Sobre o Uso de Dispositivos de Alta Granularidade, Alta Volatilidade e Alta Dispersão em *Just in Time Clouds*

Rostand Costa[1 2], Diénert Vieira[2], Francisco Brasileiro[1], Dênio Mariz[2], Guido Lemos[2]

[1] Universidade Federal de Campina Grande - Departamento de Sistemas e Computação
Av. Aprígio Veloso, s/n, Bodocongó 58.429-900 - Campina Grande, PB - Brasil
{*rostand, fubica*}*@lsd.ufcg.edu.br*

[2]Universidade Federal da Paraíba - Departamento de Informática
Laboratório de Aplicações de Vídeo Digital 58.050-900, João Pessoa, PB
{*rostand, dienert, guido, denio*}*@lavid.ufpb.br*

Abstract. *The Just in Time Clouds (JiT Clouds) approach considers the provision of cloud computing using outsourced computing resources. Depending on their characteristics and location in the spectrum of scale, outsourced resources can provide different levels of quality of service, scalability and elasticity. The level of quality of service offered by a JiT Cloud is entirely dependent on the level of quality of service supported by the resources used, which is strongly related to the pattern of granularity, volatility and dispersion that they exhibit. In this paper, we study the use of computing resources for the composition of JiT Clouds through the use of specific mechanisms for its discovery, allocation and coordination, in the worst case scenario of high granularity, high volatility and high dispersion. Our simulation results show that, even in scenarios of high turnover of autonomous and geographically distributed nodes, it is possible to build JiT Clouds with the appropriate collective availability to achieve controlled levels of computational throughput.*

Resumo. *A abordagem Just in Time Clouds (JiT Clouds) considera o provimento de computação na nuvem usando recursos computacionais terceirizados. Dependendo de suas características e da sua localização no espectro de escala, os recursos terceirizados podem fornecer diferentes níveis de qualidade de serviço, elasticidade e escalabilidade. O nível de qualidade de serviço oferecido por uma JiT Cloud é inteiramente dependente do nível de qualidade de serviço suportado pelos recursos usados, o qual está fortemente relacionado ao padrão de granularidade, volatilidade e dispersão dos recursos. Neste trabalho, nós estudamos o uso de recursos para a composição de JiT Clouds através do uso de mecanismos específicos para a sua descoberta, alocação e coordenação, no cenário mais desafiador, caracterizado por alta granularidade, alta volatilidade e alta dispersão. Nossos resultados de simulação mostram que, mesmo em cenários de altíssima rotatividade de nós autônomos e distribuídos geograficamente, é possível construir JiT Clouds com a disponibilidade coletiva adequada para atingir níveis controlados de vazão computacional.*

1. Introdução

É cada vez maior a quantidade de dados que estão sendo gerados atualmente nas mais diversas áreas de conhecimento, por sensores, experimentos científicos e modelos de simulação, dentre outras fontes. Alguns conjuntos de dados são tão grandes que a única maneira viável para processá-los em um tempo razoável é quebrar o processamento em tarefas menores e executá-las em paralelo no maior número disponível de processadores.

Obviamente, paralelismo em larga escala só pode ser alcançado se houver unidades de processamento disponíveis e um nível relativamente elevado de independência entre as tarefas que compõem a aplicação paralela. Felizmente, muitas das cargas de trabalho das aplicações paralelas podem ser mapeadas em tarefas que podem ser processadas de forma completamente independente uma das outras, compondo uma classe de aplicações conhecida como "bag-of-tasks" (BoT) [Cirne et al. 2003]. O

fato de que as tarefas de uma aplicação BoT são totalmente independentes, não só faz o agendamento trivial, no tocante ao escalonamento e execução das tarefas, mas também faz com que a tolerância a falhas seja mais simples de tratar, já que um mecanismo de repetição pode ser usado para recuperar tarefas que eventualmente falhem durante a execução.

O paradigma da computação na nuvem permite o fornecimento de infraestrutura de Tecnologia da Informação sob a forma de um serviço que os clientes adquirem sob demanda e pagam apenas pela quantidade de serviços que realmente consomem. Muitas aplicações que processam grandes cargas de trabalho em paralelo poderiam potencialmente se beneficiar da elasticidade oferecida pelos provedores de computação na nuvem pois permite ao cliente aumentar ou diminuir a capacidade de sua infraestrutura de TI sem qualquer custo adicional. Essa característica faz com que o ônus dos custos e riscos associados ao planejamento da capacidade da infraestrutura de TI passem do cliente para o provedor do serviço. Entretanto, o estado-da-prática em provimento de computação na nuvem impõe um limite a essa elasticidade para que se possa garantir uma disponibilidade suficientemente elevada para os serviços e, ao mesmo tempo, manter os custos operacionais em um nível aceitável. Isso restringe o escopo das aplicações que poderiam se beneficiar do paradigma de computação em nuvem. Em particular, esse é o caso para muitas aplicações científicas do tipo BoT [Costa et al. 2011b].

Em trabalhos anteriores, os autores propuseram as *Just in Time Clouds* ou *JiT Clouds* [Costa et al. 2011a], uma abordagem alternativa para a construção de provedores de computação na nuvem baseada na utilização de recursos terceirizados, na qual os provedores de serviço apenas incorrem em custos de provisionamento quando os recursos terceirizados que eles usam para fornecer os seus serviços são demandados pelos seus clientes e apenas durante o período que eles são necessários, permitindo uma ampliação de algumas ordens de magnitude no limite que precisa ser imposto aos clientes. Dessa forma, as *JiT Clouds* se apresentam como uma infraestrutura adequada para a execução de aplicações BoT de larga escala.

As *JiT Clouds* podem ser montadas sobre recursos que estejam distribuídos por todo o espectro de recursos terceirizados de baixa escala[1]. Uma das missões do operador de uma *JiT Cloud*, o *JiT Provider*, é descobrir e explorar o potencial dos recursos tercerizados disponíveis alinhando-os com as necessidades das aplicações dos clientes. Dependendo de suas características, os recursos terceirizados podem fornecer diferentes níveis de qualidade de serviço, elasticidade e escalabilidade. O nível de qualidade de serviço oferecido por uma *JiT Cloud* é inteiramente dependente do nível de qualidade de serviço suportado pelos recursos usados para montá-la, o qual está relacionado ao padrão de granularidade, volatilidade e dispersão dos recursos.

Por *granularidade*, entende-se o nível de capacidade computacional presente em cada recurso tercerizado. Nesta classificação, servidores de alta capacidade e *clusters*, representam *recursos terceirizados de baixa granularidade* (*coarse-grained*), que são mais densos e poderosos, enquanto computadores pessoais representam *recursos terceirizados de alta granularidade* (*fine-grained*), mais leves e de menor capacidade.

A *volatidade*, por sua vez, representa o nível de disponibilidade e confiabilidade que o recurso tercerizado oferece quando alocado para uma determinada tarefa. Dedicação exclusiva, mecanismos de contingenciamento e tolerância a falhas caracterizam os *recursos terceirizados de baixa volatilidade*, enquanto que o uso oportunista e a falta de garantias de funcionamento são algumas das principais características dos *recursos terceirizados de alta volatilidade*.

A última propriedade considerada, a *dispersão*, está relacionada com o nível de distribuição dos recursos terceirizados. Os recursos concentrados em centros de dados representam *recursos terceirizados de baixa dispersão* enquanto que recursos individuais, distribuídos geograficamente, são *recursos terceirizados de alta dispersão*.

[1]Nesta categoria situam-se todos os detentores de recursos ociosos que não possuem, sozinhos, capacidade excedente suficiente para uma atuação solo como provedores públicos de computação na nuvem, como fez a *Amazon Bookstore*, por exemplo.

Quando os recursos estão concentrados em centros de dados e sua capacidade está localizada mais próxima do topo da magnitude que limita a baixa escala de recursos tercerizados, ou seja, são recursos computacionais semelhantes aos usados pelos provedores de tradicionais de computação na nuvem, os níveis de serviço oferecidos são consistentes com os praticados no mercado. Dessa forma, *JiT Clouds* baseadas em recursos de baixa granularidade, baixa volatilidade e baixa dispersão podem ser usadas para hospedar aplicações tipicamente suportadas por computação na nuvem.

