Planejamento Regional Adaptativo em Sistemas Self-Adaptive de Larga Escala

Eudes S. Andrade e Sandro S. Andrade
GSORT – Grupo de Pesquisa em Sistemas Distribuídos, Otimização, Redes e Tempo Real
IFBA – Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Av. Araújo Pinho, nº 39 – Canela – CEP: 40.110-150 – Salvador-BA
Email: {eudes.andrade, sandro.andrade}@ifba.edu.br

***Resumo*—Com a crescente adoção de sistemas distribuídos de larga escala, tais como *clusters* para computação em nuvem, as atividades de implantação e configuração destas soluções se tornam consideravelmente mais complexas. Uma solução promissora é dotar tais sistemas com capacidades de autogerenciamento. Para tanto, diversos padrões de projeto para Sistemas autoadaptativos estão disponíveis na literatura. Alguns são mais efetivos para operação em ambientes que sugerem a adoção de arquiteturas mais centralizadas. Outros apresentam melhores resultados ao operar de forma mais distribuída. Em ambiente altamente dinâmicos e incertos, no entanto, é difícil encontrar uma única solução efetiva para todos os cenários experimentados. Este trabalho apresenta um modelo adaptativo para implementação de Sistemas autoadaptativos de larga escala que realizam autogerenciamento com diferentes graus de centralização. Os resultados obtidos mostram que o modelo apresenta desempenho satisfatório tanto em situações onde o sistema monitorado/ambiente sugerem a adoção de topologias mais centralizadas, quanto em situações onde padrões mais descentralizados apresentam melhor custo-benefício.**

I. Introdução

Uma série de requisitos e características associados aos sistemas distribuídos modernos têm modificado a forma como tais aplicações são projetadas, desenvolvidas e avaliadas. Dentre tais características, destacam-se: a utilização de dispositivos multiprocessados, a disponibilidade de paralelismo em soluções altamente distribuídas, a necessidade de integração com outros sistemas e a capacidade de autogerenciamento em ambientes incertos [1]. Muitas dessas novas demandas são consequências do aumento de poder computacional apresentado pelos computadores modernos e pela crescente velocidade e confiabilidade das tecnologias de redes de comunicação. Essa transição dos sistemas da geração *petascale* para o mundo dos sistemas *exascale* [2] implica investimento em diversas áreas de pesquisa da Ciência da Computação.

Apesar de todas as estratégias disponibilizadas pela Engenharia de *Software* para gerenciamento da complexidade gerada por esses novos desafios, acredita-se que a capacidade humana de compreensão e manipulação de artefatos de *software* se mostrará como um fator limitante no futuro [3]. A complexidade dos produtos de *software* irá crescer continuamente até um ponto onde as tecnologias mais efetivas e os profissionais mais bem formados não serão capazes de construir soluções satisfatórias. [4]. Para ilustrar essa crescente complexidade, pode-se citar a grande quantidade de parâmetros para configuração dos bancos de dados atuais, as diversas possibilidades de configuração dos servidores *web* e as constantes adaptações sofridas pelos nós que formam as plataformas de computação em nuvem modernas.

Tendo em vista que os maiores desafios expostos decorrem de requisitos não-funcionais – e que estes têm sido historicamente tratados através de decisões de projeto – linhas de pesquisa modernas apontam para uma nova abordagem: a transferência de determinadas decisões de projeto para *runtime*. Para tanto, os sistemas computacionais deverão estar dotados de alguma capacidade de autogerenciamento [3], [4], [5].

De acordo com a DARPA (*Defense Advanced Research Projects Agency*) um sistema autoadaptativo (*self-adaptive* - SSA) é aquele que avalia o seu próprio comportamento e o modifica quando a avaliação indica que: i) o seu propósito principal não está sendo efetivamente cumprido; ou ii) uma melhor funcionalidade e/ou desempenho pode ser alcançado [6]. A autoadaptação é uma estratégia utilizada para os mais diversos fins, desde a otimização de desempenho em sistemas que operam sobre ambientes que mudam constantemente até situações mais extremas onde sistemas conseguem se recuperar após a falha de parte dos seus componentes [7].

Muitas arquiteturas para autogerenciamento estão atualmente disponíveis na literatura [8]. Ainda assim, a maioria delas apresenta desempenho satisfatório apenas em sistemas centralizados [7]. A dificuldade para se alcançar autogerenciamento efetivo em ambientes descentralizados decorre do *trade-off* comumente encontrado em sistemas distribuídos: conhecimento do estado parcial do sistema vs. custo computacional de comunicação. Pode-se ilustrar esse *trade-off* analisando *clusters* para suporte a computação em nuvem. Em tais situações, quanto maior o *overhead* de comunicação (e, portanto, o conhecimento sobre o estado global do sistema), maior será a capacidade do sistema coordenar adaptações em caso de mudanças no ambiente. Por outro lado, um maior consumo de recursos pode diminuir a utilização útil do *cluster* para fins de computação. O objetivo da adoção de arquiteturas para autogerenciamento descentralizado, em tais cenários, é reagir ao *trade-off* que envolve a capacidade de atendimento do *cluster* frente às diversas configurações de ambiente apresentadas ao longo do tempo.

Este trabalho apresenta o projeto e avaliação de um modelo

adaptativo para autogerenciamento em SSA de larga escala. Esse modelo prevê a reconfiguração dinâmica da topologia e forma de comunicação entre os múltiplos componentes locais do controle. Com esta abordagem, decisões de projeto acerca da topologia de controle a ser adotada serão levadas para o tempo de execução. Com isso, obtém-se desempenho satisfatório de controle tanto em situações onde o sistema monitorado/ambiente sugerem a adoção de topologias mais centralizadas, quanto em situações onde padrões completamente descentralizados apresentam melhor *trade-off*. O mecanismo de reconfiguração dinâmica da arquitetura de controle aqui proposto foi modelado e avaliado via simulação de eventos discretos. Resultados indicam que houve uma economia estatisticamente significante de recursos computacionais ao utilizar o modelo proposto.

O restante deste trabalho está organizado como segue. A Seção II apresenta os fundamentos sobre SSA e seus principais desafios. A Seção III explica o mecanismo para reconfiguração dinâmica de arquiteturas de controle aqui proposto. A Seção IV apresenta detalhes sobre as avaliações via simulação realizadas, enquanto a Seção V discute e analisa os resultados obtidos. A Seção VI apresenta os principais trabalhos relacionados e, finalmente, as conclusões e sugestões de trabalhos futuros são discutidos na Seção VII.

II. Sistemas Autoadaptativos de Larga Escala

Serão apresentadas nessa seção definições acerca de duas matérias: Sistemas Autoadaptativos e Padrões de Projeto para Sistemas Autoadaptativos. Em um primeiro momento, será discutida a motivação para a adoção de soluções auto-adaptivas e seus fundamentos básicos. Na seção seguinte, serão apresentados seis padrões de projeto para controle descentralizado em sistemas auto-adaptativos. Após o detalhamento dos atributos de qualidade inseridos por cada um dos seis padrões, serão trazidas à luz as restrições impostas por tais decisões.

