes serviços de integração foram desenvolvidos usando sintaxe e ferramentas nativas de cada hipervisor, mantendo suporte integral de seus fabricantes.

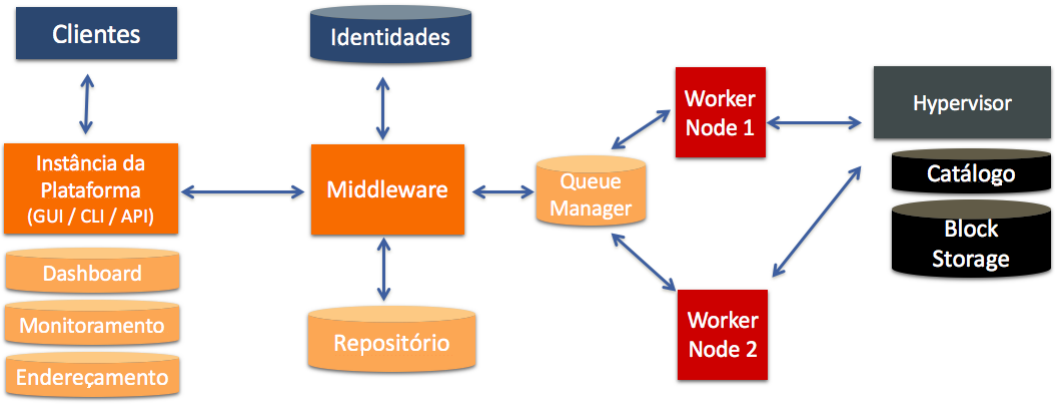

Figura 5.2: Caminho para Interação entre o *Middleware* Orquestador e os Recursos.

O *middleware* mantém um repositório de metadados, cuja atualização é comandada por processos de sincronização e notificação de eventos em sistema gerenciador de filas. Nas próximas Seções serão descritas as características de implementação de cada um dos componentes da arquitetura.

5.3.1 Painel de Controle - *Dashboard* (GUI)

A arquitetura Cloud.Jus estabelece uma interface gráfica amigável e exibe um painel de controle que demonstra a utilização dos recursos, sendo eficiente nas interações com os usuários e administradores da nuvem, mas sem comprometer o gerenciamento nativo de cada hipervisor utilizado, bem como dos ativos de rede, sistemas de armazenamento e servidores físicos das camadas subjacentes.

Assim, a plataforma fornece o *Dashboard*, que tem a função de integrar e controlar os diversos componentes da arquitetura de nuvem, abstraindo a complexidade do gerenciamento dos serviços e recursos por meio de uma GUI [127]. A implementação realizada na plataforma Cloud.Jus possibilita ao administrador monitorar a alocação e utilização dos recursos computacionais, serviços e componentes dentro do seu *tenant*. Assim, facilmente o usuário do *tenant* pode identificar qual estado de suas instâncias de VMs e quantidade de memória, disco e processamento associados.

Figura 5.3: *Dashboard* da Plataforma Cloud.Jus.

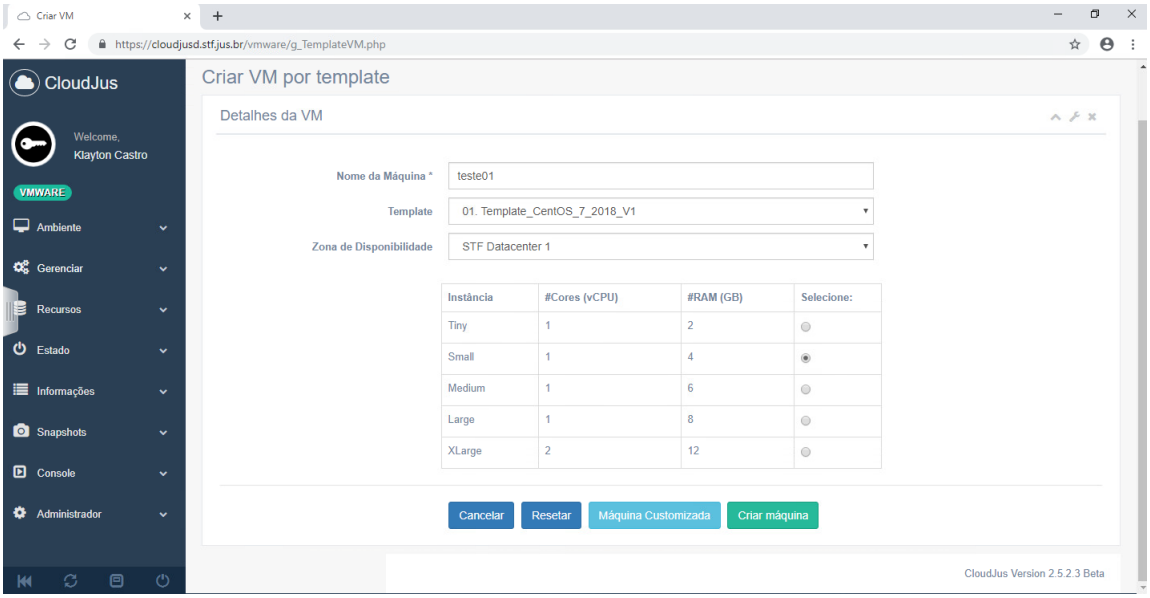

Figura 5.4: Criação de Máquina Virtual via *Dashboard*.

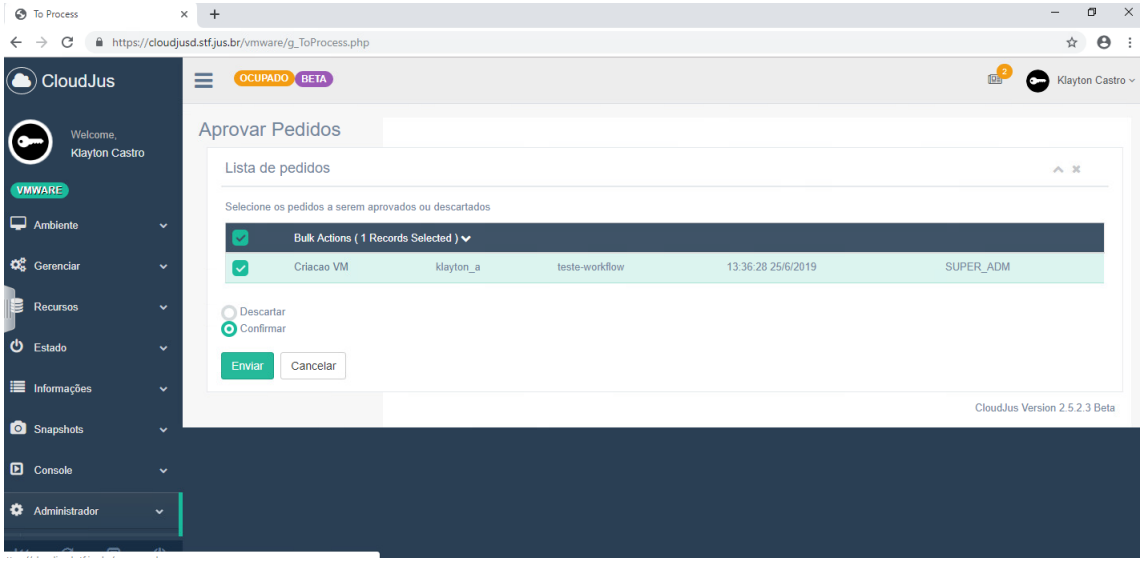

Figura 5.5: Aprovação de Máquina Virtual Customizada

O *Dashboard* centraliza as operações de criação, alteração e exclusão de recursos. A linguagem de programação escolhida para implementação foi o PHP, por sua simplicidade e agilidade ao incluir um conjunto robusto de bibliotecas padronizadas, que facilitaram a integração com o ambiente de autenticação e autorização LDAP (gerenciamento de identidades) e o uso de *frameworks* como o Boostrap, que trazem componentes visualmente ricos em uma implementação baseada no Gentelella, conforme apresentado na Figura 5.3. A autenticação e autorização de usuários é integrada ao provedor de gerenciamento de identidades do ambiente correspondente a cada *datacenter*.

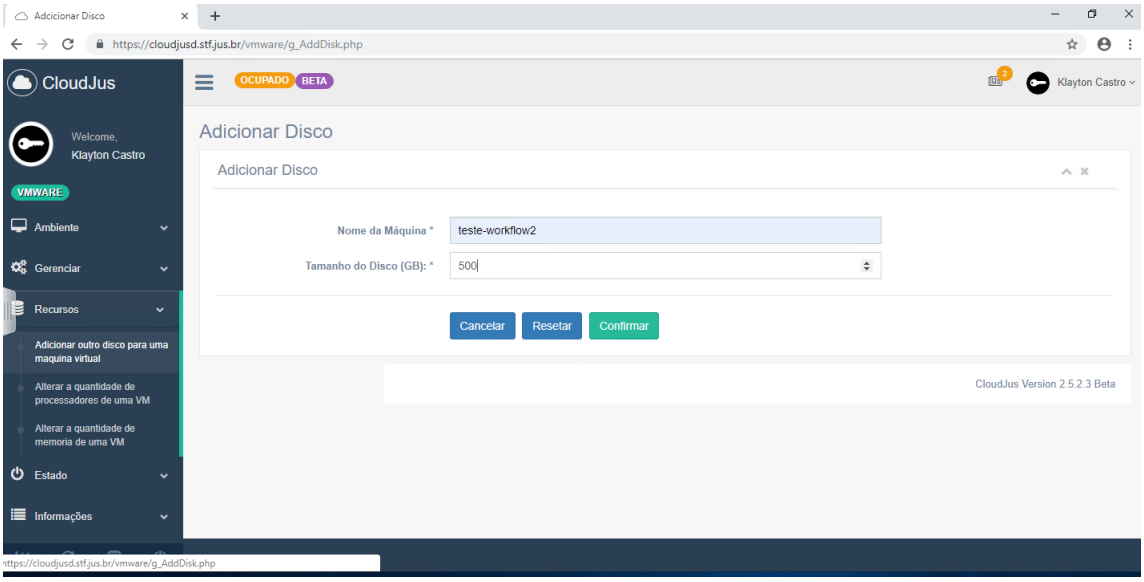

Figura 5.6: Adição de Armazenamento Persistente de Bloco

O administrador de nuvem tem o máximo de privilégios no gerenciamento no *Dashboard*, sendo assim, ele controla e define as cotas de recursos para os usuários do ambiente de nuvem que serão atribuídas em cada hipervisor. Estas configurações são realizadas localmente, nos arquivos de configuração da instância da plataforma em execução.

Por outro lado, um usuário de nuvem tem acesso a uma interface exclusiva na qual ele pode visualizar os grupos de recursos designados a ele, instanciar máquinas virtuais, tendo controle total desde a alocação de recursos (infraestrutura) até a execução de console, que utiliza as implementações nativas do VMware e do Hyper-V.