No outro extremo do espectro da escala, quando os recursos terceirizados são de grão pequeno e distribuídos, eles precisam ser agrupados e coordenados pelo *JiT Provider* para a sua exploração. Estes recursos de alta granularidade, alta volatilidade e alta dispersão podem ser **convencionais**, representados por equipamentos padrão de processamento, e **não convencionais**, como receptores de TV Digital, por exemplo. Boa parte desses dispositivos não convencionais, notadamente os mais recentes, são equipados com processadores poderosos e quantidade razoável de memória, permitindo-lhes a execução de aplicações [Costa et al. 2012]. No entanto, como estes dispositivos são tipicamente recursos não dedicados e voláteis, uma *JiT Cloud* baseada neles é menos confiável do que aquela que é construído sobre recursos privados e dedicados. No entanto, há evidências suficientes de que existem clientes dispostos a utilizar serviços com qualidade de serviço do tipo *best-effort*: por um lado, a mera existência das *spot instances*[2] da AWS é uma boa indicação disso; por outro lado, a abundância de aplicações HTC[3] científicas e industriais adaptáveis para execução em ambientes de nuvem com qualidade de serviço equivalente ao proporcionado pelas *spot instances*, fornecem evidências adicionais de que um serviço altamente elástico e escalável de computação na nuvem é de muita utilidade.

Neste trabalho, nós analisamos o potencial de uso de recursos de alta granularidade, alta volatilidade e alta dispersão para a composição de *JiT Clouds* através do uso de mecanismos específicos para a sua descoberta, alocação e coordenação. Nossos resultados de simulação mostram que, mesmo em cenários de altíssima rotatividade de nós autônomos e distribuídos geograficamente, é possível construir *JiT Clouds* com a disponibilidade coletiva [Andrzejak et al. 2008] adequada para atingir níveis controlados de vazão computacional.

O restante do documento está organizado da seguinte forma. Na Seção 2, para permitir o entendimento do restante do trabalho, apresentamos a arquitetura geral do mecanismo considerado para a construção de *JiT Clouds* dinâmicas baseadas em recursos voláteis e distribuídos. Nessa seção nós descrevemos formalmente o modelo de operação do sistema, e apresentamos os desafios trazidos pelas características particulares das *JiT Clouds* estudadas neste artigo. A Seção 3 traz uma descrição de como foi realizada a nossa avaliação, incluindo uma discussão do modelo de simulação utilizado. Na Seção 4, analisamos os resultados obtidos nos nossos experimentos, enquanto que na Seção 5 discutimos alguns trabalhos relacionados. Finalmente, a Seção 6 traz as nossas considerações finais.

2. *JiT Clouds* Baseadas em Dispositivos de Alta Granularidade, Alta Volatilidade e Alta Dispersão

A fim de construir *JiT Clouds* dinâmicas e de alta vazão baseadas em recursos terceirizados dispersos, de pequena capacidade e não dedicados é necessário fornecer uma maneira de acessar, individualmente, uma grande quantidade de processadores, enviar programas e, possivelmente, dados, para todos e, remotamente, desencadear a execução do código transmitido. Em seguida, é necessário reunir os resultados produzidos, e, finalmente, liberar os recursos alocados de forma que outras aplicações possam usá-los.

A ideia de alocar uma enorme quantidade de recursos através da abstração de um *JiT Cloud*,

[2]Nessa modalidade, o usuário faz uma oferta para instâncias do Amazon EC2 disponíveis e as executa sempre que sua oferta exceder o preço *spot* atual, o qual varia em tempo real com base na demanda e cujo aumento pode ocasionar a retomada da instância a qualquer tempo.

[3]Em uma classificação bastante ampla, a computação paralela é normalmente dividida em Computação de Alto Desempenho (HPC, do inglês *High Performance Computing*) e Computação de Alta Vazão (HTC, do inglês *High Throughput Computing*) [Litzkow et al. 1988].

habilitá-los para o processamento distribuído de aplicações paralelas (centenas de milhares de computadores conectados via Internet, por exemplo) e fazê-lo a um custo muito menor do que alternativas tradicionais, apesar de atrativa, representa um desafio não trivial. A questão principal é: **onde** encontrar milhões de processadores terceirizados disponíveis e **como** configurá-los em conformidade e instantaneamente para o uso em *JiT Clouds* dinâmicas voltadas para os requisitos de alta vazão de aplicações BoT? Além disso, como executar esta tarefa com um atraso mínimo?

Neste sentido, uma categoria singular destes dispositivos tercerizados desperta um interesse especial para este trabalho: aqueles que podem ser organizados em uma rede de *broadcast*[4]. Uma rede de *broadcast* possui o potencial de permitir a comunicação simultânea com todos os dispositivos conectados, os quais podem ser coordenados para realizar uma determinada ação. Nesta abordagem, programas transmitidos através do canal de *broadcast* podem ser carregados e executados concomitantemente por todos os recursos computacionais conectados à rede de *broadcast* em um dado momento. Este mecanismo torna possível construir, de uma forma realmente rápida[5] e controlada, *JiT Clouds* distribuídas e dinâmicas.

2.1. *Infraestrutura Computacional Distribuída Sob Demanda*

Usando o conceito de redes de *broadcast*, os autores propuseram em trabalhos anteriores [Costa et al. 2009] uma nova arquitetura para construir *Distributed Computing Infrastructures* (DCIs) dinâmicas baseadas em recursos computacionais de alta granularidade, alta volatilidade e alta dispersão que é, ao mesmo tempo flexível e altamente escalável, sendo aplicável para a execução eficiente de aplicações BoT de larga escala e curta duração. Com esta abordagem, um cliente poderá alocar, sob demanda, um conjunto com um grande número de unidades de processamento, chamada de instância DCI, que executará sua aplicação BoT de forma tão eficiente quanto possível. Após completar a execução, o cliente liberará a instância DCI que foi criada. Por causa desta singularidade, a arquitetura é chamada de *Infraestrutura Computacional Distribuída Sob Demanda* (ou OddCI, do inglês *On-Demand Distributed Computing Infrastructure*).

A arquitetura OddCI é formada por um *Provider*, um *Backend*, uma ou mais redes de *broadcast*, cada uma contendo um canal de *broadcast* e um *Controller*, e *Processing Node Agents* (PNA). Estes últimos são programas a serem enviados e executados em cada um dos recursos computacionais acessíveis pelo *Controller* através da sua rede de *broadcast* correspondente. Além disso, é assumido que os recursos computacionais também possuem um canal bidirecional, chamado de *canal direto*, o qual os conecta tanto com o *Backend* quanto com o seu respectivo *Controller* (1).

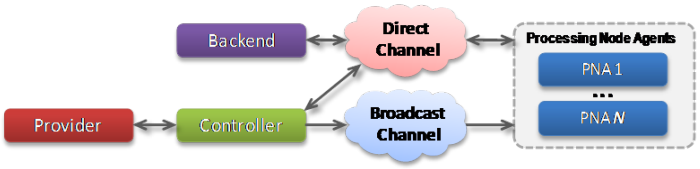


Figura 1. Visão Geral da Arquitetura OddCI

O *Provider* (provedor) é responsável por criar, gerenciar e destruir as instâncias OddCI de acordo com as solicitações dos clientes e também pela autenticação do cliente e pela verificação das

[4]O termo *broadcasting* está, originalmente, relacionado a transmissões de rádio ou televisão e significa a distribuição, de forma simultânea e através de um meio físico específico e unidirecional (o canal de *broadcast*), de sinais de áudio e/ou vídeo contendo programação para uma determinada audiência. Considerando o mesmo princípio de transmissão de um-para-muitos, será usado o termo **rede de *broadcast*** para representar uma rede composta por um transmissor digital de dados, um **canal de *broadcast***, um conjunto de equipamentos receptores com capacidade de processamento de aplicações paralelas e possibilidade de acesso a um canal de interação *full-duplex*, comumente uma conexão com a Internet.