A. Sistemas Autoadaptativos

O autogerenciamento é uma estratégia de projeto utilizada para os mais diversos fins. A despeito disto, a maior parte destas soluções realiza a migração de decisões – anteriormente eram tomadas em tempo de projeto – para o tempo de execução. Com este artifício, busca-se minimizar possíveis inflexibilidades/ineficiências decorrentes de comprometimentos prévios com determinadas decisões arquiteturais. A ocorrência de situações não previstas pelo arquiteto de *software* é consequência, sobretudo, do dinamismo dos ambientes de execução das aplicações.

Para ilustrar tal situação, podemos considerar um cenário onde uma solução não-adaptativa foi escolhida em detrimento de um algoritmo adaptativo. Um arquiteto de *software* pode, por exemplo, ser obrigado a fazer opção por uma determinada solução para *caching* e paralelismo em detrimento de outras. Apesar de essa decisão do arquiteto parecer inicialmente acertada, ela pode implicar perda de eficiência caso os padrões de acesso e/ou a natureza dos dados mudem. Ao dotar o *software* de capacidades de autogerenciamento o arquiteto poderia migrar a decisão de qual solução de *caching* e paralelismo será usada para tempo de execução. Dessa forma, soluções ótimas seriam adotadas para os diversos padrões de acesso e natureza de dados que o sistema estará exposto ao longo de sua execução.

Uma definição de SSA comumente adotada na literatura é aquela proposta em [6]. Em [6] foi definido que um SSA é aquele que avalia o seu próprio comportamento e o modifica quando esta avaliação indica que: *i)* o seu propósito principal não está sendo efetivamente cumprido; ou *ii)* uma melhor funcionalidade e/ou desempenho pode ser alcançado [6]. Um SSA é geralmente caracterizado pela presença de dois elementos arquiteturais principais, ilustrados na Figura 1. O primeiro – denominado **sistema gerenciado** – é o elemento responsável pela implementação das regras de negócio da aplicação. O segundo elemento – denominado **sistema gerenciador** – é responsável pela implementação das ações necessárias para realização das adaptações.

Embora diferentes mecanismos para autogerenciamento tenham sido propostos nos últimos anos, a maior parte deles baseia-se na utilização de algum tipo de *loop* de adaptação. Uma arquitetura de referência amplamente adotada no projeto destes *loops* de adaptação é a arquitetura MAPE-K [4]. Esta arquitetura define a execução, em sequência, das seguintes fases: **M**onitoramento, **A**nálise, **P**lanejamento e **E**xecução – com o auxílio de uma base de conhecimento (**K***nowledge Base*). A fase de monitoramento é responsável pela obtenção de informações sobre o estado atual do sistema/ambiente. A atividade de análise realiza a avaliação das informações obtidas na fase de monitoramento. Após a avaliação destes dados, delibera-se pela necessidade ou não da realização de uma adaptação. A fase de planejamento é responsável pela definição de quais atividades de adaptação devem ser realizadas, com o objetivo que o sistema retorne a um estado particular de interesse. Por fim, a atividade de execução realiza a implantação, no sistema gerenciado, das ações de adaptação definidas pela fase de planejamento.

Em ambientes altamente distribuídos, heterogêneos e complexos, um único *loop* de controle pode não ser suficiente para gerenciar toda a sua estrutura [7]. Nestes cenários, portanto, múltiplos *loops* devem ser empregados para gerenciar as diferentes partes (múltiplos sistemas gerenciados) que compõem o sistema. Esta associação de *loops* de controle aos diversos nós do sistema distribuído traz uma complexidade a mais durante o projeto da solução: a coordenação e interação entre estes múltiplos *loops*.

B. Padrões para Controle Descentralizado em SSA

Quando os sistemas são heterogêneos, grandes e complexos um único *loop* de controle pode não ser suficiente para gerenciar toda a adaptação [9], [10]. Nesses casos, múltiplos *loops* de controle podem ser empregados para gerenciar diferentes partes do sistema. Essa distribuição dos *loops* de controle nos diversos nós do sistema traz uma complexidade a mais para o *design* da solução: a coordenação na execução dos diversos *loops*. Dessa problemática surge uma importante decisão a

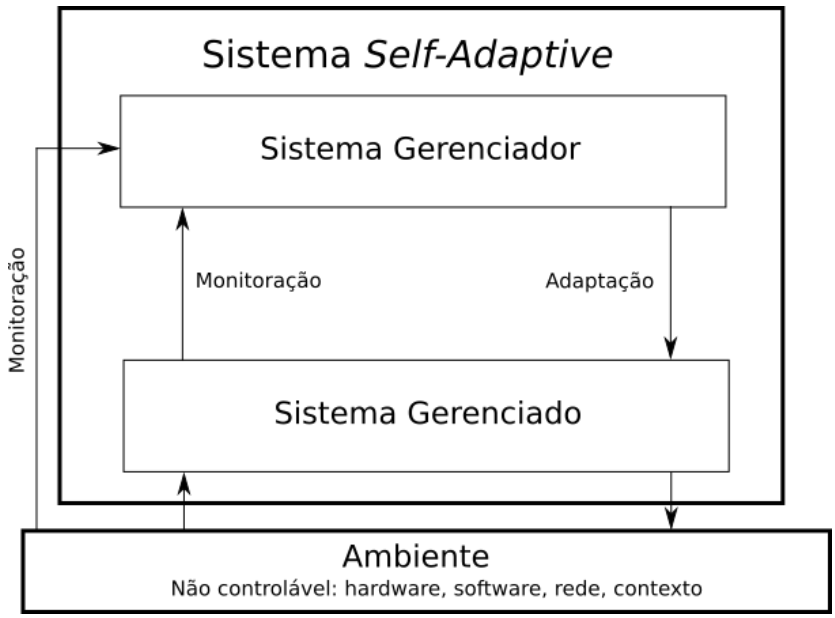

Figura 1. Elementos arquiteturais principais de um Sistema Self-Adaptive.

ser tomada em tempo de projeto: como se dará a cooperação entre os diversos *loops* de controle. Nesse momento, algumas questões emergem. Todas as fases dos controles MAPE (monitoração, análise, planejamento e execução) irão cooperar entre si em todo o sistema? Cada sistema gerenciador realizará isoladamente a fase de monitoração (sem cooperação) e as demais fases serão realizadas em um único *loop* de controle? A resposta para essas questões é apresentada em [7].