A Figura 5.4 demonstra a criação de uma máquina virtual no *tenant* "VMware"do "Datacenter 01"do STF, com uma imagem padrão do CentOS 7.5 e configuração de instância do tipo "Small", com 1vCPU, 4GB RAM e 60GB de HD. Além dos tipos padrão de instância ("*Tiny*", "*Small*", "*Medium*", "*Large*"e "*eXtra Large*"), que são pré-autorizados para implantação dos usuários das áreas de desenvolvimento e operação, há a possibilidade de solicitar uma configuração personalizada que, neste caso, é submetida a um *workflow* de aprovação. Os administradores da nuvem poderão validar ou não a configuração customizada proposta pelos usuários, conforme Figura 5.5.

Também é possível a criação de um conjunto de máquinas semelhantes em lote e outras funções básicas, como acrescentar armazenamento de bloco adicional à VM criada. Estas funções estão disponíveis na aba "Recursos", conforme Figura 5.6.

A efetiva configuração dos recursos dependerá da disponibilidade da cota estabelecida para o grupo de usuários daquele *tenant*. Ela pode ser do tipo *hard*, que obriga que os recursos requeridos estejam dentro da cota estabelecida de processamento, memória e disco, ou do tipo *soft*, cujos limites são flexíveis e podem ser extrapolados mediante

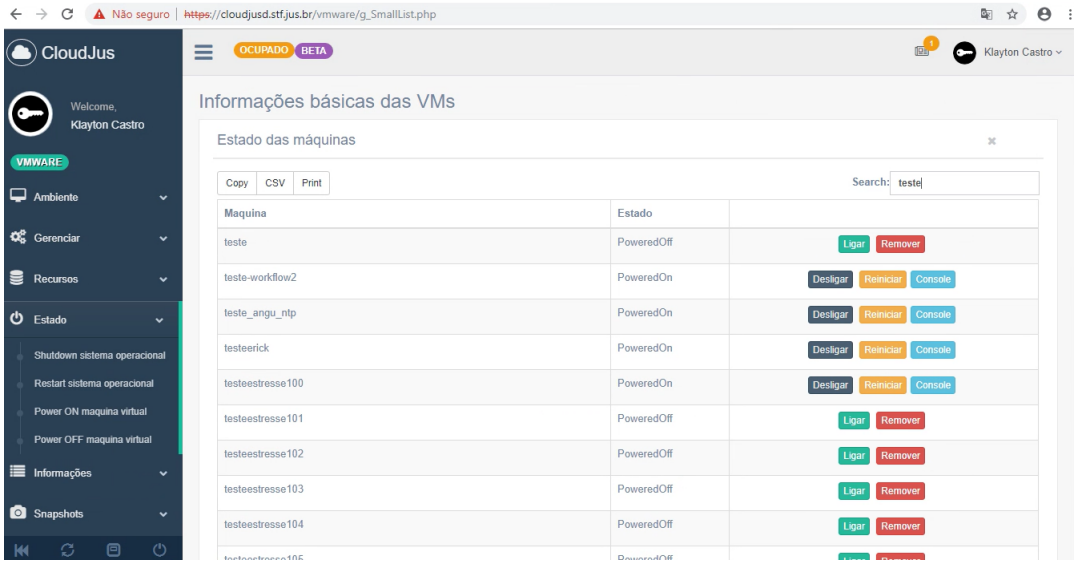

Figura 5.7: Operações de Estado e Acesso à Console em VMs

realização de um contrato prévio. Assim, a cota poderá ser reajustada automaticamente conforme a necessidade.

Os usuários podem autenticar-se na plataforma para realizar a criação das instâncias de VMs e configurações adicionais porventura necessárias como, por exemplo, acréscimo de volumes de bloco, interface de rede, quantidade de memória ou núcleos de processamento. Também é possível realizar operações de estado, tais como, ligar, desligar e reiniciar a instância, conforme mostrado na Figura 5.7. E, além disso, é possível acionar a console das máquinas, realizar cópias instantâneas e reversões de estado (*snapshots*), conforme apresentado na Figura 5.8.

5.3.2 Orquestrador - *Middleware*

São fornecidas aos usuários da Cloud.Jus máquinas virtuais, particularmente configuradas com quantidades personalizadas de recursos de armazenamento em bloco, núcleos de processamento e memória, em *tenants* que são associados a determinado conjunto de recursos de um provedor, estabelecidos em hipervisor. No entanto, a forma de interagir com o ambiente é a mesma. A plataforma automatiza e padroniza a interação entre os componentes para provisionar, reconfigurar e acompanhar a utilização destes recursos, via interface web.

O *middleware* orquestrador surgiu da necessidade de executar funções básicas de gerenciamento de maneira transparente, inclusive em operações de larga escala como, por exemplo, a criação de instâncias de VMs em lote. Dessa forma, ele oferece um gerencia-

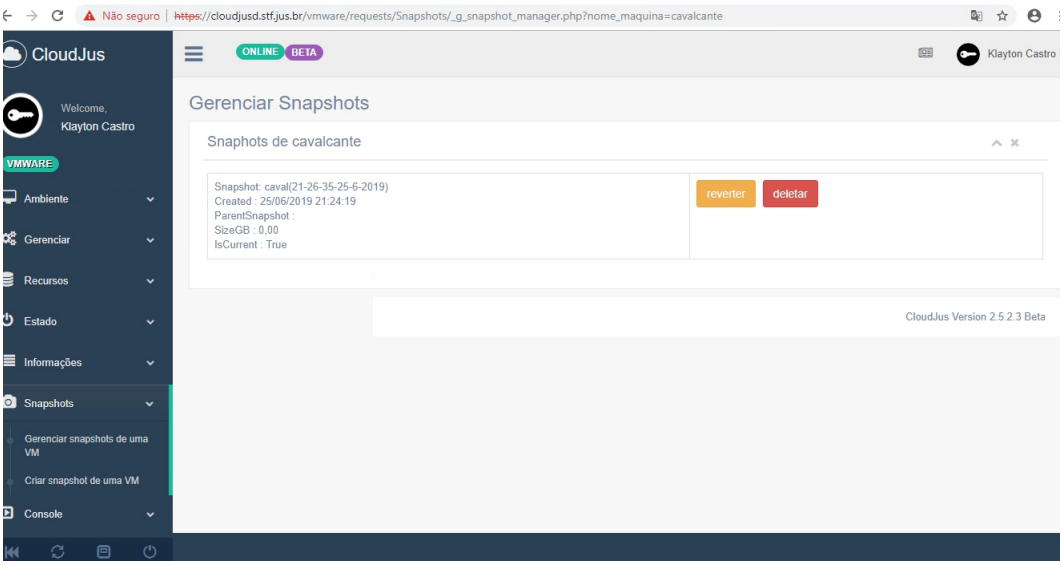

Figura 5.8: Operações de Instantâneos (*Snapshots*)

mento automatizado das tarefas e procedimentos necessários para utilização do ambiente, coordenando as demandas repassadas pelas interfaces de *front-end*.

Foram desenvolvidos módulos de *back-end* para tratar dos serviços de integração entre as Camadas de Infraestrutura associadas à Cloud.Jus. A implementação foi realizada por meio de *scripts* Powershell que encaminham as requisições aos hipervisores por meio dos *workers*, e comunicam as ações ao *middleware*, que trata da atualização do repositório de metadados conforme apresentado na Figura 5.10, que demonstra a interação entre os componentes de orquestração e os demais recursos.

O padrão OCCI foi tomado como referência por oferecer um protocolo RESTful e API capaz de atuar como *front-end* genérico de serviço para a estrutura de gerenciamento interno de cada provedor e suportar os tipos básicos de tarefas de gerenciamento de computação, armazenamento e rede. Assim, basta formatar requisições HTTP para fornecer os metadados necessários via aplicação web (*front-end*) ou linha de comando, fazendo uso do *curl*, ou ferramenta semelhante, de modo que estas mensagens formatadas sejam submetidas ao orquestrador. Por sua vez, este irá enfileirá-las para processamento nos nodos atribuídos com a função de (*worker*), para serem executados no *cluster* de hipervisores correspondente ao *tenant*.

Por exemplo, o usuário "klayton@rede.stf.gov.br", do grupo "super-administrador", deseja instanciar uma máquina virtual de nome "teste-krc", com arquitetura "x64", Template de Sistema Operacional "Linux CentOS 7.5, kernel 4.4", do tipo "Small", ou seja, com "4GB"de memória RAM e "1"núcleo de processamento. Deste modo, ao acessar o Painel de Gerenciamento da plataforma Cloud.Jus e preencher o formulário específico para criação de uma VM, será submetida ao orquestrador uma requisição contendo os

```
{
   "occi.compute.architecture": "x64",
   "occi.compute.cores": "1",
   "occi.compute.hostname": "teste-krc",
   "occi.compute.memory": "4",
   "occi.compute.state": "inactive",
   "occi.compute.state.message": "14-26-10 24-5-2019",
   "occi.compute.template": "1",
   "occi.compute.folder": "1",
   "occi.compute.user": "klayton_a@rede.stf.gov.br",
   "occi.compute.group": "SUPER_ADM"
}
```


Figura 5.9: Requisição de criação de VM no formato OCCI.

atributos de computação descritos na Figura 5.9.

Após a primeira implementação do protótipo, foi verificada uma lacuna em relação ao processamento elevado do componente de orquestração, decorrente da dificuldade em paralelizar as tarefas de manipulação de recursos da nuvem e manter a sincronia do repositório de metadados. Após a realização dos primeiros experimentos, foi considerada a a adoção de um sistema de mensageria distribuída como ferramenta de apoio *middleware* orquestrador, isto é, o NATS [85]. O intuito desta abordagem foi desacoplar a fila de mensagens do orquestrador, paralelizar o processamento das tarefas e facilitar a escalabilidade do *middleware*, considerando que adotar um componente baseado em sistema de arquivos demonstrou limitações em termos de arquitetura e otimização de recursos.

Ao empregar o NATS como responsável para comunicação entre os componentes, a solução proposta de fato ganhou escalabilidade e desacoplamento, vindo a substituir os sistemas de arquivos (FS - *File System*) de rede, que dependiam de uma programação mais complexa para manutenção da integridade da base de configuração do repositório, a qual precisava ser atualizada de maneira serializada a cada tarefa, tornando a plataforma menos responsiva e limitando suas possibilidades de expansão.