[5]Na verdade, o quão rápido o *software* será carregado dependerá do tamanho dos dados a serem transmitidos e da velocidade do canal de *broadcast*.

suas credenciais para usar os recursos que estão sendo requisitados. O *Controller* (controlador) é encarregado de configurar a infraestrutura, conforme instruído pelo *Provider*, e formatar e enviar, via canal de *broadcast*, mensagens de controle e imagens (executáveis) para os PNAs, necessárias para construir e manter as instâncias OddCI. A responsabilidade do *Backend* (retaguarda) é o gerenciamento das atividades específicas de cada aplicação sendo executada: distribuição de tarefas, provisionamento de dados de entrada e recepção e pós-processamento dos resultados gerados pela aplicação paralela. Cabe aos *Processing Node Agents - PNA* (agentes processadores) o gerenciamento da execução da aplicação do cliente no dispositivo computacional e o envio de sondas periódicas (*heartbeat messages*) para sinalizar o seu estado.

O *Direct Channel* (canal direto), por sua vez, é uma rede de comunicação bidirecional que permite a comunicação entre todos os componentes da arquitetura, tal como a Internet, enquanto que o *Broadcast Channel* (canal de *broadcast*) é um canal unidirecional para envio de dados do *Controller* para os dispositivos. Pode ser, por exemplo, um canal de TV Digital ou uma ERB de uma rede celular.

2.2. Modelo de Operação OddCI

Nesta subseção é feita uma descrição mais formal do modelo de operação de sistemas OddCI que foi utilizado na nossa avaliação.

Consideramos uma rede de *broadcast* que pode acessar um conjunto $\mathbb{D}$ de dispositivos. Seja $A(d,t)$ uma função boleana no tempo que indica se um dispositivo $d \in \mathbb{D}$ está ativo no momento t. O conjunto de dispositivos *ativos* no momento t, $\mathbb{D}^a(t)$, é dado por $\mathbb{D}^a(t) = \{ d \mid d \in \mathbb{D} \bigwedge A(d,t) = true\}$ e o conjunto de dispositivos *inativos* no momento t, $\mathbb{D}^i(t)$, é dado por $\mathbb{D}^i(t) = \mathbb{D} \backslash \mathbb{D}^a(t)$. É assumido que os dispositivos são voláteis, ou seja, os dispositivos podem alternar entre os estados *ativo* e *inativo* em qualquer momento e, portanto, um mesmo dispositivo $d \in \mathbb{D}^a(t')$ pode pertencer a $\mathbb{D}^i(t''), t' \neq t''$.

Seja o serviço demandado pelos clientes de um provedor de um sistema OddCI definido por uma sequência de tuplas r_1, r_2, ..., r_n com $r_j =< t_j, q_j, l_j >$, onde t_j é o momento no qual r_j é submetida, q_j é a quantidade desejada de dispositivos simultâneos que devem ser alocados e l_j é a duração do intervalo de tempo no qual os q_j recursos serão necessários ($t_j, q_j, l_j \in \mathbb{N}$). A instância OddCI $\mathbf{I}_j$, $1 \leq j \leq n$, representa o atendimento da requisição r_j pelo sistema.

Seja $L(d,t)$ a função boleana que indica se o dispositivo d está alocado a alguma instância no tempo t, o conjunto $\mathbb{D}^a(t)$ pode ser decomposto em $\mathbb{D}^a(t) = \mathbb{D}^l(t) \cup \mathbb{D}^d(t)$, onde $\mathbb{D}^l(t)$ é o subconjunto dos dispositivos ativos e alocados a instâncias no momento t ($\mathbb{D}^l(t) = \{d \mid d \in \mathbb{D}^a(t) \bigwedge L(d,t) = true\}$) e $\mathbb{D}^d(t)$ é o subconjunto dos dispositivos ativos que estão disponíveis no momento t ($\mathbb{D}^d = \mathbb{D}^a(t) \backslash \mathbb{D}^l(t)$).

Um controlador ao ser designado, através de uma estratégia de escalonamento, pelo provedor para o atendimento de uma demanda r_j, tentará fazer a alocação dos q_j dispositivos solicitados através do envio de mensagens de controle para a rede de *broadcast* que controla. Seja m uma mensagem de controle enviada através do canal unidirecional no momento t, então todos os dispositivos pertencentes a $\mathbb{D}^d(t + T(m))$ receberão e processarão m, onde $T(m)$ é a duração da transmissão da mensagem de controle m. $T(m)$ é uma função da taxa de transmissão e do retardo médio do canal unidirecional e do tamanho da mensagem m.

Seja $\mathbb{D}^r(m) \subseteq \mathbb{D}^d(t + T(m))$ o subconjunto dos dispositivos ativos *disponíveis* em $t+T(m)$ que responderem, através dos seus respectivos canais bidirecionais, à convocação do controlador feita pela mensagem m. O subconjunto $\mathbb{D}^v(m)$ com os primeiros q_j dispositivos de $\mathbb{D}^r(m)$ que atendam a um critério de elegibilidade definido pelo *Controller* em função dos requisitos do cliente serão alocados para a instância $\mathbf{I}_j$. Os demais dispositivos, $\mathbb{D}^r(m) \backslash \mathbb{D}^v(m)$, serão descartados[6].

Para lidar com a volatilidade do sistema, assumimos que o sistema de tarifação adotado pelo

[6]Por simplicidade foi assumido que todos os dispositivos na rede de *broadcast* simulada atendiam ao critério de elegibilidade desejado.

provedor para o uso de seus recursos é baseado na apuração de cada intervalo de tempo com duração σ, chamado *slot* de processamento, durante o qual um dispositivo permanece ativo e alocado a uma instância. Sempre que um dispositivo d é alocado para a instância $\mathbf{I}_j$ em um momento t, o *slot* de processamento $s_{j,d,t}$ é iniciado. O *slot* $s_{j,d,t}$ é dito completado se d permanece alocado para a instância $\mathbf{I}_j$ até o momento $t + \sigma$. Apenas *slots* completados são tarifados.

Seja $\mathbb{S}_j^i$ o conjunto de todos os *slots* iniciados na instância $\mathbf{I}_j$ e seja $O(j,d,t)$ uma função boleana que indica se o *slot* $s_{j,d,t}$ foi completado, então o conjunto de *slots* completados na instância $\mathbf{I}_j$ é dado por $\mathbb{S}_j^c = \left\{ s_{j,d,t} \mid s_{j,d,t} \in \mathbb{S}_j^i \bigwedge O(j,d,t) = true \right\}$. Uma instância $\mathbf{I}_j$ é completada quando um mínimo de $\left\lceil \frac{l_j}{\sigma} \right\rceil \times q_j$ *slots* de processamento *completados* é atingido, ou seja, $\left|\mathbb{S}_j^c\right| \geq \left\lceil \frac{l_j}{\sigma} \right\rceil \times q_j$. Caso $\mathbf{I}_j$ ainda não tenha sido completada quando o *slot* $s_{j,d,t}$ for completado, o dispositivo d será realocado à instância $\mathbf{I}_j$, iniciando o *slot* $s_{j,d,t+\sigma}$.

Seja $I(d,t)$ a função que indica a qual instância o dispositivo $d \in \mathbb{D}^a(t)$ está alocado com exclusividade no tempo t:

$$I(d,t) = \begin{cases} j, \text{ se } \textit{d} \text{ está alocado à instância } \textit{I}_\textit{j} \text{ no momento } \textit{t} \\ 0, \text{ se } \textit{d} \text{ não está alocado em nenhuma instância no momento } \textit{t} \end{cases}, d \in \mathbb{D}^a(t),$$

então o conjunto de dispositivos alocados à instância $\mathbf{I}_j$ no momento t, $\mathbb{D}_j^l(t)$, é dado por $\mathbb{D}_j^l(t) = \{d \mid d \in \mathbb{D}^a(t) \bigwedge I(d,t) = j\}$.