+De acordo com [7] diferentes padrões para implementação de *loops* de controle têm sido utilizados de maneira prática na indústria. O *Framework Rainbow*, por exemplo, distribui as fases de monitoração e execução sobre os diversos nós do sistema ao passo que mantem as fases de análise e planejamento centralizadas [11]. Por outro lado, os guias arquiteturais da IBM [12] organizam os *loops* MAPE hierarquicamente, mantendo em cada nível da hierarquia todas as quatro fases do *loop* MAPE.

Em [7], um conjunto de seis padrões arquiteturais para implementação de autogerenciamento descentralizado é apresentado. Para cada padrão, são apresentados os atributos de qualidade decorrentes da sua adoção, bem como as restrições impostas pelas decisões que compõem a solução. Um dos padrões apresentados é o Planejamento Regional. O padrão Planejamento Regional é recomendado para situações onde diversas partes integradas (regiões) de um sistema demandam não apenas adaptações locais (dentro das regiões) mas também adaptações globais (que extrapolam os limites das regiões). A solução proposta pelo Planejamento Regional prevê um único componente P (*planning*) para cada região. Este planejador regional é responsável por coletar as informações de todos os componentes que fazem parte da sua região, para que o planejamento seja realizado. Planejadores regionais coordenam entre si (sobre múltiplas regiões) para a realização do planejamento da adaptação global.

A Figura 2 apresenta uma instância do padrão Planejamento Regional, com diferentes tamanhos de região. Na figura, dois tipos de nós de controle são apresentados. O primeiro deles apresenta apenas o componente `P` (planejador da região) e cada região é controlada por apenas um nó do tipo `P`. O segundo tipo apresenta os outros três componentes MAPE-K restantes: `MAE` (Monitoramento, Análise e Execução). Em cada região, existem múltiplos nós de controle do tipo `MAE`. Eles realizam o monitoramento do seu sistema local e do seu ambiente de execução. Através dos componentes `A` realizam uma análise local dos dados monitorados e reportam os resultados para o planejador `P` da sua região. O planejador regional `P`, por sua vez, planeja a adaptação local da região após cooperar com os planejadores `P` das outras regiões. Após a conclusão do planejamento instruções para que a execução seja realizada são enviadas para os componentes `E`. Com este arranjo, a adaptação realiza a busca por adaptações globais, extrapolando os limites do monitoramento e análise locais.

Como consequência da adoção do padrão Planejamento Regional, espera-se uma redução na quantidade de dados trafegados uma vez que o monitoramento e análise são realizadas localmente – a análise local dos dados monitorados reduz a quantidade de dados e a frequência com que interações serão realizadas com o nó central da região. Por outro lado, a necessidade de agregação local – no nó de planejamento – dos resultados das diversas operações de análise pode implicar um custo computacional excessivo. Outra desvantagem é a necessidade de uma fase de planejamento demasiadamente detalhada, uma vez que as fases de execução não coordenam. Finalmente, a adoção do padrão Planejamento Regional requer a definição – em tempo de projeto – do tamanho de região com o qual o sistema será configurado. Consequentemente,

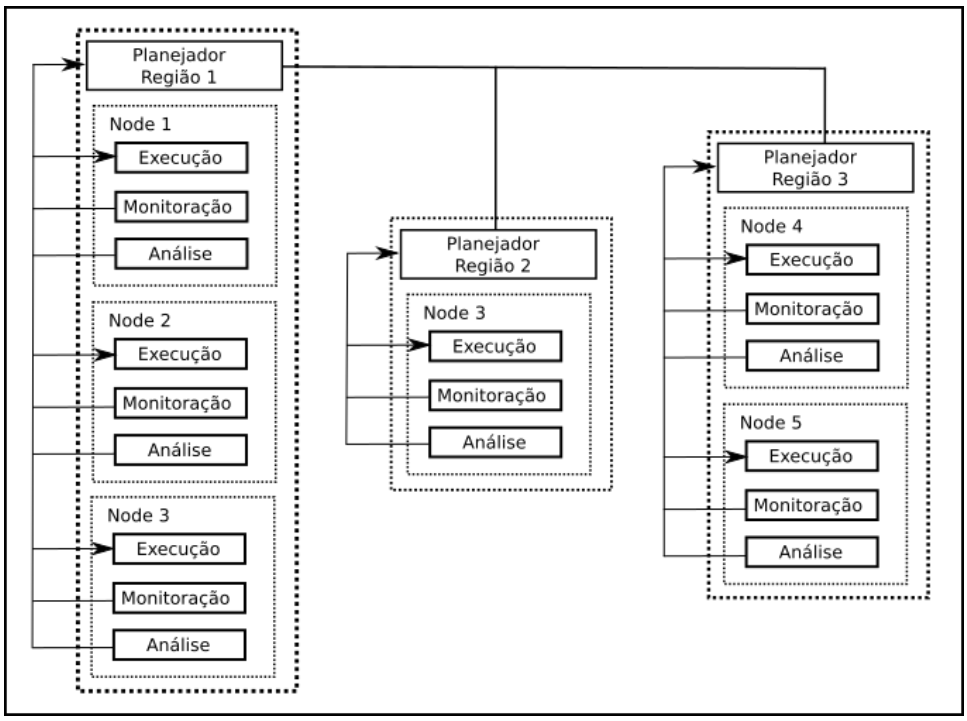

Figura 2. Instância do Padrão Planejamento Regional.

ambientes diferentes podem demandar diferentes tamanhos de região para que o autogerenciamento apresente desempenho satisfatório. Em ambientes altamente dinâmicos e incertos, autogerenciamento satisfatório só é obtido com a variação – em tempo de execução – do tamanho de região adotado.

III. O Mecanismo de Planejamento Regional Adaptativo

Este trabalho apresenta o projeto, implementação e avaliação de um modelo adaptativo para autogerenciamento em SSA de larga escala. O modelo permite a reconfiguração dinâmica da topologia do sistema gerenciador e da forma de comunicação entre os múltiplos componentes locais do controle. Com esta abordagem, decisões referentes à topologia de controle a ser adotada (tamanho da região de planejamento) são levadas para o tempo de execução. Com isso, espera-se obter desempenho satisfatório de autogerenciamento tanto em situações onde o sistema gerenciado/ambiente sugerem a adoção de topologias mais centralizadas, quanto em situações onde padrões completamente descentralizados são mais adequados. Em ambientes altamente dinâmicos e incertos, é comum que uma determinada topologia – adequada para um dado cenário – não seja mais satisfatória após a ocorrência de mudanças no ambiente.

Este trabalho viabiliza o projeto de sistemas gerenciadores de tal modo que eles mesmos sejam tratados como SSA. Dessa forma, decisões acerca dos limites das regiões em arquiteturas que seguem o padrão Planejamento Regional não mais são realizadas em tempo de projeto. Com o modelo proposto, em função do seu estado interno e das condições do ambiente, a arquitetura do sistema gerenciador será reconfigurada para adotar desde abordagens mais centralizadas até os extremos providos pelos padrões mais distribuídos.