Com um sistema de mensageria distribuída, os problemas de desempenho da abordagem com *File System* puderam ser contornados, em virtude da solicitação baseada em NATS apresentar baixa sobrecarga em termos do protocolo. Por exemplo, não há necessidade de estabelecer uma conexão ponto-a-ponto com o serviço de integração para o qual está sendo feita a solicitação. Assim, basta o produtor da mensagem, no caso, a própria

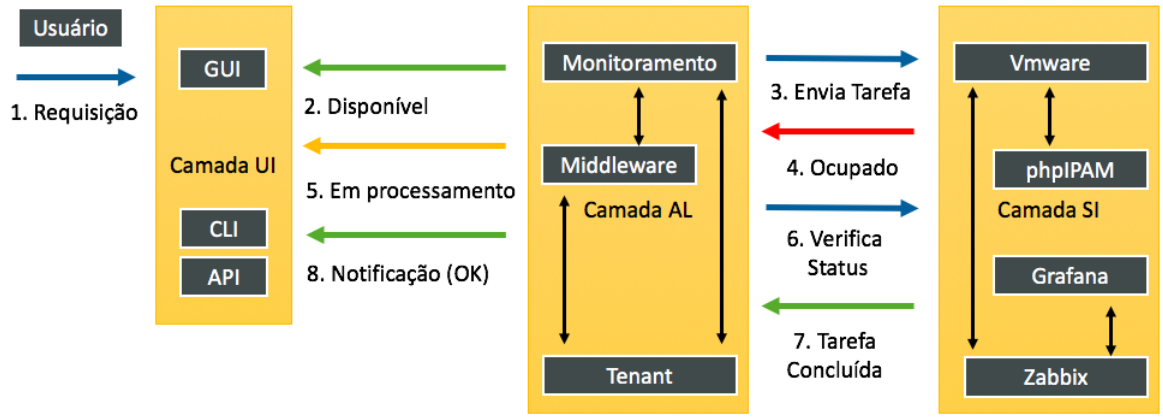

Figura 5.10: Fluxo para Implantação de uma Requisição.

interface de *front-end*, comunicar-se com um servidor NATS disponível, publicar a mensagem, e aguardar até que a mensagem de resposta do consumidor (*worker*) seja entregue em um processo que ocorre de forma assíncrona, conforme apresentado na Figura 5.11.

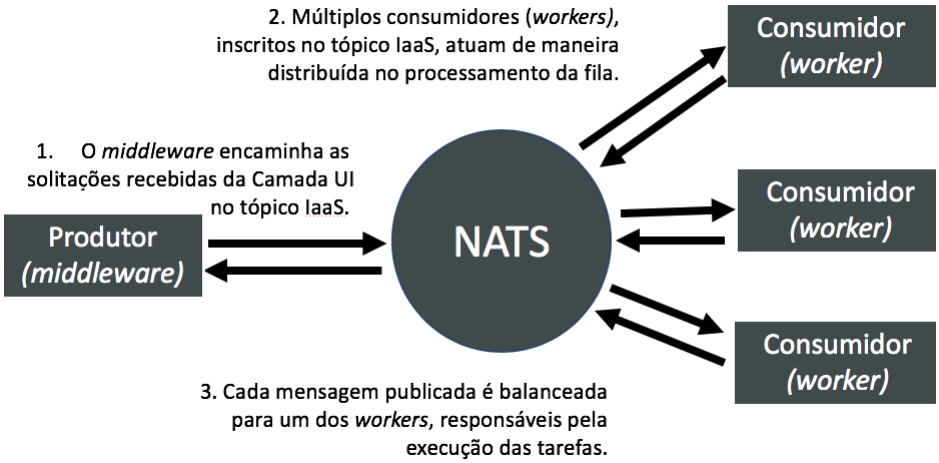

Figura 5.11: *Workflow* de *Request-Reply* utilizando o NATS.

Já na abordagem com sistema de arquivos em rede, uma série de passos para manter a convergência de estados era requerida, como por exemplo a atualização parcial de configuração, que exigia respostas do ambiente hipervisor para manter o repositório consistente. Além disso, o NFS (*Network File System*) não apresenta boa escalabilidade e é sujeito à latência de rede, especialmente em ambientes altamente distribuídos. A análise acerca destas duas abordagens em um experimento envolvendo elasticidade está relatada na Seção 5.4.

5.3.3 Gerenciamento de Federação

As instâncias de *middlewares* de orquestração podem ser combinadas em uma federação do tipo *peer-to-peer*[74], de modo que recursos em um *datacenter* remoto podem ser instanciados em uma interface padronizada e uniforme. Para isso, estes componentes precisam se comunicar via Internet, ou ainda, via redes dedicadas, como a Infovia, uma rede governamental que interliga diversos órgãos públicos. As mensagens entre os *middlewares* federados são criptografadas, mantidas no leve formato JSON. A Figura 5.12 retrata a topologia de federação adotada na implementação do protótipo da Cloud.Jus.

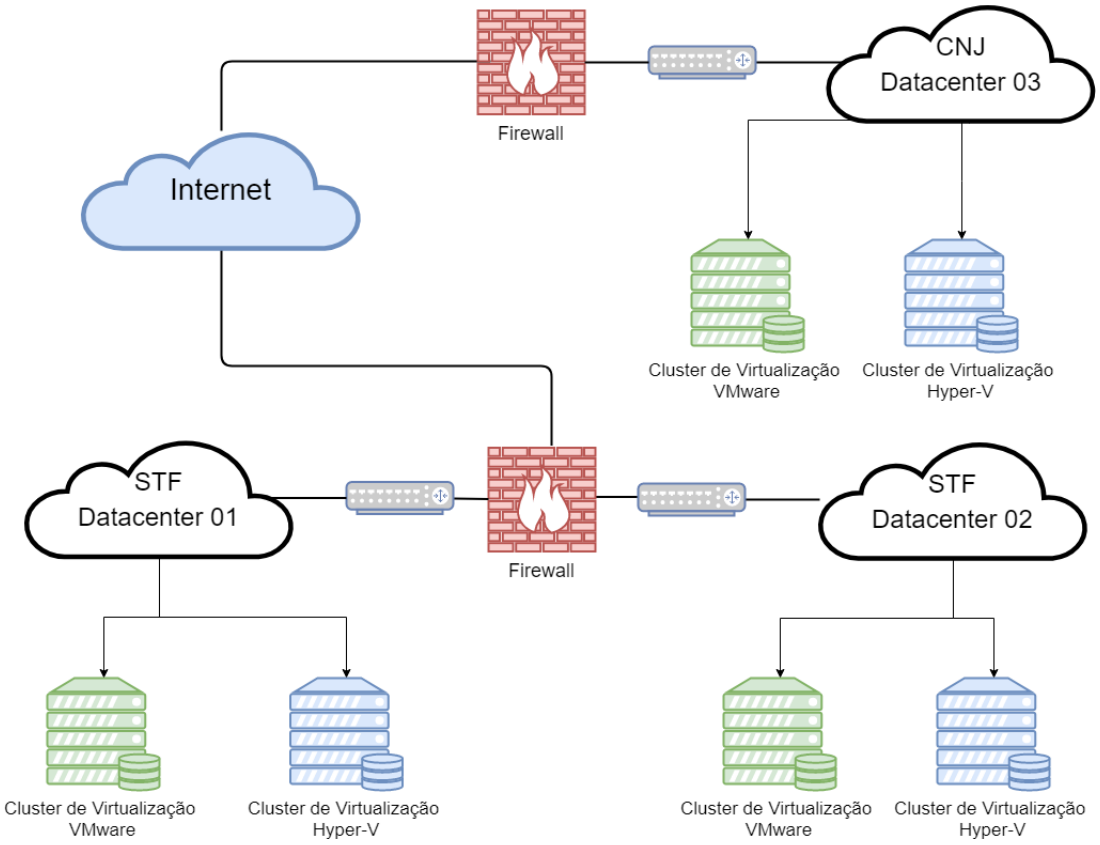

Figura 5.12: Topologia da Federação entre 03 Datacenters.

5.3.4 Gerenciamento de Recursos

Conforme descrito na Seção anterior, o *middleware* orquestrador mantem as informações atualizadas no repositório de metadados, que poderá ser consultado pelos demais componentes de maneira assíncrona. As tarefas recebidas via interface GUI, API ou CLI serão processadas em nodos de *back-end* (*workers*), cuja coordenação é conduzida pelo *middleware*, com apoio do sistema de mensageria distribuída.

Isso ocorre de maneira desacoplada ao funcionamento da infraestrutura computacional disponibilizada. A interface gera a requisição em formato de JSON, que é efetivamente processada em um nodo *worker*, o qual se comunica com o hipervisor via módulos Powershell. O formato das solicitações é semelhante, mas os *scripts* podem ser tratados separadamente para cada operação, variando de acordo com o hipervisor, no caso, Hyper-V ou VMware.

Em alguns instantes, os usuários recebem via e-mail cadastrado as credenciais para acessar o recurso provisionado. Após logar-se via console na VM instanciada, podem ser associados IP e nome DNS. Esta VM é inicializada com uma imagem do sistema escolhido (CentOS, Ubuntu ou Windows Server) que roda um script de autoconfiguração após o primeiro logon do usuário, o qual passa a ter acesso a um recurso local ou remoto que, neste caso, poderá ser gerenciado de maneira semelhante ao que seria feito em seu próprio datacenter.

Blocos de endereçamento IP são pré-reservados nos segmentos de rede interna ou DMZ (*DeMilitarized Zone*) com respectivo NAT (*Network Address Translation*) e liberação de acesso externo SSH e HTTP/HTTPs aos membros da nuvem comunitária. A DMZ é o segmento de rede onde são associadas as VMs que hospedam serviços que devem ser acessados externamente, isolando-os da rede interna do provedor. Ao rotear uma requisição da rede privativa (DMZ) para a internet, o *firewall* do provedor troca o endereço IP de origem da mensagem pelo endereço IP real e encaminha a transmissão. Foi utilizada como ferramenta de código aberto phpIPAM (*IP Address Management*)[125] para automatizar o endereçamento de rede das VMs. Seu objetivo é fornecer gerenciamento de endereços IP com sua API REST.

Permanece como responsabilidade de cada membro da comunidade atuar nas ferramentas existentes em cada solução de virtualização empregada para suportar hospedagem dos recursos da nuvem (VMware ou Hyper-V), acompanhar a alocação dos *datastores*, otimização de recursos, posicionamento das VMs, dentre outras tarefas rotineiras, que poderão ser efetuadas assim como antes. O posicionamento da VM é a funcionalidade que permite que um hipervisor realize a movimentação de uma VM para outro hipervisor, do mesmo tipo.

5.3.5 Monitoramento

Este componente permite que o administrador e usuários acompanhem de modo geral a utilização dos recursos computacionais empregados e disponíveis [127]. Foi implementada a integração entre a plataforma Cloud.Jus e as ferramentas Zabbix e Grafana, via API. O uso associado de Zabbix e Grafana se dá por serem ferramentas muito robustas e flexíveis, oferecerem APIs RESTful, já serem utilizadas em boa parte dos órgãos, aproveitando o

conhecimento dos administradores para realizar a operação e estabelecimento de novas métricas para refinar o monitoramento.

Figura 5.13: Monitoramento Automatizado por meio de Expressões Regulares do Grafana e Coleta de Dados do Zabbix.

Enquanto o Grafana é utilizado para apresentação dos dados em painéis construídos por meio de expressões regulares, o Zabbix efetua o monitoramento de desempenho e disponibilidade de ativos e serviços, com mínimo esforço para configuração das métricas desejadas. Estes dados podem ser coletados de diferentes fontes, tais como: descoberta automática, agentes de software instalados nos servidores, *scripts*, SNMP (*Simple Network Management Protocol*) ou IPMI (*Intelligent Platform Management Interface*), dentre outros.

O conjunto de métricas suportado é praticamente ilimitado, e podem ser realizadas customizações de maneira fácil na ferramenta Zabbix, refletindo imediatamente no Grafana por meio de uma integração nativa, via API. Ambos os softwares podem exibir as informações geradas em gráficos ou consolidá-las em painéis inteligentes e personalizáveis, fornecendo o monitoramento completo de infraestrutura ou aplicações. Estas ferramentas apresentam informações em tempo real acerca de métricas como utilização e fila de CPU, uso de memória, tráfego de rede, consumo de disco, dentre outras, conforme mostra a Figura 5.13.