2.3. O Desafio da Alta Volatilidade

Como os dispositivos acessíveis pela rede de *broadcast* são voláteis, os dispositivos ativos alocados à instância $\mathbf{I}_j$ podem, eventualmente, se tornar inativos em qualquer momento e tais perdas de dispositivos precisam ser identificadas e repostas. Para lidar com a eventual perda de dispositivos ativos alocados para uma instância $\mathbf{I}_j$ e recompor o seu tamanho para o valor desejado, q_j, o controlador pode enviar novas mensagens de controle para alocar os dispositivos faltantes sempre que necessário. A frequência de envio de mensagens de controle para manter o tamanho solicitado de uma instância é definida pela estratégia de provisionamento adotada.

A reposição de dispositivos para a instância $\mathbf{I}_j$ no momento t através do envio de uma mensagem de controle m levará o tempo $T(m)$ para atingir os dispositivos ativos disponíveis no momento $t + T(m)$, $\mathbb{D}^d(t + T(m))$. Neste sentido, a estratégia de provisionamento adotada pelo controlador precisa considerar a reposição tanto dos dispositivos já perdidos por $\mathbf{I}_j$ no momento t, quanto dos que poderão ser perdidos adicionalmente até o momento $t + T(m)$.

Para evitar o uso ineficiente de recursos do controlador e da rede de *broadcast* causado pelo descarte de um número grande de dispositivos, a quantidade de dispositivos que responderem à mensagem de controle m, $|\mathbb{D}^r(m)|$, deve ser o mais próximo possível da quantidade de dispositivos que serão alocados a $\mathbf{I}_j$ em decorrência do envio de m, $|\mathbb{D}^v(m)|$. Para tal, é atribuído para cada mensagem de controle m enviada, um valor $P(m)$ que representa a probabilidade de cada dispositivo em $\mathbb{D}^d(t + T(m))$ responder ou não à mensagem m enviada no momento t. O cálculo de $P(m)$ leva em consideração a quantidade de dispositivos que se necessita e a quantidade total de dispositivos que estarão disponíveis: $P(m) = |\mathbb{D}^v(m)| / |\mathbb{D}^d(t + T(m))|$. Neste sentido, como o estado dos dispositivos da rede de *broadcast* pode mudar constantemente, é necessário dispor de algum mecanismo para fazer, em t, uma estimativa do número de dispositivos disponíveis em um momento futuro, $t + T(m)$.

Por outro lado, para minimizar a perda de dispositivos em $\mathbf{I}_j$, o controlador precisa adotar algum critério de elegibilidade para indicar, dentre os dispositivos existentes em $\mathbb{D}^d(t + T(m))$ que irão responder a m, aqueles dispositivos que possuam uma expectativa de maior permanência no estado ativo.

Do ponto de vista do cliente, a existência da volatilidade do sistema implica na necessidade de adequar o tamanho máximo das tarefas da sua aplicação como um divisor do tamanho do *slot* de processamento adotado pelo provedor, ou seja, deve ser possível a conclusão total ou parcial (via

checkpoints) de uma ou mais tarefas durante a duração de um *slot* de processamento.

3. Descrição dos Experimentos

O objetivo principal da nossa avaliação, baseada em simulação, foi investigar o potencial de uso de recursos tercerizados em *JiT Clouds* no cenário mais desafiador, caracterizado por alta granularidade, alta volatilidade e alta dispersão através do uso de mecanismos específicos para a sua descoberta, alocação e coordenação.

No contexto considerado, os recursos alocados para a construção de *JiT Clouds* possuem um grau extremo de distribuição e granularidade, sendo necessário acessar cada dispositivo diretamente e escalonar individualmente cada *slot* de processamento. Além disso, os dispositivos são eminentemente voláteis e alocar um determinado conjunto de recursos para processar tarefas em paralelo implica que, ao longo do tempo, o conjunto sofrerá redução no seu tamanho. Assim, é necessário reparar a perda esperada de nós através de alguma estratégia de antecipação ou recuperação, que chamamos de algoritmos compensatórios.

A utilização de métodos de predição para suportar mecanismos que assegurem a disponibilidade coletiva (*collective availability* [Andrzejak et al. 2008]) de uma coleção volátil de recursos foi estudada por *Andrzejak et al.* O estudo mostra que através de previsão, métodos acurados podem ser usados para garantir que um subconjunto ω qualquer de nós em um conjunto volátil Ω esteja disponível durante um período ΔT ao custo de uma sobrecarga de controle razoável. As técnicas de predição avaliadas foram bem sucedidas também para avaliar o custo de disponibilidade do subconjunto em termos de nível de redundância necessário e da sobrecarga de migração entre recursos não dedicados e dedicados para compensar a perda de nós.

A taxa de sucesso (*success rate*) obtida quando se tenta manter um subconjunto ω de dispositivos disponíveis em um dado período é dependente do tempo médio de disponibilidade dos dispositivos do conjunto volátil (*historical turnover rate*) e do tamanho de ω, mas pode ser equilibrada através de um nível adequado de redundância R através da alocação de $|\omega| + R$ recursos computacionais. Os resultados de *Andrzejak et al.* indicam que a solução mais prática para controle da disponibilidade coletiva é uma combinação de uma abordagem de previsão simplificada com o ranqueamento dos dispositivos de acordo com sua disponibilidade histórica. Com base nisso, uma sequência de *bits* 0 ou 1 pode representar a disponibilidade histórica de cada dispositivo em instantes de tempo específicos e um modelo de predição processa as sequências de *bits* dos dispositivos gerando um *ranking* de regularidade que pode permitir ou não a sua escolha para o atendimento de requisitos de disponibilidade específicos. Em nossa abordagem, uma variação escalável desse método é obtida através do cálculo e manutenção das informações históricas e do ranqueamento de disponibilidade corrente nos próprios dispositivos (PNA) e enviadas ao *Controller* em cada *heartbeat message* transmitida. Um alvo de ranqueamento é informado pelo *Controller* juntamente com os outros requisitos para o dispositivo nas mensagens de controle enviadas e, considerando o histórico de disponibilidade do dispositivo, o PNA decide se irá juntar-se ou não à instância. Com o uso de ranqueamento, o critério de elegibilidade do PNA primeiro verifica o *ranking* do dispositivo e depois aplica o fator probabilístico indicado em $P(m)$, o qual deve ter sido calculado considerando uma estimativa da quantidade de dispositivos disponíveis que atendem ao alvo de ranqueamento desejado. Eventualmente, o controlador pode precisar diminuir o *ranking*-alvo para ajustá-lo à condição atual de ranqueamento dos dispositivos disponíveis e conseguir obter a quantidade necessária de dispositivos para repor o tamanho da instância.

Para analisar como a volatilidade e a contenção de recursos presentes na rede de *broadcast* podem afetar a disponibilidade coletiva necessária, foram considerados dois cenários de uso:

- *Atendendo a Aplicações Sensíveis ao Tempo*: No primeiro cenário, chamado *Vazão Mínima*, o controlador tenta garantir que a duração esperada para a instância $\mathbf{I}_j$ seja observada, ou seja, que os $\left\lceil \frac{l_j}{\sigma} \right\rceil \times\ q_j$ *slots* solicitados sejam completados no tempo l_j. Em tal contexto, o número de *slots* completados na instância $\mathbf{I}_j$ precisa permanecer em uma média maior ou igual a q_j durante todo o ciclo de vida de $\mathbf{I}_j$. Para lidar com a eventual perda de dispositivos e

mesmo assim garantir uma vazão mínima q_j, o controlador deve aplicar um determinado nível de redundância sobre o tamanho mínimo desejado para a instância. Para isso, são enviadas, proativamente, mensagens de controle para regenerar o tamanho da instância para um valor alvo $q_j + X$, onde X é a quantidade adicional necessária para compensar as eventuais perdas de dispositivos que ocorrerão até o envio do próximo comando de regeneração. Baseado na última consolidação de *heartbeat messages*, o controlador calcula X, o momento t para envio de cada mensagem de controle m para a instância $\mathbf{I}_j$ e também $|\mathbb{D}^d\left(t + T\left(m\right)\right)|$ em função da taxa histórica de perda de dispositivos observada na rede de *broadcast* em um dado período de referência, cujo momento inicial padrão é o momento de submissão da demanda r_j, ou seja, t_j. O valor $P\left(m\right)$ é definido pelo controlador para cada mensagem de controle m considerando q_j, X, $|\mathbb{D}_j^l(t)|$ e $|\mathbb{D}^d\left(t + T\left(m\right)\right)|$. Neste cenário, é aceitável que o tamanho solicitado para a instância (q_j) seja excedido para compensar regimes de maior volatilidade.