A. Motivação

Conforme discutido na Seção II, a adoção do padrão Planejamento Regional apresenta um *trade-off* que envolve a qualidade da adaptação vs. a quantidade de dados trafegados (e localmente mantidos) nos nós responsáveis pelo planejamento. Este *trade-off* é comumente administrado pelo arquiteto de *software*, em tempo de projeto. Para tanto, os tamanhos das regiões são fixos e definidos antecipadamente, sobretudo considerando as situações de maior sobrecarga previstas para a operação do sistema.

Visto que no padrão Planejamento Regional o tamanho das regiões é definido em tempo de projeto, os sistemas gerenciadores tendem a operar de forma não-ótima quando as condições ambientais se desviam daquelas consideradas em tempo de projeto. Para situações de sobrecarga do sistema – onde um máximo de controle se faz necessário – os controles regionais atuam de maneira satisfatória. Por outro lado, em situações onde a utilização de tamanhos menores de região seria suficiente para garantir a qualidade de operação do sistema, um super provisionamento de recursos estará sempre presente.

A possibilidade de utilização de um modelo onde a decisão acerca do tamanho da região não mais seja uma definição a ser administrada em tempo de projeto – mas sim uma decisão levada para o tempo de execução – é o que motiva a adoção do Planejamento Regional Adaptativo proposto neste trabalho. Ao

utilizar um tamanho variável de região, o sistema gerenciador terá a possibilidade de operar de maneira ótima independente dos diferentes estados apresentados pelo ambiente e sistema gerenciado ao longo do tempo.

B. O Mecanismo Proposto

O Planejamento Regional Adaptativo aqui proposto implementa uma variação do padrão Planejamento Regional. Diferente do Planejamento Regional, o Planejamento Regional Adaptativo transfere para o tempo de execução a decisão acerca do tamanho das regiões. Dessa forma, o sistema gerenciador passa a ser, ele mesmo, um SSA que rejeita perturbações no ambiente através da variação no tamanho das regiões de planejamento.

Para isso, um segundo nível de controle (meta-controle) é inserido na arquitetura. Os tamanhos de região, que antes eram fixos e definidos em tempo de projeto para suportar os cenários de pior caso, passam a ser agora mantidos em tempo de execução pelo meta-controle. Com isso, espera-se obter desempenho satisfatório de autogerenciamento tanto em situações onde o sistema gerenciado/ambiente sugerem a adoção de regiões maiores (planejamento global), quanto em situações onde o uso de pequenas regiões (planejamento local) já é suficiente. Trata-se, portanto, de adicionar ao padrão Planejamento Regional um segundo nível de adaptação.

Em momentos onde o sistema gerenciado e o ambiente demandem menos da adaptação, o tamanho das regiões será diminuído pelo meta-controle. Dessa forma, recursos serão poupados. Por outro lado, em momentos onde o meta-controle verifica que o componente MAPE-K responsável pelo planejamento regional necessita de monitoramento e análises mais globais – de forma a garantir os requisitos de qualidade acordados – o tamanho das regiões é aumentado. A fundamentação para tal investigação envolve os impactos decorrentes da variação do tamanho das regiões. A hipótese aqui investigada considera que a adoção de grandes regiões traz como vantagem a maior presença de informações globais, o que resulta na possibilidade de realização de adaptações mais efetivas. Por outro lado, a adoção de regiões pequenas beneficia aspectos tais como disponibilidade e escalabilidade. Sendo assim, o objetivo geral da solução aqui apresentada é regular, de forma automática, o *trade-off* existente entre a efetividade das adaptações vs. a disponibilidade e escalabilidade da solução. A Figura 3 ilustra a reconfiguração dinâmica do tamanho das regiões definida pelo Planejamento Regional Adaptativo proposto.

Algumas premissas foram consideradas nesta proposta. Primeiro, assume-se que o *overhead* introduzido pelo meta-controle é desprezível. Visto que as operações de reconfiguração geralmente envolvem a modificação de parâmetros, tais como tamanho de *buffers* ou *thread pools*, transitar tais parâmetros na rede é consideravelmente menor que transitar os dados da aplicação. Segundo, todas as regiões apresentam tamanhos iguais entre elas, embora o tamanho mude ao longo do tempo. Embora tal premissa eventualmente não seja verificada em *clusters* com bastante heterogeneidade, suportar diferentes tamanhos de região traria uma complexidade maior ao modelo e será analisado em trabalhos futuros. Por fim, considera-se que o *cluster* possui tarefas sendo demandados a todo momento, o que viabiliza o contínuo monitoramento do desempenho das aplicação e do autogerenciamento.

IV. Avaliação

O mecanismo de Planejamento Regional Adaptativo aqui proposto foi modelado e avaliado via simulação de eventos discretos. Os resultados da simulação foram avaliados considerando dois diferentes grupos: o primeiro, decorrente da execução da simulação utilizando o Planejamento Regional Adaptativo proposto neste trabalho; e o segundo, utilizando o padrão Planejamento Regional convencional, com o uso de um tamanho fixo de região de planejamento. As seguintes hipóteses foram investigadas no experimento:

- Hipótese alternativa (H_1): mantida a mesma qualidade do serviço, o tamanho médio da região (e, portanto, o consumo de recursos) ao utilizar o Planejamento Regional Adaptativo aqui proposto (T_{PRA}) é menor que o tamanho médio da região ao utilizar o Planejamento Regional (T_{PR}).

$$\boxed{T_{PRA} < T_{PR}} \quad (1)$$

- Hipótese nula (H_0): mantida a mesma qualidade do serviço, o tamanho médio da região (e, portanto consumo de recursos) ao utilizar o Planejamento Regional Adaptativo aqui proposto (T_{PRA}) não é menor que o tamanho médio da região ao utilizar o Planejamento Regional convencional (T_{PR}).

$$\boxed{T_{PRA} > T_{PR}} \quad (2)$$

A. Simulação

O modelo da simulação define o sistema gerenciado, gerenciador e meta-controle como um único sistema distribuído. Este sistema distribuído é modelado como um grafo $E(N, C)$. O grafo E é composto por um conjunto finito de nós, denominado $N = \{N_1, N_2, \ldots, N_n\}$ e um conjunto finito de canais de comunicação $C = f(N_i, N_j)|N_i, N_j \in N$. Vértices representam nós (sistemas gerenciados e gerenciadores) e as arestas representam canais de comunicação. O grafo E implementa os detalhes referentes ao sistema simulado, tais como os componentes que simulam os nós do sistema, informações acerca das regiões e eventuais perturbações (P) presentes no ambiente.