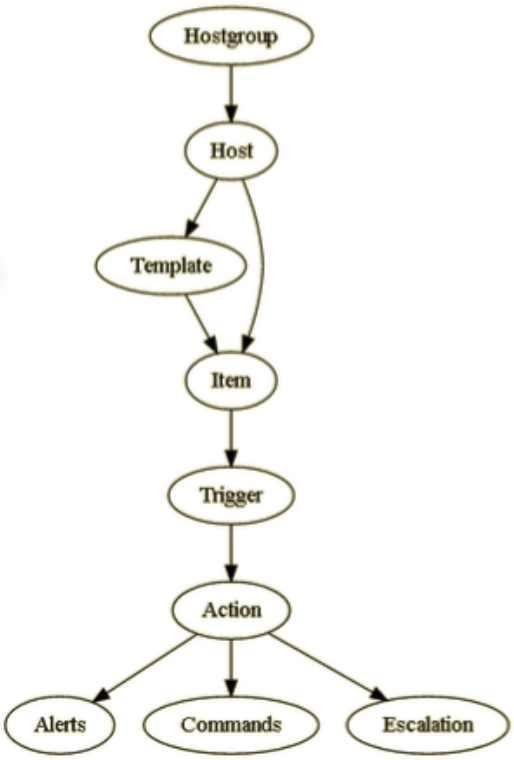

Figura 5.14: Esquema de Relacionamento e Gerenciamento de Eventos.

5.3.6 Gerenciamento de Eventos

O Gerenciamento de Eventos é responsável pela detecção de incidentes de infraestrutura, norteando as ações que irão viabilizar o cumprimento dos SLAs estabelecidos [127]. A Cloud.Jus possui como componente central de monitoramento o Zabbix. Assim, após realizar a coleta de estados (via agente, *scripts* ou SNMP) e persistir dados na forma de séries temporais, ou seja, uma coleção de observações feitas sequencialmente ao longo do tempo, o controle das informações é estabelecido pelas seguintes associações:

1. Criação de *hosts*, que são ativos a serem monitorados;
2. Criação de itens de monitoramento, que são as métricas estabelecidas para coleta;
3. Criação de *triggers*, que são as regras lógicas para analisar os valores obtidos nos itens e podem gerar eventos, cujo nível de severidade é atribuído em cinco níveis ("Desastre", "Alto", "Médio", "Atenção"e "Informação");
4. Criação de ações, que permitem disparar notificações, comandos remotos ou escalações com base nos eventos gerados.

Por sua vez, os *templates* são *hosts* abstratos, para os quais são configurados itens, *triggers* e ações. Ao associar um *host* a um *template*, este herda todas as suas configura-

ções. Os *hosts* também podem ser agrupados em uma entidade de associação (*Hostgroup*). Estas relações estão descritas na Figura 5.14.

5.4 Avaliação Experimental

A elasticidade permite que os usuários alterem rapidamente o número de VMs utilizadas de acordo com suas necessidades e de maneira oportuna, sendo uma das mais importantes características para o conceito de nuvem [21]. Assim, para avaliação da plataforma, será analisado o comportamento da solução proposta em um possível caso de uso envolvendo rápida elasticidade para criação de dezenas de máquinas virtuais em rajada.

Experimentos de abordagem semelhante foram conduzidos por outros pesquisadores, com intuito de avaliar CMPs abertas, tais como o OpenStack, o OpenNebula e o CloudStack [15][105]. Além do impacto ao adotar um sistema de arquivos em rede ou um sistema de mensageria para lidar com as requisições na plataforma Cloud.Jus, foram avaliados o consumo de CPU, os tempos de execução e o *throughput*, cuja métrica estabelece a quantidade de VMs/min implantadas no ambiente após a solicitação de criação de 50 instâncias em *tenants* atendidos pelos hipervisores VMware e Hyper-V. A seguir, estão descritos os cenários utilizados em cada um dos experimentos.

5.4.1 Cenários de Teste

Foram conduzidos 4 (quatro) conjuntos de experimentos. O objetivo foi avaliar o comportamento dos mecanismos de gerenciamento e a arquitetura da nuvem ao lidar com uma situação típica de estresse como, por exemplo, o atendimento de requisições em rajada para rápida implantação de máquinas virtuais. Em todos os experimentos foi utilizada uma instância padrão do CentOS 7.5, cuja imagem de sistema operacional possui 2.5GB.

Durante os experimentos, os *workers* acionam uma infraestrutura composta por 8 (oito) nodos do VMware ESXi 6.5 e 8 (oito) nodos do Microsoft Hyper-V 2012R2, com 128GB de memória cada, 2 processadores Intel Xeon 2.4Ghz com 6 núcleos de processamento cada, e *datastores* estão hospedados em um *storage* EMC VNX5400, com grupos de discos em RAID, de mesma configuração, acessados via protocolo *fibre-channel*. Após a plataforma despachar a execução das tarefas de implantação, a criação das VMs é conduzida pelo VMware vCenter e pelo Microsoft *Cluster*. A infraestrutura está posicionada de maneira distribuída nos três *datacenters* à disposição do STF e CNJ.

- Experimento 1: o intuito foi verificar o comportamento de cada nó *worker* ao rodar um ou dois processos para atender o *tenant* correspondente. Para isso, foram criadas 50 requisições simultâneas, 25 para cada *tenant* e utilizados os nodos *worker* 1

e *worker* 2. O Teste 1 utilizou o nó *worker* 1 rodando 2 processos, um atendendo ao *tenant* VMware e outro atentendo ao Hyper-V. Já o Teste 2 utilizou simultaneamente os nodos *worker* 1 e *worker* 2, cada um deles rodando um processo dedicado por *tenant*. Com isso pudemos avaliar se ao empregar múltiplos processos em cada nó ou utilizar nodos dedicados para cada *tenant* seria apresentada diferença significativa de desempenho. Neste experimento foram utilizados o sistema de arquivos, o repositório de mensagens e os nodos *worker* 1 e 2, que foram configurados com 8GB/RAM e 4vCPUs cada.

- Experimento 2: o intuito foi verificar o comportamento da plataforma ao lidar com 50 requisições, 25 para cada *tenant*, utilizando os nodos *worker* 1, 2 e 3, rodando 2 dois processos cada, sendo capazes, portanto, de atender simultaneamente as requisições destinadas ao Hyper-V e ao VMware. Assim, foi possível avaliar os ganhos de desempenho obtidos ao estabelecer 3 nodos *workers* para atender as requisições e também comparar o desempenho ao adotar uma abordagem de processamento com um sistema de mensageria, no caso o NATS, ao invés do sistema de arquivos anteriormente utilizado. Assim, neste experimento foram considerados 2 testes. O Teste 3 utilizou o sistema de arquivos como repositório de mensagens e os nodos *worker* 1, 2 e 3 foram configurados com 8GB/RAM e 4vCPUs. O Teste 4 utilizou o sistema de mensageria e os nodos *worker* 1, 2 e 3 mantiveram a mesma configuração.

- Experimento 3: o intuito foi verificar o comportamento da plataforma ao lidar com 50 requisições, 25 para cada *tenant*, utilizando os nodos *worker* 4, 5 e 6 rodando 2 dois processos, sendo capazes de atender tanto as requisições destinadas ao Hyper-V quanto ao VMware. Com isso, foi possível avaliar a variação de desempenho ao estabelecer o sistema de mensageria em uma configuração mais modesta para os nodos, com 1vCPU e 4GB/RAM.

- Experimento 4: o intuito foi verificar o comportamento da plataforma ao lidar com 50 requisições, 25 para cada *tenant*, utilizando os nodos *worker* 4, 5, 6, 7 e 8, rodando 2 dois processos cada e atendendo tanto as requisições destinadas ao Hyper-V quanto ao VMware. Com isso, foi possível avaliar os ganhos de desempenho obtidos ao estabelecer 1, 3 ou 5 nodos *workers* para atender as solicitações.

5.4.2 Resultados e Discussão

Os resultados obtidos foram consolidados na Tabela 5.1. Durante o protocolo de testes, cada processo de integração ao *middleware* rodando nos *workers* adicionou uma média de 8% de utilização da CPU ao nó correspondente. Não houve variação considerável no consumo de memória RAM, que permaneceu em média 60% livre nos *workers* operando

Tabela 5.1: Resultados dos Experimentos Conduzidos.

Experimento	**Tenant**	**Tempo de Execução (min)**	**Throughput (VMs/min)**	**Sumário**
Experimento 1	1-VMware	26min28seg	0,94	1 *worker*, 2 processos, 4vCPUs, sistema de arquivos
	2-Hyper-V	57min07seg	0,43	
	1-VMware	25min20seg	0,98	2 *workers*, 1 processo, 4vCPUs, sistema de arquivos
	2-Hyper-V	57min05seg	0,43	
Experimento 2	1-VMware	7min03seg	3,54	3 *workers*, 2 processos, 4vCPUs, sistema de arquivos
	2-Hyper-V	27min06seg	0,92	
	1-VMware	11min35seg	2,15	3 *workers*, 2 processos, 4vCPUs, NATS
	2-Hyper-V	12min0seg	2,08	
Experimento 3	1-VMware	16min49seg	1,48	1 *worker*, 2 processos, 1vCPU, NATS
	2-Hyper-V	29min01seg	0,86	
Experimento 4	1-VMware	12min04seg	2,07	3 *workers*, 2 processos, 1vCPU, NATS
	2-Hyper-V	12min32seg	1,99	
	1-VMware	10min15seg	2,43	5 *workers*, 2 processos, 1vCPU, NATS
	2-Hyper-V	7min01seg	3,57	

com 8GB e 40% livre nos *workers* operando com 4GB. No experimento 1, o processamento variou entre 66% e 78% nos testes realizados. Não houve diferença significativa no desempenho ao trabalhar com 2 processos por *worker* ou operar com *workers* dedicados por *tenant*.

Desta maneira, o Experimento 2 permitiu verificar que é possível obter uma estratégia de paralelismo mais adequada com 03 nodos, pois o ambiente VMware alcançou uma melhoria de desempenho cerca de 3,5 vezes superior ao ambiente com 1 nó apenas, enquanto o Hyper-V alcançou uma melhoria de aproximadamente 2 vezes sobre o número de VMs criadas por minuto, mesmo ao operar com sistema de arquivos, conforme Figura 5.15.

Além disso, observou-se que, ao operar com sistema de mensageria ao invés de sistema de arquivos, o tempo de execução foi reduzido em 66%, e os *tenants* VMware e Hyper-V obtiveram um desempenho mais equilibrado, respectivamente com 2,15 e 2,08 VMs/min, demonstrando a superioridade desta abordagem. Essa superioridade é tanto em termos de facilidade para implementar a escalabilidade da plataforma, desacoplando os produtores e consumidores, quanto para conferir maior fluidez às tarefas da plataforma, conforme resultado apresentado na Figura 5.16.