- *Lidando com Capacidade Limitada no Backend*: No segundo cenário, chamado *Paralelismo Máximo*, o controlador tenta cumprir, tanto quanto possível, o limite do tamanho q_j solicitado para a instância sem excedê-lo. Assim, o número de dispositivos alocados para a instância $\mathbf{I}_j$, tende a permanecer em uma quantidade sempre igual ou menor do que q_j durante todo o seu ciclo de vida para respeitar a condição de que o *Backend* do cliente só consegue tratar, no máximo, q_j dispositivos simultaneamente. Sempre que a perda de dispositivos causada pela volatilidade da rede de *broadcast* atingir um determinado limite Y, ou seja, $\left|\mathbb{D}_j^l\left(t\right)\right| \leq q_j - Y$, serão enviadas, reativamente, mensagens de controle para regenerar o tamanho da instância para o valor alvo q_j. O valor adequado de Y, que representa o tempo de reação para regeneração da instância, é definido pelo controlador a partir do tempo $T\left(m\right)$ necessário para transmissão da mensagem de controle m, bem como em função da taxa histórica de perda de dispositivos observada na rede de *broadcast*. O valor $P\left(m\right)$ é definido pelo controlador para cada mensagem de controle m considerando q_j, Y, $|\mathbb{D}_j^l(t)|$ e $|\mathbb{D}^d\left(t + T\left(m\right)\right)|$. Neste cenário, é aceitável que a duração solicitada (l_j) não seja cumprida em regimes de maior volatilidade.

3.1. Implementação do Modelo de Simulação

O simulador usado nos experimentos foi baseado no ambiente OMNeT++ [Varga and Hornig 2008]. O OMNeT++ é composto por uma biblioteca de componentes e de um *framework* de simulação modular e flexível, que pode ser estendido usando a linguagem C++ para a lógica dos componentes, enquanto que a linguagem *NEtwork Description* (NED) é usada para descrição da topologia da rede, portas de comunicação, canais, conexões, dentre outros parâmetros. Para essa avaliação, algumas extensões nos componentes originais foram realizadas. Em particular, foram acrescentados os aspectos de transmissão em *broadcast* e o comportamento dos componentes da arquitetura, de acordo com o modelo de operação descrito na Seção 2.2 e os mecanismos descritos na Seção 3[7].

Parte da configuração do simulador foi baseada em uma etapa anterior da pesquisa na qual foram obtidas medições de campo em um *testbed* real: um protótipo de sistema OddCI para redes de TV Digital [Costa et al. 2012], cujos resultados permitiram confirmar o comportamento linear na transmissão de mensagens de controle por radiodifusão, a adequação dos recursos de comunicação direta dos receptores para troca de tarefas/resultados e o potencial de processamento de STBs de baixo custo (*low-end*).

O comportamento estocástico do sistema OddCI simulado foi modelado usando algumas variáveis independentes (aleatórias). A população total de dispositivos computacionais (ou nós) potencialmente acessíveis através da rede de *broadcast*, denominada $\mathbb{D}$, é determinada, *a priori*, como um parâmetro de simulação. Entretanto, a quantidade de nós ativos (i.e, que podem ser efetivamente atingidos por uma mensagem de controle) no início da simulação é modelada como uma variável aleatória com distribuição uniforme: $|\mathbb{D}^a(0)| = U(\mu, |\mathbb{D}|)$, onde μ é o número mínimo de dispositivos

[7]O modelo completo do simulador usado neste trabalho pode ser encontrado no sítio http://www.lsd.ufcg.edu.br/∼rostand/JiTDC_OddCISim.zip.

acessíveis através da rede de *broadcast*. Uma vez que o número inicial de dispositivos ativos $|\mathbb{D}^a(0)|$ é determinado no início da simulação, os dispositivos ativos iniciais são selecionados entre a população de dispositivos, $\mathbb{D}$, com igual probabilidade. Sempre que um nó individual é selecionado para ser ativado, ele permanece ativo por um tempo de sessão τ_{ON} e então é desativado por um período de espera (*standby*) τ_{OFF}. Dessa forma, os dispositivos ativos em um determinado momento na rede de *broadcast* configuram um processo estocástico que depende das seguintes variáveis: tamanho da população $|\mathbb{D}|$, o número inicial de dispositivos ativos, $|\mathbb{D}^a(0)|$, o tempo em sessão, τ_{ON}, e o tempo em *standby*, τ_{OFF}. Foi assumido um mesmo *ranking* de disponibilidade para os dispositivos em $\mathbb{D}$.

A volatilidade ($\mathcal{V}$) inserida no sistema simulado foi normalizada, através das probabilidades utilizadas em τ_{ON} e τ_{OFF} (que foram modeladas como variáveis aleatórias com distribuição Bernoulli), de forma a obter uma variação percentual controlada da quantidade de dispositivos que alternam entre o estado ativo e inativo na rede de *broadcast* dentro de cada período de tempo de tamanho σ, o intervalo de referência considerado, mas mantendo o total de ativos em qualquer tempo próximo da disponibilidade inicial configurada. Em resumo, o parâmetro $\mathcal{V}$ regula o percentual de dispositivos ativos ganhos e perdidos em um dado intervalo de tempo de tamanho σ, o mesmo adotado como duração de um *slot* de processamento. Uma vantagem desta associação da volatilidade à duração do *slot* de processamento é tornar os resultados obtidos em uma dada configuração aplicáveis em outros cenários de tarifação e granularidade de tarefas.

Para analisar o comportamento do sistema sob alta volatilidade em regimes de contenção de recursos, a carga de trabalho utilizada teve como objetivo estressar dois gargalos potenciais: a disponibilidade de dispositivos para atendimento da demanda e a concorrência pelo uso do canal de transmissão em *broadcast*. Para tal, foi fixado um pico de demanda ($\mathcal{P}$), representando o máximo da soma de dispositivos alocados para instâncias em um dado momento de um período de observação. A partir de $\mathcal{P}$, as cargas de trabalho de cada experimento foram construídas de forma relativa usando dois parâmetros do simulador: quantidade de instâncias simultâneas ($\mathcal{S}$) e a duração das instâncias em *slots* ($\mathcal{D}$). Assim, o *workload* de cada experimento é baseado na sua configuração e formado por $\mathcal{S}$ instâncias simultâneas iguais, todas iniciando no mesmo momento ($t_j = 0$), solicitando a mesma quantidade de dispositivos ($q_j = \frac{\mathcal{P}}{\mathcal{S}}$) pelo mesmo intervalo de tempo ($l_j = \mathcal{D} \times \sigma$). O tamanho de $\mathbb{D}$ é regulado pela aplicação de um *fator de contenção*, ζ, sobre $\mathcal{P}$: $|\mathbb{D}| = \zeta \times \mathcal{P}$.

3.2. Parâmetros do Sistema

Para atribuição dos parâmetros do sistema foram usadas duas estratégias: projeto de experimento (DoE, do inglês *Design of Experiment*) e varredura de parâmetros. Inicialmente, os parâmetros foram tratados em cada cenário considerado através de um DoE do tipo 2^k fatorial [Jain 1991].

Os fatores considerados no DoE foram: *Volatilidade* ($\mathcal{V}$), *Tamanho da População* ($|\mathbb{D}|$), *Tamanho da Imagem* ($\mathcal{T}$), *Instâncias Simultâneas* ($\mathcal{S}$) e *Duração da Instância* ($\mathcal{D}$).