Uma vez que os nós N têm o seu tempo de serviço afetado diretamente pela qualidade da adaptação, tem-se que o tempo de serviço das operações (TS) é função do tamanho das regiões (R). Além do tamanho das regiões, nota-se que o tempo de serviço das operações é diretamente influenciado pela perturbação (P) presente no ambiente. Dessa forma, tem-se que TS é função tanto do tamanho das regiões quanto da perturbação existente no ambiente: $TS(R, P)$. A perturbação (P) foi modelada através de uma variável aleatória, cuja intensidade é obtida utilizando a distribuição normal, ao passo que

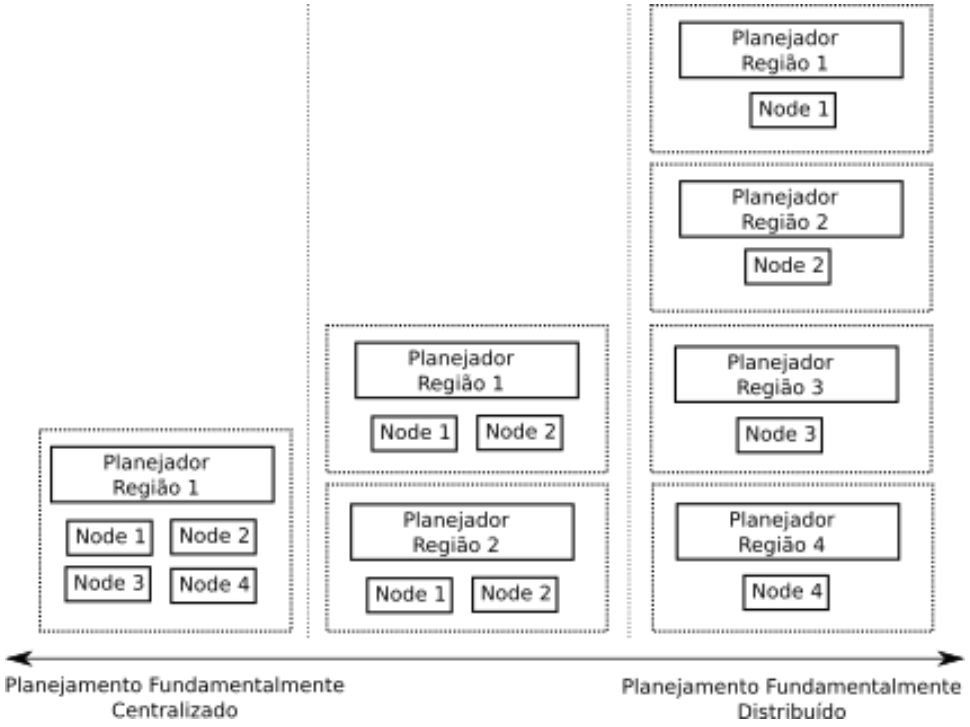

Figura 3. Reconfiguração dinâmica do tamanho das regiões.

sua frequência de ocorrência é denotada por uma distribuição exponencial.

Além das relações definidas na função $TS(R, P)$, uma variável aleatória K foi inserida no cálculo do tempo de serviço das operações para representar componentes estocásticos internos ao nó. Os valores de K são obtidos de acordo com a distribuição *Erlang* [13]. E armazena ainda a fila de operações a ser executada. O modelo proposto considera a existência de uma fila de operações constantemente alimentada e representa as tarefas demandadas por usuários. Os nós presentes no sistema atuam consumindo as operações que aguardam por processamento. O tempo de chegada de novas operações na fila foi modelado através de uma variável aleatória com distribuição exponencial.

Dois níveis de adaptação são observáveis no modelo proposto. O primeiro nível implementa o padrão Planejamento Regional e é responsável por garantir a adaptação nos nós. Esse controle é implementado na função que calcula o tempo de serviço das operações $TS(R, P)$. O tamanho das regiões (R) é a variável que representa a contribuição, do controle de primeiro nível, no tempo de serviço das operações. O segundo nível implementa a operação de meta-controle (MC), responsável por adaptar os controladores de primeiro nível, sobretudo atuando no tamanho das suas regiões. O MC atua no tamanho das regiões da seguinte forma: se a capacidade de processamento não utilizada – operação ociosa do processador, aqui definida como ciclos inativos – é maior que um determinado valor de referência, o tamanho das regiões é reduzido. Por outro lado, caso a quantidade de ciclos inativos mantenha-se abaixo do valor de referência, o MC atuará aumentando o tamanho das regiões. Apesar de o modelo não depender diretamente da lei de controle, para que a simulação pudesse ser executada fez-se necessário optar por uma implementação concreta. Nesta simulação, optou-se por utilizar controladores PID (Proporcional-Integral-Derivativo) [14] para realização da adaptação.

B. Implementação da Simulação

A simulação foi implementada utilizando a linguagem de programação *Python* e o *framework SimPy* (http://simpy.readthedocs.org). O SimPy é um *framework* para suporte à implementação de simuladores de eventos discretos baseados em processos. Requisitos como geração de tarefas, gerenciamento de perturbações, cálculo do custo computacional da execução de processos e gerenciamento do tamanho das regiões foram implementados e monitorados durante a execução da simulação. Pacotes adicionais foram utilizados para geração dos números aleatórios de acordo com as distribuições definidas no modelo.

Todas as operações de entrada e saída da simulação foram implementadas através de arquivos *yaml*. Os parâmetros de configuração da simulação foram definidos através do arquivo *simconfig.yaml*. Dentre os parâmetros de configuração da simulação destacam-se:

- *DEFAULT_REGION_SIZE*: define o tamanho padrão das regiões para as replicações que ocorrem no nível do padrão Planejamento Regional.
- *NODES_NUMBER*: define o número de nós existentes na simulação.
- *SIMULATION_TIME*: define a duração da simulação.

- *MEAN_DISTURBING_LEVEL*: define o valor médio do nível das perturbações verificadas no ambiente.
- *DEVIATION_DISTURBING_LEVEL*: define o desvio padrão das perturbações verificadas no ambiente.
- *JOB_COST_SHAPE*: define o *Shape* da distribuição *Erlang* que implementa o componente *estocástico* internos aos nós do *cluster*.
- *JOB_COST_RATE*: define a escala da distribuição *Erlang* que implementa o componente *estocástico* internos aos nós do *cluster*.

As informações de saída da simulação são persistidas em arquivos de relatório no formato *yaml*. Nesses relatórios encontram-se informações acerca das diversas atuações do meta-controle como, por exemplo, valor da atuação do controle e o valor da variável controlada. Os diversos níveis das perturbações ao longo do tempo também são verificados nesse relatório. A despeito de todas as outras informações existente no relatório destacam-se a quantidade de tarefas executadas em cada nó do *cluster* e a quantidade de ciclos inativos verificados nos processadores. Essas duas informações são de fundamental importância uma vez que elas serão as variáveis consideradas no estudo estatístico que validará o modelo proposto. A figura 4 apresenta uma amostra do relatório gerado pela simulação. O diagrama de classes apresentado na figura 5 denota o *design* adotado na implementação do simulador.