Outro fator que chamou a atenção ao adotar o sistema de mensageria foi observar que a média de processamento dos nodos foi reduzida do patamar máximo de quase 80% de utilização para apenas 27%. Assim, ainda há bastante margem para realizar uma otimização em termos de capacidade de processamento por meio de abordagens *multi-thread*, potencialmente extraindo melhor desempenho com a mesma quantidade de recursos alocados.

Considerando os resultados do Experimento 2, que utilizou nodos com 4vCPUs e 8GB de memória RAM, que a abordagem com sistema de mensageria reduziu bastante o *foot-*

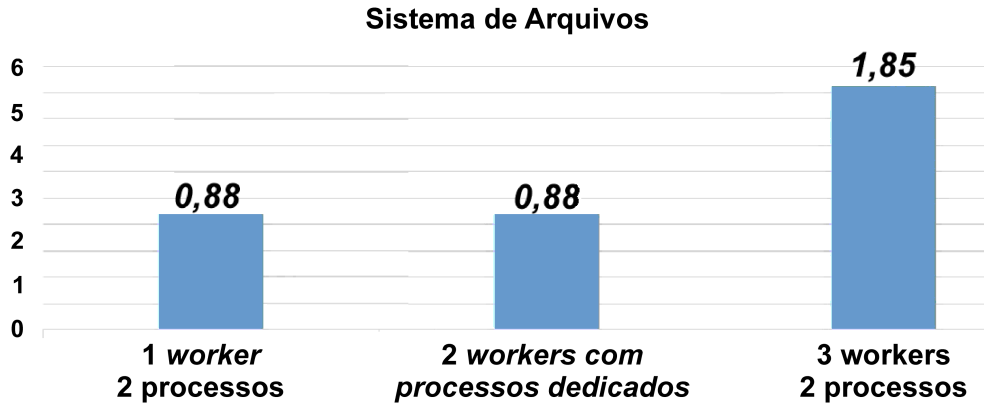

Figura 5.15: *Throughput* (VMs/min) com 1, 2 ou 3 nodos e Sistema de Arquivos.

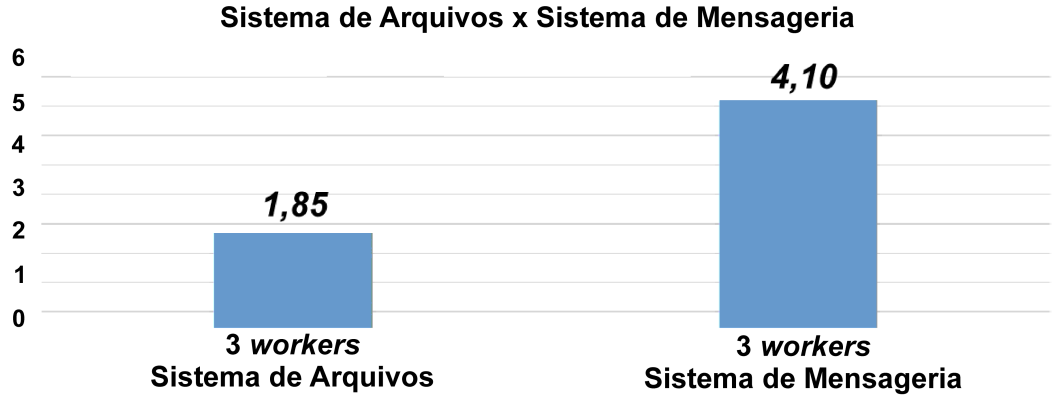

Figura 5.16: *Throughput* (VMs/min) com 3 nodos e Sistema de Arquivos ou Mensageria.

print de CPU, e que também não houve impacto ao consumo de memória, o Experimento 3 permitiu avaliar o comportamento da solução ao empregar nodos de configurações mais modestas. Assim, a configuração total de núcleos foi reduzida para 25% em relação a originalmente empregado, ou seja, de 12vCPUs para apenas 3vCPUs, 1 núcleo para cada nó. A memória foi reduzida à metade para cada nó, ou seja, de 8GB para 4GB.

Mesmo com a enorme economia de recursos, os resultados foram muito próximos aos obtidos no Experimento 2 para o ambiente VMware, acrescentando 5 minutos no tempo de execução, enquanto o Hyper-V apresentou uma degradação considerável, com aumento de 17 minutos no tempo de execução, conforme apresentado na Tabela 5.1. Isso ocorreu em função da limitação na configuração da quantidade de vCPUs imposta.

Dessa forma, o Experimento 4 foi conduzido para comparar o desempenho nas abordagens com 1, 3 e 5 nodos, mantendo apenas 1 núcleo de processamento em cada *worker*. A abordagem com 5 nodos se mostrou equilibrada ao lidar com os *tenants*, e permitiu a melhoria de desempenho do Hyper-V em relação ao VMware. O *throughput* com esta

abordagem atingiu 4,88 VMs/min, conforme descrito na Figura 5.17. Isso é bastante superior aos resultados obtidos com apenas 1 nó, levando cerca de 17% do tempo de execução original, que foi reduzido de aproximadamente 1h para apenas 10 minutos, mesmo com uso de 5vCPUs, ou seja, somente 1 núcleo a mais que no primeiro teste realizado, que contava com apenas um nó.

Diante do exposto, é importante destacar a viabilidade da plataforma ao lidar com requisições em rajada. De acordo com o número de usuários da plataforma e a quantidade de operações realizadas, entende-se que os *workers* podem ser escalados e terem a configuração ajustada para conferir melhor desempenho na realização das tarefas de gerenciamento da nuvem de acordo com a necessidade.

Figura 5.17: *Throughput* (VMs/min) com 1, 3 ou 5 nodos e Sistema de Mensageria.

5.5 Contribuição da Pesquisa

A plataforma desenvolvida está operando na infraestrutura computacional nos três *datacenters* à disposição do STF, ajudando a otimizar o provisionamento de infraestrutura. Antes de adotar a plataforma, a implantação de uma VM levava de 1 a 3 dias para ser concluída, dada a complexidade e diversas interações necessárias para que solicitação percorresse todos os silos de infraestrutura envolvidos.

Esta demora era em função da falta de ferramentas apropriadas para lidar com o ambiente de maneira integrada e da dependência de diversas equipes atuarem. Atualmente, a mesma atividade é gerenciada com os mecanismos automatizados da Cloud.Jus, pode ser realizada pela própria equipe de DevOps, liberando os recursos instanciados em aproximadamente 3 minutos. A arquitetura é escalável, de maneira que pode ser expandida para comportar o gerenciamento de IaaS em outros *Datacenters*.

Tabela 5.2: Avaliação das Características das Principais Contribuições.

	Acoplamento	**Provedores Suportados**	**APIs Suportadas**	**Integração**	**Monitoramento**
SDCon: Son e Buyya (2019)	Baseado em Componentes	OpenStack	OpenStack API, OpenDayLight API	Centrada no Provedor	Ceilometer
CLOUDIATOR: Baur e Domaschka (2016)	Baseado em Componentes	OpenStack	API própria	Centrada no Provedor	Componente próprio (AXE)
MCOF: Lu et al. (2017)	Baseado em Componentes	OpenStack	OpenStack API	Centrada no Provedor	Zabbix
FogBow: Brasileiro (2016)	Fortemente Acoplado	OpenStack, OpenNebula	OCCI	Centrada no Provedor	Descentralizado
DIRAC: Casaj et al. (2014)	Fortemente Acoplado	OpenStack, OpenNebula, CloudStack	OCCI, AWS	Centrada no Provedor	Descentralizado
Aurora: Wickboldt et al. (2014)	Fortemente Acoplado	Libvirt	API própria	Centrada no Cliente	Nagios/ FlexACMS
Cloud.jus: este trabalho	**Fracamente Acoplado**	**VMware, Hyper-V**	**API própria**	**Centrada no Cliente**	**Zabbix/ Grafana**

Nesse sentido, acredita-se que com uma arquitetura fracamente acoplada em nível de provedor de recursos e, além disso, características de integração e monitoramento centrados no cliente, foi possível confirmar a hipótese deste trabalho, isto é: "que as instituições possam se beneficiar ao empregar implementações semelhantes, no intuito de promover interesses comuns, tais como manutenção da segurança na distribuição geográfica de seus dados, maior redundância e tolerância a falhas e catástrofes, implementação de mecanismos para garantir a estratégia de continuidade de serviços e, futuramente, economia de recursos em larga escala, resguardando o interesse público e garantindo a qualidade na implementação de novos serviços".

Ao contrário de outras soluções que evitam realizar múltiplas integrações, cuja abordagem pode inviabilizar atualizações de versões, uma vez que o suporte aos hipervisores também passa a depender de evoluções nas bibliotecas ou APIs genéricas de integração, a exemplo de libvirt, libcloud e adaptações no uso de OCCI. A Cloud.Jus também oferece uma GUI, CLI e API que viabilizam uma abordagem de automatização do ciclo de vida das VMs e um monitoramento bastante flexível, suportado por sistemas populares e robustos: Zabbix e Grafana, que atuam de maneira associada e permitem tanto o monitoramento disponibilidade e desempenho quanto ações proativas e reativas para o restabelecimento de serviços, além da orientação a eventos, particularmente útil para conferir elasticidade, caso desejado.

Um típico caso de uso seria adotar uma plataforma de orquestração de contêineres, cujas VMs utilizadas para hospedar o *cluster* Kubernetes [6] podem ser associadas dina-

micamente ao ambiente conforme necessidade. Isso pode ser detectado por um evento associado a uma *trigger* no Zabbix como, por exemplo, atingir 70% de processamento do *cluster*. Isso automatizaria uma ação via API para implantação de VMs na plataforma Cloud.Jus e ingresso ao *cluster* Kubernetes. Deste modo, as aplicações legadas poderiam ser migradas pelos órgãos para uma abordagem em contêineres, utilizando *Docker* [5] e, posteriormente, hospedadas em *clusters* distribuídos entre os *datacenters* da comunidade federada.

Capítulo 6

Conclusão

Elaborar uma abordagem independente de arquitetura geralmente implica em riscos relacionados à escalabilidade e controle. Em arquiteturas fracamente acopladas, a coordenação pode ser flexibilizada uma vez que um único agente pode satisfazer todos os objetivos ou boa parte deles. Contudo, a abordagem da Cloud.Jus fornece uma opção de baixo acoplamento que mitiga os riscos de perda de escalabilidade e controle, contribuindo na área de interoperabilidade entre nuvens.

A plataforma é capaz de conciliar as particularidades de vários provedores por meio de um *middleware* que suporta os dois principais hipervisores corporativos do mercado (VMware, Hyper-V), mantendo as características de suporte que as tornam mais utilizadas pelos órgãos, minimizando assim o esforço de reutilização e integração. Além disso, foi associada de maneira coesa aos serviços e ferramentas de monitoramento disponíveis, tornando-a uma alternativa possivelmente atraente para instituições que desejem adotar o modelo de computação em nuvem sem passar por uma agressiva disruptura com as aplicações legadas.