Para o tamanho da imagem da aplicação, o qual está associado ao tempo de uso do canal de transmissão em *broadcast* para envio de cada mensagem de controle, foram considerados dois valores diferentes: *pequeno* (representativo do tamanho de módulos clientes de aplicações como o SETI@home [Anderson et al. 2002] e *grande* (representando "*workers*" de implementações padrão de *desktop grids* como o OurGrid [Cirne et al. 2006]). As imagens do tipo *pequeno* têm 512 Kbytes de tamanho e correspondem ao nível **baixo** do fator, enquanto que as imagens do tipo *grande* possuem tamanho de 5 Mbytes e representam o nível **alto** do fator. Os níveis atribuídos para os demais fatores em cada DoE estão apresentados nas Tabelas 1 e 2.

A variável de resposta considerada para o cenário do *Vazão Mínima* foi o *coeficiente médio de vazão* ($\overline{\Phi}$) das instâncias, o qual representa a relação entre a quantidade média de *slots* completados por ciclo e a quantidade necessária para que a duração esperada para a instância seja cumprida. Essa métrica é dada por $\overline{\Phi} = (\sum_{j=1}^{\mathcal{S}} (|\mathbb{S}_j^c|/\mathcal{D}/q_j))/\mathcal{S}$ e seu valor de referência é 1.

Para o cenário do *Paralelismo Máximo* foi escolhida a variável de resposta *coeficiente médio*

Tabela 1. DoE 2^k: Fatores, níveis e efeitos para o cenário *Vazão Mínima*

Fator	Baixo	Alto	Efeito Estimado	Soma dos Quadrados	Contribuição
A: Volatilidade ($\mathcal{V}$)	5%	75%	$-0,3335$	$0,8896$	$28,41\%$
B: População ($\|\mathbb{D}\|$)	$(1+\mathcal{V})\times\mathcal{P}$	$10\times\mathcal{P}$	$0,2672$	$0,5710$	$18,24\%$
C: Tamanho da Imagem ($\mathcal{T}$)	$512Kb$	$5Mb$	$-0,1667$	$0,2223$	$7,10\%$
D: Instâncias Simultâneas ($\mathcal{S}$)	10	100	$-0,1729$	$0,2392$	$7,64\%$
E: Duração da Instância ($\mathcal{D}$)	$10\ horas$	$100\ horas$	$-0,0184$	$0,0027$	$0,09\%$

de paralelismo ($\overline{\Pi}$) das instâncias, o qual representa a relação entre a quantidade efetiva de dispositivos fornecida e a quantidade de dispositivos solicitada. Esta métrica é dada por $\overline{\Pi} = (\sum_{j=1}^{\mathcal{S}}(|\overline{\mathbb{D}_j^l}|/q_j))/\mathcal{S}$ e seu valor de referência também é 1.

Tabela 2. DoE 2^k: Fatores, níveis e efeitos para o cenário *Paralelismo Máximo*

Fator	Baixo	Alto	Efeito Estimado	Soma dos Quadrados	Contribuição
A: Volatilidade ($\mathcal{V}$)	5%	75%	$-0,2216$	$0,3927$	$16,17\%$
B: População ($\|\mathbb{D}\|$)	$(1+\mathcal{V})\times\mathcal{P}$	$10\times\mathcal{P}$	$0,0449$	$0,0161$	$0,66\%$
C: Tamanho da Imagem ($\mathcal{T}$)	$512Kb$	$5Mb$	$-0,2327$	$0,4331$	$17,83\%$
D: Instâncias Simultâneas ($\mathcal{S}$)	10	100	$-0,2412$	$0,4653$	$19,16\%$
E: Duração da Instância ($\mathcal{D}$)	$10\ horas$	$100\ horas$	$0,0080$	$0,0005$	$0,02\%$

Foram conduzidas várias repetições dos 32 experimentos previstos no DoE realizado para cada um dos cenários considerados para obter médias com 95% de confiança com erro máximo de 5% para mais ou para menos. A contribuição de cada fator em cada cenário é mostrada nas Tabelas 1 (*Vazão Mínima*) e 2 (*Paralelismo Máximo*).

No cenário de *Vazão Mínima*, os fatores da *Volatilidade* e do *Tamanho da População* foram preponderantes com participação de $28,41\%$ e $18,24\%$, respectivamente (Tabela 1). Enquanto que no cenário de *Paralelismo Máximo*, além da *Volatilidade*, que responde por $16,17\%$, os fatores *Tamanho da Imagem* com $17,83\%$ e *Instâncias Simultâneas* com $19,16\%$ foram determinantes na variação da métrica observada (Tabela 2).

Como resultado da análise dos efeitos através de ANOVA [Jain 1991], o F-Value de $164,4793$ (*Vazão Mínima*) e $252,9781$ (*Paralelismo Máximo*) implicam que os modelos são significativos. O R^2 ajustado indica que os modelos explicam $98,75\%$ e $98,27\%$ da variação observada e o R^2 de predição está dentro de $0,20$ do R^2 ajustado, representando uma boa capacidade de predição dos modelos. Maiores detalhes sobre este estudo, incluindo os gráficos de diagnóstico, cubo e interação, podem ser encontrados *online* no mesmo endereço citado na Seção 3.1.

Para a realização das simulações, os valores dos parâmetros que não afetaram o comportamento da variável de resposta foram ajustados para os valores médios entre os níveis "Alto" e "Baixo" usados em cada DoE[8]. Para os fatores mais relevantes: *Volatilidade* e *Tamanho da População* (*Vazão Máxima*) e *Volatilidade*, *Tamanho da Imagem* e *Instâncias Simultâneas* (*Paralelismo Máximo*), foi aplicada uma varredura de parâmetros. Para a varredura não foi necessário ampliar os níveis usados no DoE, posto que já ocorreram restrições relevantes nos respectivos intervalos.

A Tabela 3 mostra como o sistema foi configurado para os experimentos dos dois cenários usando o resultado do DoE, os valores obtidos no *testbed* real e alguns padrões de mercado, como no caso da duração do *slot* de processamento baseada na forma de tarifação usada nas *spot instances*.

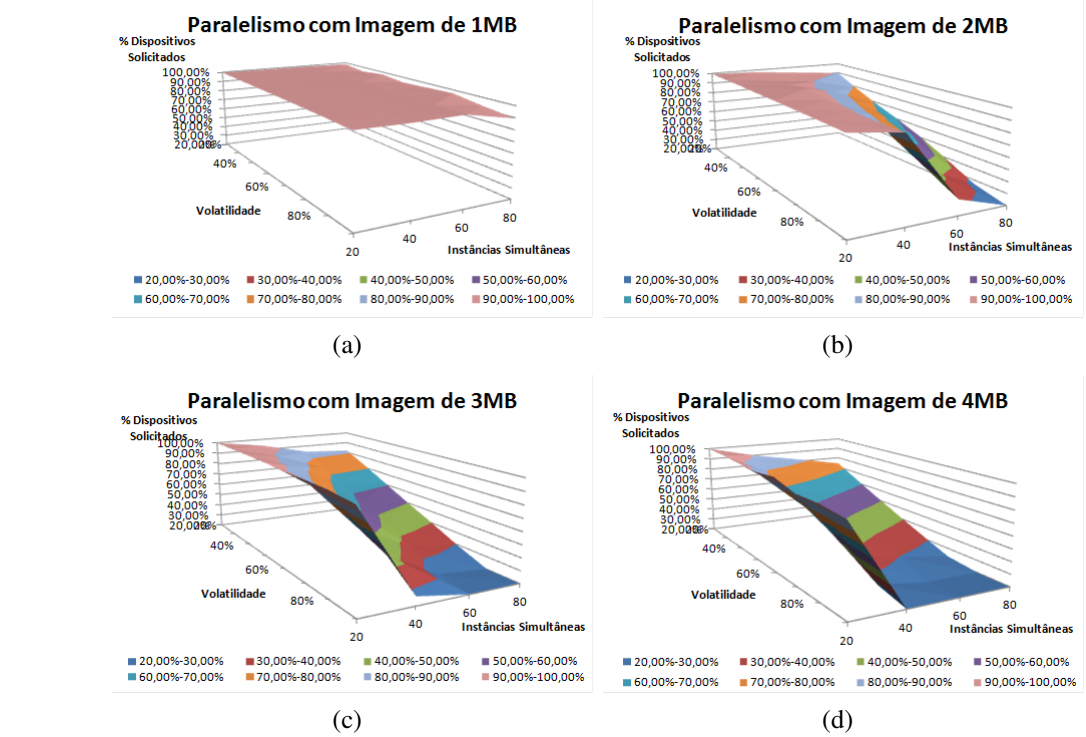
[8]Exceto no caso da *Duração da Instância*, com contribuição irrelevante, onde foi usado o nível "Baixo" com o objetivo de diminuir o tempo de execução de cada experimento.