O código-fonte do simulador pode ser encontrado em https://github.com/oides/regional-adaptation-simulation.

C. Análise dos Resultados

A análise dos dados foi realizada com base em um estudo comparativo com dois diferentes grupos. O primeiro grupo representou a execução da simulação utilizando o padrão Planejamento Regional convencional. O segundo grupo, por sua vez, representou a execução da simulação com a utilização do Planejamento Regional Adaptativo proposto neste trabalho. O experimento apresentou um único fator: o tamanho das regiões em tempo de execução. Esse único fator apresenta dois diferentes níveis: Planejamento Regional Adaptativo (PRA) ou Planejamento Regional convencional (PR).

Para cada nível foram executadas cinquenta replicações utilizando as mesmas sementes de geração de valores para as variáveis aleatórias. A quantidade de ciclos inativos foi medida para cada uma das replicações. Os resultados das cinquenta replicações foram inicialmente analisados com os testes de *Anderson-Darling* e *Levene*, com o objetivo de analisar, respectivamente, a normalidade e homoscedasticidade dos resultados. Com base nos resultados destes testes, decidiu-se pela avaliação da hipótese nula utilizando ou testes paramétricos (*t-test*) ou testes não-paramétricos (*Wilcoxon*).

A análise dos resultados foi realizada utilizando a ferramenta `R` [1]. `R` é, além de uma linguagem, um ambiente de desenvolvimento integrado, fundamentalmente utilizado para implementação de cálculos estatísticos e gráficos.

[1]http://www.r-project.org

Conforme descrito na seção IV-C, fez-se necessário a execução de quatro diferentes testes para que a significância estatística dos resultados fosse analisada. Para tanto, optou-se por utilizar as bibliotecas `R` que seguem para implementação dos testes citados:

- Teste de *Levene*: biblioteca *lawstat*, responsável por avaliar a homoscedasticidade dos grupos avaliados.
- Teste de *Anderson-Darling*: biblioteca *fBasics*, responsável por avaliar a homoscedasticidade dos grupos avaliados.
- Teste T: biblioteca padrão do ambiente `R`, responsável por implementar um teste paramétrico para análise de diferença não-nula de média de populações.
- Teste de *Wilcoxon*: biblioteca *MASS*, responsável por implementar um teste não-paramétrico para análise de diferença não-nula de média de populações.

Um mecanismo de otimização de execução da simulação e da análise de resultados foi implementado em *shell script*. Através desses *scripts* viabilizou-se a integração dos resultados da execução da simulação implementada em *Python* com a execução da análise dos resultados utilizando a tecnologia `R`.

O código-fonte da análise dos resultados pode ser encontrado em https://github.com/oides/regional-adaptation-simulation.

V. Resultados

Para realização dos testes considerando o nível PR (tamanho fixo das regiões) foi necessário definir um tamanho fixo para as regiões. Este tamanho foi o mínimo necessário para que o controle tivesse condições de adaptar o sistema satisfatoriamente na situação de pior caso. Uma vez que o tempo de serviço TS é função do tamanho da região e do nível de perturbação verificado em dado instante, calculou-se o tamanho mínimo de região necessário para suportar o ambiente nos momentos de maior perturbação possível. A perturbação gerada no ambiente é uma variável aleatória com distribuição normal. Logo, considerou-se como perturbação máxima o seu valor médio somado ao erro máximo (valor de referência - valor observado). A segunda execução da simulação foi realizada considerando o nível PRA (tamanho variável das regiões). A configuração inicial da simulação inicia o tamanho da região com o mesmo valor utilizado no nível PR, porém, com o decorrer da simulação, o tamanho da região é administrado pelo meta-controle.

Após a execução da simulação para o nível PR considerando as cinquenta replicações chegou-se na seguinte quantidade de ciclos *idle* do sistema:

Tabela I
Resultados do Planejamento Regional

Replicações									
223	299	336	211	140	150	195	153	126	198
208	203	206	232	260	239	137	187	108	194
219	145	192	245	181	122	136	233	178	174
281	256	178	123	153	131	154	144	139	141
135	99	152	72	149	113	193	159	165	241

```
actuations:
- {actuation_value: -1, controlled_variable: 16, current_time: 20}
- {actuation_value: 2, controlled_variable: 13, current_time: 40}
- {actuation_value: 3, controlled_variable: 14, current_time: 60}
disturbings:
- {currentTime: 0, disturbingDuration: '47', disturbingLevel: '16'}
- {currentTime: 47, disturbingDuration: '49', disturbingLevel: '21'}
- {currentTime: 96, disturbingDuration: '45', disturbingLevel: '22'}
idle_cycles: 101
jobsExecutedOnNodes:
- {jobsExecuted: 189, nodeId: Node 1}
- {jobsExecuted: 194, nodeId: Node 2}
- {jobsExecuted: 156, nodeId: Node 3}
jobs_costs:
- {current_time: 5, job_cost_value: 13}
- {current_time: 10, job_cost_value: 11}
- {current_time: 15, job_cost_value: 12}
- {current_time: 20, job_cost_value: 15}
- {current_time: 35, job_cost_value: 15}
- {current_time: 50, job_cost_value: 18}
- {current_time: 68, job_cost_value: 16}
- {current_time: 84, job_cost_value: 14}
```


Figura 4. Relatório de saída da simulação.

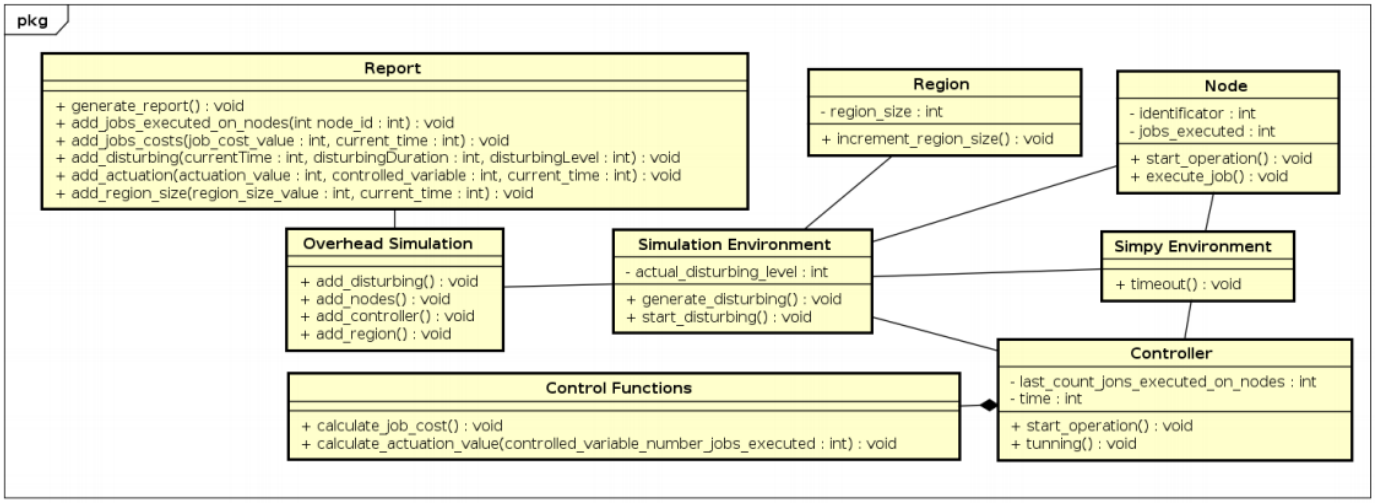

Figura 5. Diagrama de Classes do Simulador.