Este trabalho resultou em dois artigos que tratam especificamente da arquitetura proposta [128][129], respectivamente publicados em conferências de estrato A2 (ACM CLOSER 2019) e B1 (ACM DG.o 2019), segundo o Qualis/CAPES. Durante o período de realização do Mestrado, além destes dois artigos específicos sobre a plataforma, foram realizadas outras quatro publicações na área de sistemas distribuídos, computação em nuvem e arquitetura de sistemas, em conferências e revistas com Qualis/CAPES entre A1 e B2[130][131][132][133].

Como trabalhos futuros, pretende-se englobar a integração com sistemas virtualizados de rede para, assim, adicionar funções como balanceamento de carga e regras de *firewall*, acionadas diretamente pela plataforma. Além disso, espera-se concluir a integração com o hipervisor Oracle VM e realizar otimizações nas comunicações entre processos de sincronização e atualização do repositório de estados. Futuramente, poderá haver a delegação de

objetivos, de modo a paralelizar as contribuições dos órgãos interessados e deliberar sobre os próximos passos de desenvolvimento, viabilizando a infraestrutura necessária para um sistema distribuído de caráter mais abrangente, fomentando a possibilidade de adesão em maior escala.

Referências

[1] Loeffler, Bill: *Cloud Computing - What is Infrastructure as a Service*. TechNet Magazine, 10, 2011. `https://social.technet.microsoft.com/wiki/contents/articles/4633.what-is-infrastructure-as-a-service.aspx`. x, 14, 15

[2] Vaquero, Luis M., Luis Rodero-Merino, Juan Caceres e Maik Lindner: *A Break in the Clouds: Towards a Cloud Definition*. Computer Communication Review, 39(1):50–55, dec 2008, ISSN 0146-4833. x, 16, 17, 25

[3] Toosi, Adel Nadjaran, Rodrigo N. Calheiros e Rajkumar Buyya: *Interconnected cloud computing environments: Challenges, taxonomy, and survey*. ACM Computing Surveys (CSUR), 47(1):1–47, ayMay 2014, ISSN 0360-0300. x, 19, 20, 49, 59

[4] Buyya, Rajkumar, James Broberg e Andrzej M. Goscinski: *Cloud Computing Principles and Paradigms*. Wiley Publishing, 2011, ISBN 9780470887998. x, 22, 47

[5] Docker: *What is docker?* `https://www.docker.com/what-docker`, 2018. x, 23, 24, 80

[6] Red Hat: *What is kubernetes*. https://www.redhat.com/pt-br/topics/containers/what-is-kubernetes, 2018. x, 24, 25, 79

[7] Quevedo, Waldemar: *Introduction to NATS*. Apress, 2018. `https://doi.org/10.1007/978-1-4842-3570-6_1`. x, 25, 26, 29, 30

[8] Pivotal Software: *RabbitMQ - Messaging that just works*. https://www.rabbitmq.com, 2018. x, 26, 27, 34, 41, 43

[9] Kreps, Jay, Neha Narkhede e Jun Rao: *Kafka: a Distributed Messaging System for Log Processing*. NetDB, 2011. x, 26, 28, 29, 43

[10] Jackson, Kevin e Cody Bunch: *OpenStack Cloud Computing Cookbook - Second Edition*. Packt Publishing, 2013, ISBN 9781782167587. x, 7, 15, 33, 34, 41

[11] Lima, Stanley, Álvaro Rocha e Licinio Roque: *An overview of OpenStack architecture: a message queuing services node*. Em *Cluster Computing*. Springer US, 2017. x, 32, 35, 36

[12] Milojičić, Dejan, Ignacio M. Llorente e Rubén S. Montero: *OpenNebula: A cloud management tool*, 2011. ISSN 1089-7801. x, 7, 15, 36, 37, 54

[13] Apache Foundation: *CloudStack - Open Source Cloud Computing*, 2018. `https://cloudstack.apache.org`. x, 15, 32, 37, 38, 39

[14] Eucalyptus Systems, Inc: *Eucalyptus - Open source software for building AWS-compatible private and hybrid clouds*, 2018. `https://www.eucalyptus.cloud`. x, 15, 32, 39, 40

[15] Kostantos, Konstantinos, Andrew Kapsalis, Dimosthenis Kyriazis, Marinos Themistocleous e Paulo Rupino Cunha: *Open-Source IaaS Fit for Purpose: A Comparison between OpenNebula and OpenStack*. International Journal of Electronic Business Management (IJEBM), 11(3):192–201, 2013. x, 37, 43, 44, 74

[16] Lu, Ming, Lijuan Wang, Youyan Wang, Zhicheng Fan, Yatong Feng, Xiaodong Liu e Xiaofang Zhao: *An orchestration framework for a global multi-cloud*. Em *Proceedings of the 2018 Artificial Intelligence and Cloud Computing Conference*, AICCC '18, páginas 58–62, New York, NY, USA, 2018. ACM, ISBN 978-1-4503-6623-6. `http://doi.acm.org/10.1145/3299819.3299823`. xi, 49, 53, 54

[17] Edwards, D: *Introducing DevOps to the Traditional Enterprise*. InfoQueue / eMag Issue, 14, jun 2014. 1, 24, 31, 57

[18] Lui, A. e I. Gordon: *Service Based Integration*. https://msdn.microsoft.com/library/aa480046.aspx, 2005. 1

[19] Aceto, Giuseppe, Alessio Botta, Walter de Donato e Antonio Pescapè: *Cloud monitoring: A survey*. Computer Networks, 57(9):2093–2115, jun 2013. 1

[20] Murugesan, San e Irena Bojanova: *Encyclopedia of Cloud Computing*. Wiley-IEEE Press, 2016, ISBN 978-1-118-82197-8. 1

[21] Buyya, Rajkumar, Christian Vecchiola e S. Thamarai Selvi: *Mastering Cloud Computing: Foundations and Applications Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, t1st edição, 2013. 9780124095397, 9780124114548. 1, 12, 14, 33, 74

[22] Mell, P. e T. Grance: *The NIST Definition of Cloud Computing*. Relatório Técnico, NIST, National Institute of Standards and Technology, Gaithersburg, MD, United States, 2011. 1, 3, 4, 12, 13, 14, 15

[23] Lu, Wei, Hongqiang Fang e Zuqing Zhu: *AI-assisted resource advertising and pricing to realize distributed tenant-driven virtual network slicing in inter-DC optical networks*. Em *2018 International Conference on Optical Network Design and Modeling (ONDM)*. IEEE, may 2018. 2

[24] Ahn, J., C.H. Park, J. Huh, J. Lewis e M. Fowler: *Microservices*. Em *Proceedings of the 47th International Symposium on Microarchitecture*, 2014. `http://martinfowler.com/articles/microservices.html`. 3

[25] Mezgár, István e Ursula Rauschecker: *The challenge of networked enterprises for cloud computing interoperability*. Computers in Industry, 65(4):657–674, may 2014. `https://doi.org/10.1016/j.compind.2014.01.017`. 4, 6

[26] TRF4: *Lava Jato - 645 processos em três anos no TRF4*. https://www.trf4.jus.br/trf4/controlador.php?acao=noticia_visualizar&id_noticia=12712, 2017. Tribunal Regional Federal da 4a Região. 4

[27] VMware: *vSphere ESXi Hypervisor*. https://www.vmware.com/br/products/esxi-and-esx.html, 2018. 5, 34

[28] Microsoft: *Hyper-V Technology Overview*. https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview, 2016. 5, 34

[29] Mesa da Câmara dos Deputados e Senado Federal: *Emenda Constitucional n. 95*, 2016. http://www.planalto.gov.br/ccivil_03/constituicao/emendas/emc/emc95.htm. 5

[30] Brasileiro, F., G. Silva, F. Araújo, M. Nóbrega, I. Silva e G. Rocha: *Fogbow: A middleware for the federation of iaas clouds*. Em *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, páginas 531–534, may 2016. https://ieeexplore.ieee.org/document/7515732. 6, 7, 49

[31] Bégin, Marc Elian, Bob Jones, James Casey, Erwin Laure, Francois Grey, Cal Loomis e Rolf Kubli: *An EGEE Comparative Study: Grids and Clouds evolution or revolution?*, 2008. https://edms.cern.ch/document/925013/4. 6

[32] OCCI: *Open Cloud Computing Interface*. http://occi-wg.org/, 2018. 6, 46

[33] Baur, D., D. Seybold, F. Griesinger, A. Tsitsipas, C. B. Hauser e J. Domaschka: *Cloud orchestration features: Are tools fit for purpose?* Em *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, páginas 95–101, dec 2015. 6, 7, 15

[34] Brasileiro, Francisco, Jose Luis Vivas, Giovanni Farias Da Silva, Daniele Lezzi, Carlos Diaz, Rosa M. Badia, Miguel Caballer e Ignacio Blanquer: *Flexible federation of cloud providers: The EUBrazil cloud connect approach*. Em *Proceedings - IEEE 30th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2016*, 2016, ISBN 9781509018574. 7, 49

[35] Slawik, Mathias, Christophe Blanchet, Yuri Demchenko, Fatih Turkmen, Alexy Ilyushkin, Cees De Laat e Charles Loomis: *CYCLONE: The multi-cloud middleware stack for application deployment and management*. Em *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, páginas 347–352, 2017, ISBN 9781538606926. 7

[36] Parmar, Hiren e Tushar Champaneria: *Comparative Study of Open Nebula, Eucalyptus, Open Stack and Cloud Stack*. International Journal of Advanced Research in Computer Science and Software Engineering, 4(2):991–996, 2014. 7, 15, 41

[37] Wickboldt, Juliano Araujo, Rafael Pereira Esteves, Márcio Barbosa de Carvalho e Lisandro Zambenedetti Granville: *Resource management in iaas cloud platforms made flexible through programmability*. Computer Networks, 68:54–70, aug

2014, ISSN 1389-1286. http://www.sciencedirect.com/science/article/pii/S138912861400084X, Communications and Networking in the Cloud. 7, 49, 51

[38] Linux Foundation: *LibVirt - Virtualization API.* https://libvirt.org/drivers.html, 2018. 7, 56

[39] Apache Foundation: *jClouds - The Java Multi-Cloud Toolkit.* http://jclouds.apache.org/, 2018. 7

[40] Apache Foundation: *LibCloud - A Python library for interacting with many of the popular cloud service providers using a unified API.* https://libcloud.apache.org, 2018. 7

[41] Turchetti, Gustavo Heimovski Rogerio, Juliano Wickboldt, Lisandro Zambenedetti Granville e Elias Duarte Júnior: *Alta disponibilidade de um gerenciador de nuvem iaas baseada em replicação: Experiência e resultados.* Em *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 15–26, 2017. 7, 49, 51

[42] Baur, Daniel e Jorg Domaschka: *Experiences from building a cross-cloud orchestration tool.* Em *Proceedings of the 3rd Workshop on Cross-Cloud Infrastructures and Platforms*. ACM Press, 2016. 7, 49, 52

[43] Parkhill, Douglas F.: *The Challenge of the Computer Utility.* Addison-Wesley Pub. Co., 1966, ISBN 0201057204. 11

[44] Colman-Meixner, Carlos, Chris Develder, Massimo Tornatore e Biswanath Mukherjee: *A survey on resiliency techniques in cloud computing infrastructures and applications.* IEEE Communications Surveys and Tutorials, 18(3):2244–2281, 2016, ISSN 1553-877X. 11