Tabela 3. Parâmetros Usados nas Simulações

Parâmetro	**Cenário *Vazão Mínima***	**Cenário *Paralelismo Máximo***
Pico de Demanda ($\mathcal{P}$)	10.000 dispositivos	10.000 dispositivos
Taxa Canal Direto	1 Mbps	1 Mbps
Taxa Canal de *Broadcast*	1 Mbps	1 Mbps
Duração *slot* de processamento (σ)	1 hora	1 hora
Retardo Máximo	5 segundos	5 segundos
Disponibilidade Inicial ($\lvert\mathbb{D}^a(0)\rvert$)	100% da população	100% da população
Duração da Instância ($\mathcal{D}$)	10 *slots*	10 *slots*
Instâncias Simultâneas ($\mathcal{S}$)	50 instâncias	{20,40,60,80} instâncias
Tamanho da Imagem ($\mathcal{T}$)	{ 2, 5MB	{1MB,2MB,3MB,4MB}
População ($\lvert\mathbb{D}\rvert$)	{2.$\mathcal{P}$,3.$\mathcal{P}$,4.$\mathcal{P}$,5.$\mathcal{P}$, 6.$\mathcal{P}$,7.$\mathcal{P}$, 8.$\mathcal{P}$,9.$\mathcal{P}$}	10.$\mathcal{P}$
Volatilidade ($\mathcal{V}$)	{20%,30%,40%,50%, 60%,70%,80%,90%}	{20%,30%,40%,50%, 60%,70%,80%,90%}

4. Resultados e Análise

No primeiro experimento, realizado para o cenário de *Paralelismo Máximo*, o objetivo foi observar como a variação da volatilidade ($\mathcal{V}$), da quantidade de instâncias simultâneas ($\mathcal{S}$) e do tamanho da imagem da aplicação ($\mathcal{T}$) impacta na manutenção da quantidade desejada de dispositivos ativos para cada instância. Para eliminar a variável de contenção de dispositivos, a população foi configurada para 10 vezes o total da demanda concomitante esperada ($\lvert\mathbb{D}\rvert = 10 \times \mathcal{P}$). Para cobrir todas as combinações dos parâmetros de entrada foram realizados 128 experimentos - repetidos até que as médias obtidas tivessem 95% de confiança e erro máximo de 5% para mais ou para menos. A métrica de interesse observada foi o coeficiente médio de paralelismo das instâncias, $\overline{\Pi}$. Os resultados obtidos estão exibidos graficamente na Fig. 2.



(a) (b)

(c) (d)

Figura 2. *Paralelismo Máximo*: Métrica $\overline{\Pi}$ para diferentes tamanhos de imagem ($\mathcal{T}$)

Como pode ser observado na Fig. 2(a), quando lida com imagens de aplicação pequenas, o

controlador consegue compensar a perda de dispositivos em praticamente todos os regimes de volatilidade simulados, mesmo quando coordenando muitas instâncias simultâneas. Entretanto, à medida que o tamanho da imagem aumenta, aumenta o tamanho da mensagem de controle correspondente e diminui a capacidade do controlador de restabelecer o nível de paralelismo máximo das instâncias devido ao aumento proporcional do tempo de transmissão de cada mensagem de controle (Fig. 2(b)). Isso fica ainda mais evidenciado com o incremento no número de instâncias simultâneas, o que implica, na prática, no enfileiramento de mensagens de controle para serem enviadas pelo transmissor de *broadcast*. Esse efeito, que pode ser visualizado também nas figuras 2(c) e 2(d), é ampliado pelas restrições ao paralelismo máximo impostas neste cenário de uso, que ao limitar o tamanho que pode ser praticado para cada instância, não permite uma compensação antecipada das perdas através de redundância, o que diminuiria a quantidade de mensagens de controle reparatórias a serem enviadas e, consequentemente, a concorrência das instâncias pelo canal de *broadcast*. Associadamente, a inclusão de mecanismos adequados no controle de admissão pode otimizar o uso dos recursos do sistema através de um melhor escalonamento das instâncias ao longo do tempo.

No segundo experimento, realizado para o cenário de *Vazão Mínima*, o objetivo foi observar como a variação da volatilidade ($\mathcal{V}$) e do tamanho da população de dispositivos ($|\mathbb{D}|$) impactam na manutenção da quantidade desejada de *slots* de processamento completados, ou vazão, obtida em cada instância. Para controlar o nível de contenção de recursos, o tamanho da população foi iniciada em um patamar operacional mínimo, correspondente ao pico da demanda esperada acrescido da volatilidade inserida ($|\mathbb{D}| = \mathcal{P} \times (1+V)$), e foi sendo aumentada pela aplicação de um fator de contenção (um fator 2 equivale a uma população com o dobro da quantidade operacional mínima, um fator 3, ao triplo, e assim por diante). Para cobrir todas as combinações dos parâmetros de entrada foram realizados 64 experimentos - repetidos até que as médias obtidas tivessem 95% de confiança e erro máximo de 5% para mais ou para menos. A métrica de interesse principal foi a mesma usada no DoE, o coeficiente médio de vazão das instâncias, $\overline{\Phi}$. Os resultados obtidos estão exibidos na Fig. 3.

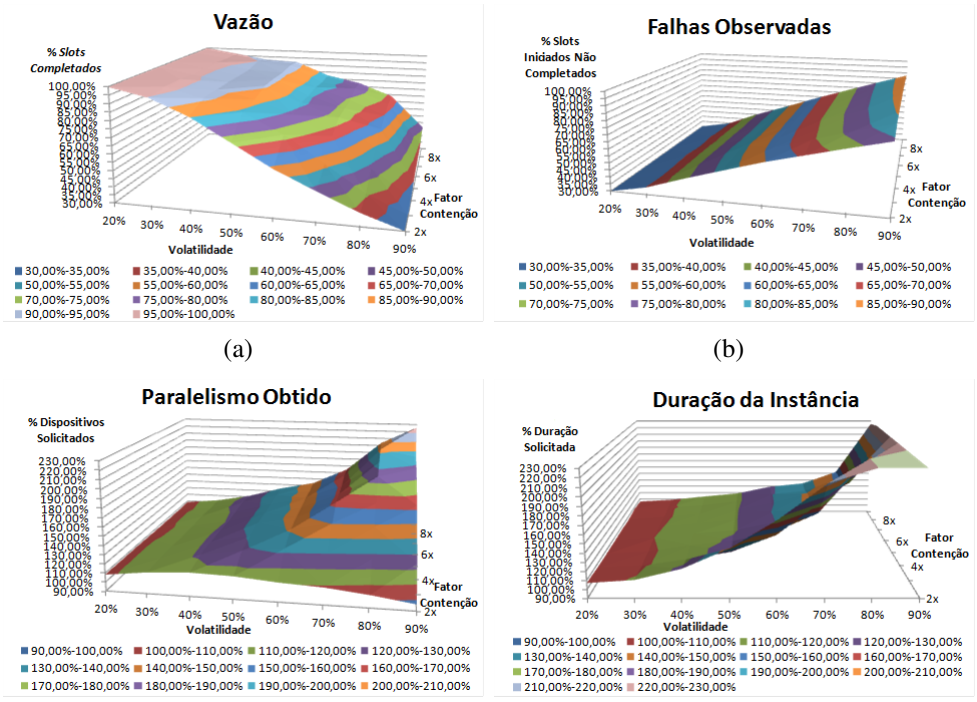


Figura 3. *Vazão Mínima*: Vazão, Duração, Paralelismo e Falhas observadas

Como ilustrado na Fig. 3(a), a quantidade média de *slots* de processamento completados por ciclo é fortemente afetada à medida que é inserida mais volatilidade no sistema. Nas configurações

com até 40% de volatilidade, ou seja, onde até 40% dos dispositivos alocados às instâncias falham em cada ciclo, foi possível manter níveis de vazão apenas 10% abaixo do solicitado, dependendo do fator de contenção do tamanho da população aplicado. Em tais níveis de volatilidade, o esforço de coordenação do provedor também é mantido controlado, como pode ser visto na Fig. 3(b), a qual traz o percentual de *slots* iniciados que não foram completados. Entretanto, à medida que a volatilidade é incrementada, a vazão entregue diminuiu consideravelmente apesar do aumento do custo operacional do provedor, com perdas de até 90% para a obtenção de vazão de apenas 30%. Cada *slot* não finalizado implica em custos operacionais, diretos e indiretos, para o provedor, principalmente no consumo de recursos de comunicação via canal de *broadcast* e canal direto dos dispositivos.