Uma segunda execução da simulação foi realizada considerando o nível PRA. Após a execução das cinquenta replicações chegou-se na seguinte quantidade de ciclos *idle* do sistema:

Tabela II
RESULTADOS DO PLANEJAMENTO REGIONAL ADAPTATIVO

Replicações									
108	102	103	108	109	107	105	104	101	100
98	101	99	104	103	97	98	101	100	98
111	122	109	105	104	106	99	104	102	103
98	97	100	102	98	99	96	105	102	101
100	107	108	101	98	101	107	103	102	101

Uma investigação acerca da normalidade e homoscedasticidade dos dados foi realizada, a fim de decidir se a hipótese nula do experimento poderia ser analisada através de um teste paramétrico (*t-test*).

O resultado da análise de normalidade foi obtido através do teste de *Anderson-Darling*. Com um nível de significância $\alpha = 0.01$, o resultado da análise de normalidade apresentou um $p - value$ igual a 0.78. Sabe-se que o teste de *Anderson-Darling* rejeita a hipótese de normalidade quando o $p - value$ é inferior a 0.05, logo, uma vez que a normalidade não foi rejeitada, seguiu-se para a análise de homoscedasticidade.

O resultado da análise de homoscedasticidade foi obtido através do teste de *Levene*. O teste apresentou como resultado um *p-value* igual a 0.003. Sabe-se que o teste de *Levene* rejeita a hipótese de igualdade das variâncias quando o $p - value$ é inferior a 0.05. Do resultado anterior conclui-se que o teste de *Levene* rejeita a igualdade das variâncias.

Apesar de a análise de normalidade dos resultados ter apresentado um resultado positivo, a hipótese de igualdade das variâncias foi rejeitada. Sabe-se que ambos os testes são condições necessárias para que a análise de diferença não-nula

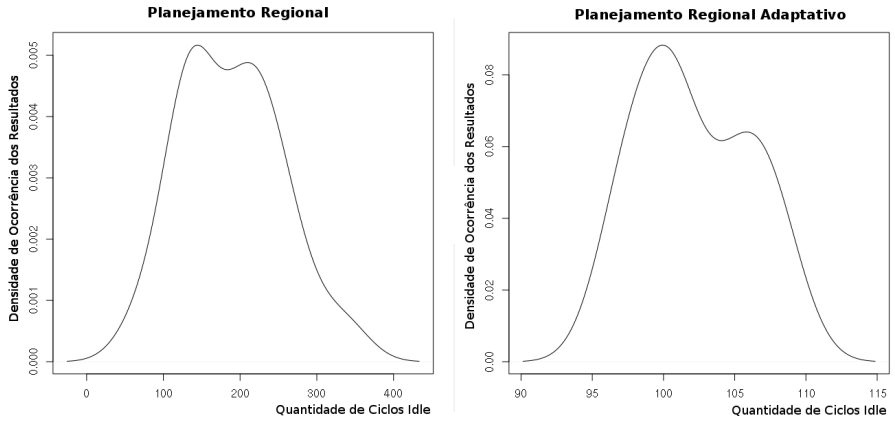

Figura 6. Distribuição de Densidade dos Resultados da Simulação.

de médias seja realizada através de testes paramétricos. Logo, a análise dos resultados não pode ser realizada através de *t-teste*, uma vez que este é um teste paramétrico. Optou-se então por realizar a análise das diferenças das medias utilizando o teste de *Wilcoxon*.

A Figura 6 apresenta a distribuição de densidade dos resultados da simulação.

O resultado do teste de *Wilcoxon* para as amostras anteriores apresentou, com um nível de significância $\alpha = 0.01$, um *p-value* igual a 0.0002. O resultado obtido indica que a hipótese nula H_0 pode ser rejeitada e, portanto, que as amostras avaliadas apresentam melhora significativa dos resultados. Uma representação gráfica do estudo comparativo dos resultados é apresentada na Figura 7.

VI. Trabalhos Relacionados

Diferentes padrões de projeto podem ser encontrados na literatura para tentar resolver o problema do planejamento [15], [7]. Um destes padrões de projeto é o "*information sharing*", que restringe a comunicação entre os componentes presentes no padrão descentralizado. Nesta organização, apenas o componente de monitoramento `M` pode se comunicar com outros componentes externos. Esta abordagem resulta em soluções com melhor escalabilidade de comunicação do que o controle totalmente descentralizado, agilizando o processo de execução da adaptação. Porém, este padrão não é indicado para ambientes altamente dinâmicos pois, como as análises e planejamentos são feitos localmente. Outra possibilidade é o padrão de projeto denominado controle hierárquico. Esta abordagem define uma separação de *concerns* por camadas, onde cada nível executa seu próprio *loop* MAPE-K. Os níveis inferiores, com seus componentes de monitoramento e execução, lidam diretamente com o subsistema gerenciado. A camada mais alta se preocupa com a adaptação geral do sistema. O padrão de controle hierárquico permite gerenciar a complexidade do SSA. Porém, esta abordagem não é indicada para ambientes em constante mudança pois a divisão em camadas permite que existam dois planejamentos de adaptação, sendo eles global e específico.

VII. Conclusão

Este trabalho apresentou o Planejamento Regional Adaptativo – uma extensão do padrão arquitetural Planejamento Regional que promove uma economia de recursos computacionais e menor necessidade de provisionamento de ambiente em SSA de larga escala. Diferente do padrão Planejamento Regional, o modelo proposto dotou, tanto o sistema gerenciador quanto o sistema gerenciado, de capacidades de autogerenciamento. Com isso, viabiliza-se a execução do sistema gerenciador utilizando tamanhos reduzidos de regiões, o que implica em uma economia de recursos. O mecanismo foi modelado e avaliado via simulação de eventos discretos. Dois grupos foram comparados: grupo de controle utilizando o padrão Planejamento Regional convencional e um segundo grupo utilizando o Planejamento Regional Adaptativo proposto. A análise dos resultados indica que, com um nível de significância $\alpha = 0.01$, houve um aumento de desempenho do sistema gerenciador ao utilizar o Planejamento Regional Adaptativo.