[45] Han, Lei: *Market Acceptance of Cloud Computing - An Empirical Analysis of Market Structure, Price Models and Service Requirements.* Tese de Doutoramento, University of Bayreuth, Masterarbeit, Germany, 2009. http://nbn-resolving.de/urn:nbn:de:bvb:703-opus-5530. 11

[46] Madhavaiah, C., Irfan Bashir e Syed Irfan Shafi: *Defining cloud computing in business perspective: A review of research.* Vision: The Journal of Business Perspective, 16(3):163–173, sep 2012. 12

[47] Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin e Matei Zaharia: *Above the clouds: A berkeley view of cloud computing.* Relatório Técnico, UCLA,University of California, Berkeley, CA, 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf., Tecnical Report No. UCB/EECS-2009-28. 12

[48] Foster, I., Y. Zhao, I. Raicu e S. Lu: *Cloud Computing and Grid Computing 360-Degree Compared.* Em *Grid Computing Environments Workshop*, páginas 1–10, nov 2008. 12, 31

[49] Sosinsky, Barrie: *Cloud Computing Bible*, volume 762. John Wiley & Sons, 2010. 12, 25, 31, 33

[50] Tanenbaum, A.S. e M. Van Steen: *Sistemas Distribuídos: Princípios e paradigmas*. Pearson Education, 2007, ISBN 9788576051428. `https://books.google.com.br/books?id=r2SGPgAACAAJ`. 12

[51] L. Fang, T. Jin, M. Jian B. Robert L.B. John Messina D. Leaf: *Cloud Computing Reference Architecture. National Institute of Standards and Technology (NIST)*. NIST Special Publication, 2011. 13

[52] Bermbach, D., T. Kurze e S. Tai: *Cloud federation: Effects of federated compute resources on quality of service and cost*. Em *2013 IEEE International Conference on Cloud Engineering (IC2E)*, páginas 31–37, mar 2013. 13

[53] Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg e Ivona Brandic: *Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility*. Future Generation Computer Systems, 25(6):599–616, jun 2009, ISSN 0167-739X. 14

[54] Sotomayor, B, R Santiago Montero, I Martín Llorente e I Foster: *Virtual Infrastructure Management in Private and Hybrid Clouds*. IEEE Internet Computing, 13:5, 2009. 14, 31, 33

[55] Amazon: *Computação em nuvem com a Amazon Web Services*. https://aws.amazon.com/pt/what-is-aws/, 2018. 15, 19

[56] Microsoft: *O que é o Azure?* https://azure.microsoft.com/pt-br/overview/what-is-azure/, 2018. 15, 19, 22

[57] Google, Inc.: *Google Cloud: Produtos e Serviços*. https://cloud.google.com/products/, 2018. 15, 19

[58] VMware: *VMware NSX*. https://www.vmware.com/br/products/nsx.html, 2018. 15, 42

[59] Llorente, Ignacio M.: *Eucalyptus, CloudStack, OpenStack and OpenNebula: A Tale of Two Cloud Models*. `https://opennebula.org/eucalyptus-cloudstack-openstack-and-opennebula-a-tale-of-two-cloud-models/`, 2013. 15, 36, 41, 58

[60] Bittman, Thomas J. and Margevicius, Mark A. and Dawson, Philip: *Magic Quadrant for x86 Server Virtualization Infrastructure*. Relatório Técnico, Gartner Research, 2014. 17, 22

[61] Gartner Group: *Gartner Magic Quadrant*. `https://www.gartner.com/en/research/methodologies/magic-quadrants-research`, 2018. 18

[62] Jamsa, Kris: *Cloud Computing - SaaS, PaaS, IaaS, Virtualization, Business Models, Mobile, Security and More*. Jones and Bartlett Learning, 2012, ISBN 1449647391. 18, 19, 21, 59

[63] Amazon: *Amazon Elastic Compute Cloud - EC2*, 2018. 18

[64] Foundation, Linux: *Xen - The Power of Virtualization Everywhere*. https://xenproject.org, 2018. 19, 23, 34

[65] Open Virtualization Alliance - OVA: *KVM (Kernel-based Virtual Machine) - full virtualization solution for Linux on x86 hardware*. `https://www.linux-kvm.org/page/Main_Page`, 2018. 19, 34

[66] Amazon: *Amazon Elastic Block Store - armazenamento em bloco persistente para o Amazon EC2*. https://aws.amazon.com/pt/ebs/, 2018. 19

[67] Amazon: *Amazon Simple Storage Service (Amazon S3)*. https://aws.amazon.com/pt/s3/, 2018. 19

[68] Amazon: *Amazon Elastic Container Service - Aplicativos Conteinerizados no ambiente de produção*. https://aws.amazon.com/pt/ecs/, 2018. 19

[69] Microsoft: *Azure Virtual Machines - Crie máquinas virtuais do Linux e do Windows em segundos*. https://azure.microsoft.com/pt-br/services/virtual-machines/, 2018. 19

[70] Google, Inc.: *Google Compute Engine - Máquinas Virtuais escalonáveis de alto desempenho*. https://cloud.google.com/compute/, 2018. 19

[71] Google, Inc.: *Google App Engine*. https://cloud.google.com/appengine/docs/, 2018. 19

[72] Google, Inc.: *Google Cloud Storage*. https://cloud.google.com/storage, 2018. 19

[73] Google, Inc.: *Kubernetes Engine - Uma maneira confiável, eficiente e segura de executar clusters do Kubernetes*. https://cloud.google.com/kubernetes-engine/, 2018. 19, 24

[74] Grozev, Nikolay e Rajkumar Buyya: *Inter-Cloud architectures and application brokering: Taxonomy and survey*. Software - Practice and Experience, December:369–390, 2014, ISSN 0038-0644. 19, 20, 70

[75] Celesti, Antonio, Francesco Tusa, Massimo Villari, Antonio Puliafito, Contrada Dio e S Agata: *How to enhance cloud architectures to enable cross-federation*. Em *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE, jul 2010. 19, 47

[76] Assis, M.R.M. e L.F. Bittencourt: *A survey on cloud federation architectures: Identifying functional and non-functional properties*. Journal of Network and Computer Applications, 72:51–71, sep 2016. 20

[77] Puthal, Deepak, B.P.S. Sahoo, Sambit Mishra e Satyabrata Swain: *Cloud computing features, issues, and challenges: A big picture*. Em *2015 International Conference on Computational Intelligence & Networks (CINE)*, volume 00, páginas 116–123, jan 2015. 21

[78] Saldanha, Hugo V.: *BionimbuZ: uma arquitetura de federação de nuvens computacionais híbrida para a execução de workflows de bioinformática*. Tese de Mestrado, Universidade de Brasília, Departamento de Ciência da Computação, 2012. 21, 47

[79] Gu, Zonghua e Qingling Zhao: *A State-of-the-Art Survey on Real-Time Issues in Embedded Systems Virtualization*. Journal of Software Engineering and Applications, 2012, ISSN 1945-3116. 21

[80] Sahoo, Jyotiprakash, Subasish Mohapatra e Radha Lath: *Virtualization: A survey on concepts, taxonomy and associated security issues*. Em *2nd International Conference on Computer and Network Technology, ICCNT 2010*, 2010, ISBN 9780769540429. 21, 23

[81] Tarasov, V, D Hildebrand, G. Kuenning e B E Zadok: *Virtual machine workloads: The case for new NAS benchmarks,*. Em FAST, Proc. 11th USENIX Conf. (editor): *The evolution of linux containers and their future*, páginas 307–320, 2013. 23

[82] Dua, R, A R Raja e D Kakadia.: *Virtualization vs containerization to support PaaS*. IEEE International Conference on Cloud Engineering, páginas 610–614, mar 2014. 24

[83] Coulouris, G., J. Dollimore, T. Kindberg e G. Blair: *Sistemas Distribuídos - 5a Edição: Conceitos e Projeto*. Bookman Editora, 2013, ISBN 9788582600542. `https://books.google.com.br/books?id=6WU3AgAAQBAJ`. 25

[84] Dobbelaere, Philippe e Kyumars Sheykh Esmaili: *Kafka versus rabbitmq - a comparative study of two industry reference pubsub implementations*. Em *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, DEBS '17, páginas 227–238, New York, NY, USA, 2017. ACM, ISBN 978-1-4503-5065-5. `http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/3093742.3093908`. 25, 27, 28

[85] Collison, Derek: *NATS: The Importance of Messaging*. https://nats-io.github.io/docs/, 2018. 26, 30, 68

[86] Apache Foundation: *Apache Zookeeper*. https://zookeeper.apache.org/, 2018. 28

[87] Sefraoui, Omar, Mohammed Aissaoui e Mohsine Eleuldj: *Openstack: Toward an open-source solution for cloud computing*. International Journal of Computer Applications, 55(3):38–42, oct 2012. Full text available. 31

[88] Lynn, Theo, Graham Hunt, David Corcoran, John Morrison e Philip Healy: *A comparative study of current open-source infrastructure as a service frameworks*. Em *Proceedings of the 5th International Conference on Cloud Computing and Services Science*. SciTePress - Science and Technology Publications, 2015, ISBN 9789897581045. 32, 41

[89] Bhardwaj, Tushar, Mohit Kumar e S. C. Sharma: *Megh: A Private Cloud Provisioning Various IaaS and SaaS*. Em *Advances in Intelligent Systems and Computing*, páginas 485–494. Springer Singapore, ovNov 2017. 32

[90] Casaj, Adrian, Ricardo Graciani Diaz, Andrei Tsaregorodtsev, Víctor Méndez Muñoz, Adrian Casajús Ramo, Ricardo Graciani Diaz e Andrei Tsaregorodtsev: *Cloud governance by a credit model with DIRAC*. Em *Proceedings of the 4th International Conference on Cloud Computing and Services Science*, páginas 679–686. SciTePress - Science and Technology Publications, 2014, ISBN 9789897580192. 32, 51

[91] Endo, Patricia Takako, Glauco Estácio Gonçalves, Djamel Sadok e Judith Kelner: *A Survey on Open-source Cloud Computing Solutions*. Brazilian Symposium on Computer Networks and Distributed Systems, páginas 3–16, 2010, ISSN 0140-3664. 32

[92] Maron, Carlos Alberto Franco, Dalvan Griebler, Claudio Schepke e Luiz Gustavo Fernandes: *Desempenho De Openstack E Opennebula Em Estações De Trabalho: Uma Avaliação Com Microbenchmarks E Npb*. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação, 1(6), 2016, ISSN 2446-7634. `http://revistas.setrem.com.br/index.php/reabtic/article/view/186`. 32

[93] Mohammed, Bashir e Mariam Kiran: *Analysis of Cloud Test Beds Using OpenSource Solutions*. Em *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015 and 2015 International Conference on Open and Big Data, OBD 2015*, 2015, ISBN 9781467381031. 32

[94] Couto, Rodrigo S., Hugo Sadok, Pedro Cruz, Felipe F. da Silva, Tatiana Sciammarella, Miguel Elias M. Campista, Luís Henrique M.K. Costa, Pedro B. Velloso e Marcelo G. Rubinstein: *Building an IaaS cloud with droplets: a collaborative experience with OpenStack*. Journal of Network and Computer Applications, 117(June):59–71, sep 2018, ISSN 1095-8592. 32