A métrica coeficiente médio de paralelismo das instâncias, $\overline{\Pi}$, também foi apurada para esse experimento. Pode ser visualizado na Fig. 3(c) que, por não haver restrição de tamanho para as instâncias, a quantidade de dispositivos ativos nas instâncias foi sendo aumentada à medida que a volatilidade percebida no sistema aumentava e ainda havia disponibilidade de recursos. O resultado do aumento do paralelismo repercute em uma atenuação dos efeitos da volatilidade sobre a vazão, como pode ser visualizado na Fig. 3(d), na qual a duração das instâncias torna a diminuir nos cenários com menor contenção de recursos mesmo em regimes de maior volatilidade. Obviamente, em contextos cuja disponibilidade de recursos não apresentem restrições ao nível de redundância praticados, como é o caso de redes de TV Digital com milhões de dispositivos, é possível aplicar níveis de paralelismo ainda maiores nas instâncias e ampliar a faixa de volatilidade onde alta vazão pode ser praticada. Entretanto, é necessário conciliar o nível de paralelismo com a capacidade do *Backend* e com o custo operacional do provedor.

5. Trabalhos Relacionados

Dentro do nosso conhecimento, nós somos o primeiro grupo a investigar o potencial do uso de redes de *broadcast* para a construção de infraestruturas computacionais distribuídas instantâneas e sob demanda [Batista et al. 2007, Costa et al. 2009]. Existem, entretanto, alguns outros trabalhos que apresentam convergência com a nossa pesquisa.

Fedak *at al.* [Fedak et al. 2010] construíram uma plataforma experimental para computação distribuída usando dispositivos de baixa capacidade conectados através de banda larga, chamada DSL-Lab, que oferece a possibilidade para pesquisadores realizarem experimentos em condições próximas àquelas que normalmente estão disponíveis com conexões domésticas com a Internet. Os resultados confirmam que é possível construir uma pilha completa de *software* em uma plataforma de design leve e de baixo custo sobre os dispositivos conectados em banda larga implementando gestão de recursos, eficiência energética, segurança e conectividade.

Neill *at al.* [Neill et al. 2011] investigam o uso de uma arquitetura de sistema heterogêneo que combina um *cluster* de computadores tradicionais, com um conjunto integrado de *set-top-boxes* para executar aplicações paralelas. Os resultados experimentais também confirmam que a rede de banda larga de processadores embarcados é uma nova e promissora plataforma para uma variedade de aplicações paralelas com uso intensivo de processamento e armazenamento (*computationally intensive and data-intensive grid applications*) e já é capaz de proporcionar ganhos significativos de desempenho para algumas classes de aplicações Open MPI.

6. Conclusão

Com o objetivo de investigar o uso de recursos terceirizados de alta granularidade, alta volatilidade e alta dispersão para a construção de *JiT Clouds* de alta vazão e usando uma nova arquitetura, chamada de *On-demand Distributed Computing Infrastructure* (OddCI), nós estudamos o comportamento do sistema e o impacto que os seus parâmetros têm sobre a sua eficiência através de simulação.

Nossos resultados mostram que, mesmo em cenários de altíssima rotatividade de nós autônomos e distribuídos geograficamente, é possível construir *JiT Clouds* com a disponibilidade coletiva adequada para atingir níveis controlados de vazão computacional usando os mecanismos de coordenação adequados.

No caso particular da aplicabilidade de sistemas OddCI para a descoberta, alocação e operação de *JiT DCs* dinâmicos, ficou evidenciado que a concorrência pelo uso do canal de *broadcast*, notadamente em contextos que envolvam a coordenação de muitas DCIs simultaneamente, requer a inclusão de mecanismos específicos em nível de controle de admissão e também na otimização da utilização dos recursos de comunicação de forma a permitir conciliar a qualidade do serviço prestado pelo provedor com os custos operacionais envolvidos. A investigação de mecanismos aplicáveis em tais aspectos será tratada em etapas futuras da nossa pesquisa, bem como a avaliação de aspectos de QoS e de consumo de energia (e recursos) nos dispositivos usados nas instâncias.

Referências

Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M., and Werthimer, D. (2002). Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45:56–61.

Andrzejak, A., Kondo, D., and Anderson, D. P. (2008). Ensuring collective availability in volatile resource pools via forecasting. In *Proceedings of the 19th IFIP/IEEE International Workshop on Distributed Systems*, DSOM '08, pages 149–161, Berlin, Heidelberg. Springer-Verlag.

Batista, C. E. C. F., de Araujo, T. M. U., dos Anjos, D. O., de Castro, M., Brasileiro, F., and Filho., G. (2007). Tvgrid: A grid architecture to use the idle resources on a digital tv network. In *Proc. 7th IEEE Intl Symposium on Cluster Computing and the Grid*, Rio de Janeiro, Brazil.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., and Mowbray, M. (2006). Labs of the World, Unite!!! *Journal of Grid Computing*, 4(3):225–246.

Cirne, W., Paranhos, D., Costa, L., Santos-Neto, E., and Brasileiro, F. e. a. (2003). *Running Bag-of-Tasks applications on computational grids: the MyGrid approach*. IEEE.

Costa, R., Bezerra, D. H. D., Vieira, D. A., Brasileiro, F., Sousa, D. M., and Filho, G. S. (2012). Oddci-ginga: A platform for high throughput computing using digital tv receivers. *IEEE/ACM International Conference on Grid Computing - GRID'12*, 0:155–163.

Costa, R., Brasileiro, F., de Souza Filho, G. L., and Sousa, D. M. (2009). Oddci: on-demand distributed computing infrastructure. In *2nd Workshop on Many-Task Computing on Grids and Supercomputers*, volume 16, pages 1–10, Portland, Oregon. ACM.

Costa, R., Brasileiro, F., Lemos, G., and Mariz, D. (2011a). Just in Time Clouds: Enabling Highly-Elastic Public Clouds over Low Scale Amortized Resources. In *3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011)*, Athens - Greece.

Costa, R., Brasileiro, F., Lemos, G., and Mariz, D. (2011b). Sobre a Amplitude da Elasticidade dos Atuais Provedores de Computação na Nuvem. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2011)*, Campo Grande - MS.

Fedak, G., Gelas, J.-P., Herault, T., Iniesta, V., Kondo, D., and Lefèvre, L. (2010). DSL-Lab: a platform to experiment on domestic broadband internet. In *9th Intl Symposium on Parallel and Distributed Computing (ISPDC'2010)*, Istanbul, Turkey.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. John Wiley and Sons.

Litzkow, M., Livny, M., and Mutka, M. (1988). Condor - a hunter of idle workstations. In *Proc. of the 8th International Conference of Distributed Computing Systems*, pages 104–111. IEEE.

Neill, R., Carloni, L. P., Shabarshin, A., Sigaev, V., and Tcherepanov, S. (2011). Embedded processor virtualization for broadband grid computing. In *Proc. of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID '11, pages 145–156, Washington, DC, USA. IEEE.

Varga, A. and Hornig, R. (2008). An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 60:1–60:10, Brussels, Belgium. ICST.