Como trabalho futuro verifica-se a possibilidade de ampliação do modelo da simulação, sobretudo inserindo como variáveis do modelo as premissas utilizadas nesse trabalho. A realização de experimentos controlados utilizando implementações reais do Planejamento Regional Adaptativo também mostra-se como um interessante trabalho futuro a ser realizado.

Agradecimentos

Concluir esse trabalho não foi fácil. Não foi fácil sobretudo por conta do envolvimento, seriedade e cobrança dos professores do GSORT. E é justamente por toda essa dedicação e seriedade que eu gostaria de agradecer a cada um dos professores da Especialização em Computação Distribuída e Ubíqua do IFBA. Obrigado Sandro Andrade. As aulas sobre arquitetura e conversas sobre método para produção de conhecimento científico em engenharia de *software* foram impecáveis. Obrigado Allan Freitas. É sempre uma honra ser seu aluno, desde a graduação. Obrigado Manoel Neto. Entender que tecnologia vai muito além do *software* que

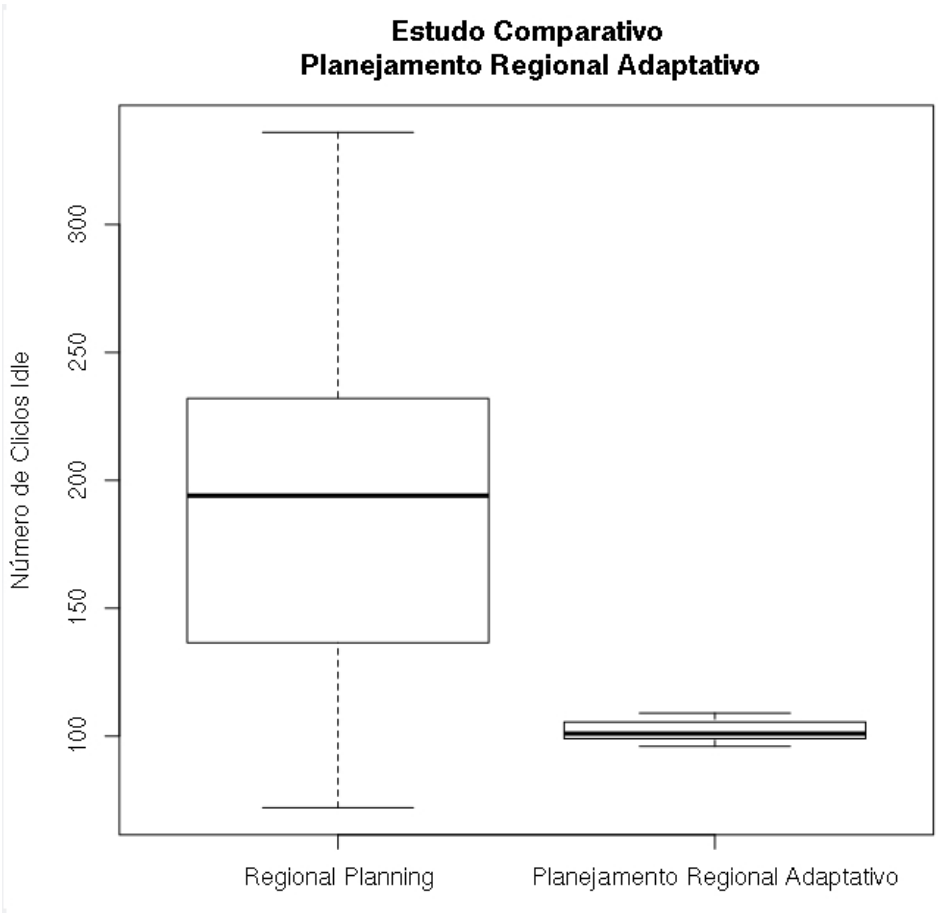

Figura 7. Resultado comparativo dos padrões Planejamento Regional e Planejamento Regional Adaptativo.

escrevemos e queimar um pouco os dedos com componentes eletrônicos foi muito divertido. Obrigado Flávia Nascimento. Obrigado Renato Novais.

Mais do que conhecimento científico ou um título, a Especialização em Computação Distribuída e Ubíqua do IFBA me permitiu estreitar os laços com uma pessoa muito especial. Uma pessoa que transformava aulas longas e densas, não obstante todo o meu cansaço após dias de trabalho, no momento mais divertido do meu dia. Um professor com um conhecimento imenso, que despertou em mim o prazer em estudar redes de computadores (eu achava que isso jamais aconteceria!). O professor, sem dúvidas, mais querido por toda a turma. Um mestre. Se um dia eu me tornar cientista e professor, espero conseguir ajudar meus alunos e orientandos como o professor Romildo nos ajudou. Muito obrigado por tudo, sobretudo por sua alegria, Professor Romildo Martins.

REFERÊNCIAS

[1] L. M. Northrop, "Does scale really matter? ultra-large-scale systems seven years after the study (keynote)," in *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, D. Notkin, B. H. C. Cheng, and K. Pohl, Eds. IEEE / ACM, 2013, p. 857.

[2] K. A. Yelick, "Programming models for petascale to exascale," in *22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, Miami, Florida USA, April 14-18, 2008*. IEEE, 2008, p. 1.

[3] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing - degrees, models, and applications," *ACM Comput. Surv.*, vol. 40, no. 3, 2008.

[4] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[5] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *TAAS*, vol. 4, no. 2, 2009.

[6] B. A. A. DARPA, "Self-adaptive software," DARPA (Defense Advanced Research Projects Agency), Tech. Rep. 98-12, 1997.

[7] D. Weyns, "On patterns for decentralized control in self-adaptive systems," in *Software Engineering for Self-Adaptive Systems II*, ser. Lecture Notes in Computer Science, R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds., vol. 7475. Springer, 2010, pp. 76–107.

[8] T. Patikirikorala, A. Colman, J. Han, and L. Wang, "A systematic survey on the design of self-adaptive software systems using control engineering approaches," in *Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, june 2012, pp. 33 –42.

[9] J. Andersson, R. Lemos, S. Malek, and D. Weyns, "Software engineering for self-adaptive systems," B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, ch. Modeling Dimensions of Self-Adaptive Software Systems, pp. 27–47.

[10] B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, "Software engineering for self-adaptive systems," B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, ch. Software Engineering for Self-

Adaptive Systems: A Research Roadmap, pp. 1–26.
[11] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, Oct. 2004. [Online]. Available: http://dx.doi.org/10.1109/MC.2004.175
[12] IBM, "An architectural blueprint for autonomic computing," 2006.
[13] R. Jain, "The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling," *SIGMETRICS Performance Evaluation Review*, vol. 19, pp. 5–11, 1991.
[14] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.
[15] Y. Abuseta and K. Swesi, "Design patterns for self adaptive systems engineering," *CoRR*, vol. abs/1508.01330, 2015. [Online]. Available: http://arxiv.org/abs/1508.01330