[95] Sharma, Upendra, Prashant Shenoy e Sambit Sahu: *A Flexible Elastic Control Plane for Private Clouds*. Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference on - CAC '13, 2013. 32

[96] Razavi, Kaveh, Stefania Costache, Andrea Gardiman, Kees Verstoep e Thilo Kielmann: *Scaling VM Deployment in an Open Source Cloud Stack*. Proceedings of the 6th Workshop on Scientific Cloud Computing - ScienceCloud '15, páginas 3–10, 2015, ISSN 1364-548X. `http://dl.acm.org/citation.cfm?id=2755644.2755645{%}5Cnhttp://dl.acm.org/citation.cfm?doid=2755644.2755645`. 32

[97] Singh, Sukhpal e Inderveer Chana: *A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges*. Journal of Grid Computing, 14(2):217–264, 2016, ISSN 1570-7873. 33

[98] Cunha Rodrigues, Guilherme, Rodrigo N. Calheiros, Vinícius Tavares Guimarães, Glederson Lessa Santos, Márcio Barbosa Carvalho, Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco e Rajkumar Buyya: *Monitoring of cloud computing environments*. Em *Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16*, 2016, ISBN 9781450337397. 33, 52

[99] Narayana, K.E. e K. Jayashree: *A overview on cloud computing platforms and issues*. International Journal of Advanced Research in Computer Science and Software Engineering, 7(1):238–242, jan 2017. 36

[100] Amazon: *Amazon Simple Queue Service*. https://aws.amazon.com/pt/sqs/, 2018. 40, 43

[101] Martin, Robert: *Design Principles and Design Patterns*. `http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf`, 2000. 41

[102] Roveda, Demétrius, Adriano Vogel e Dalvan Griebler: *Understanding, Discussing and Analyzing the OpenNebula and the OpenStack IaaS Management Layers*, 2016. 41

[103] Zabbix: *Zabbix - Solutions for any kind of IT infrastructure, services, applications and resources*. https://www.zabbix.com/, 2018. 43, 62

[104] Caron, Eddy, Lamiel Toch e Jonathan Rouzaud-Cornabas: *Comparison on OpenStack and OpenNebula Performance to Improve Multi-Cloud Architecture on Cosmological Simulation Use Case*. Hyper Articles en Ligne - The HAL OpenArchive, dezembro 2013. `https://hal.inria.fr/hal-00916908`. 44

[105] Chilipirea, Cristian, Ghita Laurentiu, Mirona Popescu, Sorin Radoveneanu, Vladimir Cernov e Ciprian Dobre: *A comparison of private cloud systems*. Proceedings - IEEE 30th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2016, 1(March):139–143, 2016. 44, 74

[106] Petcu, Dana, Ciprian Cr e Massimiliano Rak: *On the Interoperability in Multiple Clouds*. Em *Proceedings of the 3rd International Conference on Cloud Computing and Services Science*. SciTePress - Science and Technology Publications, 2013, ISBN 9789898425522. 45, 49

[107] Demchenko, Yuri, Marc X. Makkes, Rudolf Strijkers e Cees De Laat: *Intercloud Architecture for interoperability and integration*. Em *CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science*, 2012, ISBN 9781467345095. `https://www.researchgate.net/publication/273945605`. 46

[108] Breitenbücher, Uwe, Tobias Binz, Oliver Kopp, Frank Leymann e Matthias Wieland: *Context-aware cloud application management*. Em *Proceedings of the 4th International Conference on Cloud Computing and Services Science*. SciTePress - Science and Technology Publications, 2014. 46, 47

[109] Joukov, Nikolai e Vladislav Shorokhov: *Cloud interoperability via quick enterprise applications re-builds*. Em *Proceedings of the 3rd International Conference on Cloud Computing and Services Science*. SciTePress - Science and Technology Publications, 2013, ISBN 9789898565525. 47, 57

[110] Saatkamp, Karoline, Uwe Breitenbücher, Oliver Kopp e Frank Leymann: *Topology Splitting and Matching for Multi-Cloud Deployments.* Em *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, 2017, ISBN 978-989-758-243-1. 47, 49

[111] Parpaillon, Jean, Philippe Merle, Olivier Barais, Marc Dutoo e Fawaz Paraiso: *OCCIware - A formal and tooled framework for managing everything as a service.* Em *CEUR Workshop Proceedings*, 2015. 49

[112] Šustr, Zdeněk, Diego Scardaci, Jiří Sitera, Boris Parák e Víctor Méndez Muñoz: *Easing Scientific Computing and Federated Management in the Cloud with OCCI.* Em *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, volume 2, páginas 347–354. SciTePress, 2016, ISBN 9897581820. 49

[113] Erbel, Johannes, Fabian Korte e Jens Grabowski: *Proceedings of the 8th International Conference on Cloud Computing and Services Science.* SciTePress, 2018, ISBN 9897582959. 49

[114] Venticinque, Salvatore, Rocco Aversa e Beniamino Di Martino: *Agent-based cloud provisioning and management - design and prototypal implementation.* Em *Proceedings of the 1st International Conference on Cloud Computing and Services Science.* SciTePress - Science and Technology Publications, 2011, ISBN 9789898425522. 49

[115] Buyya, Rajkumar e Jungmin Son: *SDCon - Integrated Control Platform for Software-Defined Clouds.* IEEE Transactions on Parallel and Distributed Systems, 30:230–240, jan 2019. 49, 54

[116] Falcão, Eduardo De Lucena, Francisco Brasileiro, Andrey Brito e José Luis Vivas: *Enhancing P2P Cooperation through Transitive Indirect Reciprocity.* Em *Proceedings - 2016 IEEE 36th International Conference on Distributed Computing Systems Workshops, ICDCSW 2016*, 2016, ISBN 9781509014828. 49

[117] Fiore, Sandro, Marco Mancini, Donatello Elia, Paola Nassisi, Francisco Vilar Brasileiro e Ignacio Blanquer: *Big data analytics for climate change and biodiversity in the EUBrazilCC federated cloud infrastructure.* Em *Proceedings of the 12th ACM International Conference on Computing Frontiers - CF '15*, 2015, ISBN 9781450333580. 49

[118] XMPP Standards Foudation: *Extensible Messaging and Presence Protocol (XMPP) - open XML technology for real-time communication*, 2018. `https://xmpp.org/about/`. 50

[119] Schneider, Fabian, Marcus Brunner, Juliano Araujo, Wickboldt Lisandro, Zambenedetti Granville e Dominique Dudkowski: *Rethinking cloud platforms - Network-aware flexible resource allocation in IaaS clouds.* Em *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM2013) - Mini-Conference*, 2013. `https://www.researchgate.net/publication/261116671`. 51

[120] LF Networking: *Opendaylight - Platform Overview.* https://www.opendaylight.org/what-we-do/odl-platform-overview, 2018. 54

[121] Breitenbücher, Uwe, Tobias Binz, Oliver Kopp, Frank Leymann e Johannes Wettinger: *A Modelling Concept to Integrate Declarative and Imperative Cloud Application Provisioning Technologies*. Em *Proceedings of the 5th International Conference on Cloud Computing and Services Science*, 2015, ISBN 978-989-758-104-5. 57

[122] Sessions, Roger: *Um melhor caminho para arquiteturas corporativas*, 2006. `http://msdn.microsoft.com/pt-br/library/aa479371.aspx`. 57

[123] Bosch, Jan, Morven Gentleman, Christine Hofmeister e Juha Kuusela (editores): *Software Architecture*. Springer US, 2002. 57

[124] Saffer, Dan: *Designing for interaction: creating smart applications and clever devices*. Berkeley: Aiga, 4:5, 2009. 59

[125] PHP IPAM: *phpIPAM – Open source IP address management*. https://phpipam.net, 2018. 62, 71

[126] Grafana Labs: *Grafana - The analytics platform for all your metrics*. https://grafana.com/grafana, 2018. 62

[127] Dukarić, Robert e Matjaž B. Jurič: *A Taxonomy and Survey of Infrastructure-as-a-Service Systems*. Lecture Notes on Information Theory, 2013, ISSN 2301-3788. 63, 71, 73

[128] Castro, Klayton, Gabriel Macedo, Leonardo Reboucas e Aleteia Araujo: *Cloud.jus: Architecture for provisioning infrastructure as a service in the government sector*. Em *Proceedings of the 9th International Conference on Cloud Computing and Services Science, CLOSER 2019, Heraklion, Crete, Greece, May 2-4, 2019.*, páginas 412–421, 2019. `https://doi.org/10.5220/0007731804120421`. 81

[129] Castro, Klayton, Edna D. Canedo e Aleteia P.F. Araujo: *A feasible community cloud architecture for provisioning infrastructure as a service in the government sector*. Em *20th Annual International Conference on Digital Government Research*, dg.o 2019, páginas 35–40, New York, NY, USA, 2019. ACM, ISBN 978-1-4503-7204-6. `http://doi.acm.org/10.1145/3325112.3325229`. 81

[130] Hondo, Fernanda, Polyane Wercelens, Waldeyr da Silva, Iasmini Lima, Klayton Castro, Ingrid Santana, Gabriel Araujo, Aleteia Araujo, Maria Emília Walter e Maristela Holanda: *Nosql databases for management of bioinformatics workflows*. Em *32nd Brazilian Symposium on Databases, SBBD 2017, Uberlandia, MG, Brazil, October 2-5, 2017*, páginas 2139–2144, 2017. `https://doi.org/10.1109/BIBM.2017.8217989`. 81

[131] Almeida, Rodrigo F., Waldeyr Mendes Cordeiro da Silva, Klayton Castro, Aletéia Patrícia Favacho de Araújo, Maria Emilia Telles Walter, Sérgio Lifschitz e Maristela Holanda: *Managing data provenance for bioinformatics workflows using aprovbio*. I. J. Computational Biology and Drug Design, 12(2):153–170, 2019. `https://doi.org/10.1504/IJCBDD.2019.099761`. 81

[132] Hondo, Fernanda, Polyane Wercelens, Waldeyr Mendes Cordeiro da Silva, Klayton Castro, Ingrid Santana, Maria Emilia Telles Walter, Aletéia Patrícia Favacho de Araújo, Maristela Holanda e Sérgio Lifschitz: *Data provenance management for bioinformatics workflows using nosql database systems in a cloud computing environment*. Em *2017 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2017, Kansas City, MO, USA, November 13-16, 2017*, páginas 1929–1934, 2017. https://doi.org/10.1109/BIBM.2017.8217954. 81

[133] Almeida, Rodrigo F., Waldeyr Mendes Cordeiro da Silva, Klayton Castro, Maria Emilia Telles Walter, Aletéia Patrícia Favacho de Araújo, Maristela Holanda e Sérgio Lifschitz: *Aprovbio: An architecture for data provenance in bioinformatics workflows using graph database*. Em *2017 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2017, Kansas City, MO, USA, November 13-16, 2017*, páginas 2139–2144, 2017. https://doi.org/10.1109/BIBM.2017.8217989. 81