Proposta de uma Plataforma de Sistemas Multiagentes para Suportar a Gerência Autônoma de Recursos em Ambientes de Computação em Nuvem

Alexandre de Limas Santana[1], **Lucas Berri Cristofolini**[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

alexandre.limas.santana@gmail.com, lucas.cristofolini@grad.ufsc.br

Abstract. *Considering the complexity, heterogeneity and dynamism presented on cloud computing environments, performing an optimization in such environments turns into a challenging task. This work is driven by the similarities between cloud computing and multiagent systems paradigms and aims to propose the addition of a multi-agent systems platform into an existing orchestration tool, thereby enabling autonomous agents to handle the analysis, planning and management of cloud computing environments.*

Resumo. *Dada a complexidade, heterogeneidade e dinamismo crescentes presentes nos ambientes de computação em nuvem, otimizar a utilização de recursos nestes ambientes torna-se uma tarefa desafiadora. Motivado pela semelhança entre entre os paradigmas de computação em nuvem e sistemas multiagentes, este trabalho se propõe a adaptar uma ferramenta de orquestração existente, de modo a utilizar uma plataforma de sistemas multiagentes para possibilitar que agentes inteligentes realizem a análise, o planejamento e a gerência em ambientes de computação em nuvem de forma independente e autônoma.*

1. Introdução

Computação em nuvem (CN) é um paradigma tecnológico que vem chamando a atenção dos provedores de serviços, por propor uma mudança na forma de disponibilizar seus produtos [Armbrust et al. 2010]. A grande aceitação aos serviços de CN vista recentemente se deve, entre outros motivos, à não necessidade de um grande investimento inicial, sendo que os clientes que buscam esse serviço pagam apenas pelo que utilizam [Ibrahim et al. 2011].

Os benefícios da CN estão altamente ligados com a *Quality of Service* (QoS - Qualidade de Serviço) percebida pelos usuários. A necessidade de entregar QoS aos consumidores exige que recursos sejam mantidos ociosos a espera de cargas estocásticas, consumindo mais energia e influenciando diretamente nos custos de manutenção de um ambiente na nuvem [Weingärtner et al. 2015]

Uma alternativa para reduzir este consumo desnecessário consiste em agrupar as máquinas virtuais ativas no menor número de servidores físicos, possibilitando desligar os recursos ociosos e ligá-los quando a demanda torná-los necessários [Awada et al. 2014]. Entretanto, dada a falta de suporte a medidas de gerenciamento energético nas ferramentas

de orquestração de ambientes de CN disponíveis atualmente, [Bräscher 2015] propôs a adoção de um *framework* para a consolidação de recursos em ambientes de CN, visando uma melhor eficiência energética do ambiente orquestrado.

Problemas como este, onde não se conhece uma fórmula para obter a melhor solução e onde a informação está distribuída pelo ambiente, podem ser tratados através de uma abordadem de sistemas multiagentes (SMA). Dessa forma, é possível trabalhar sob diferentes pontos de vista acerca do problema, trazendo o conhecimento de especialistas para os agentes do ambiente, que por sua vez atuarão para alcançar seus objetivos [Silveira 2006].

2. Proposta

Devido ao crescimento da popularidade e da oferta de serviços de computação em nuvem, seja a nível de SaaS, PaaS ou Iaas, a gerência dos ambientes utilizados para fornecer estes serviços tem se tornado um tópico ativo de pesquisa, buscando-se métodos de manter e distribuir recursos computacionais em ambientes de CN de forma mais eficiente [Whitney and Delforge 2014]. Recentemente pode-se encontrar trabalhos que propõem modelos de sociedades de agentes para realizar tal função de maneira autônoma e distribuida tais como [Hou et al. 2014] e [De la Prieta et al. 2013], constatando assim o interesse de pesquisas no casamento dos paradigmas de SMA e CN.

Este trabalho busca expandir a solução implementada por [Bräscher 2015], adaptando a arquitetura inicialmente proposta introduzindo uma plataforma SMA, que por sua vez estará ligada a ferramenta de orquestração. Assim, também abre-se a possibilidade de que a ferramenta de orquestração possa futuramente fazer uso de agentes alheios aos processos propostos em [Bräscher 2015].

Analisando o modelo de componentes de uma ferramenta de orquestração proposto em [Bräscher 2015] na Figura 1, nota-se, em verde, as modificações realizadas por seu trabalho. Estas mudanças são a adição de um novo componente chamado **gerente de consolidação** e a modificação do módulo já existente **gerente de alocação**. Tais ajustes são oriundos das adições de seus gerentes autônomos cujos objetivos são a consolidação e re-alocação de máquinas dentro do ambiente de CN. Para que tais gerentes possam ser modelados na forma de agentes inteligentes é necessário que seja providenciado um local onde esses possam residir dentro do ambiente. Através do uso de uma plataforma de SMA para alocar os agentes, consegue-se realizar a desacoplação dos gerentes da ferramenta de orquestração, implicando em um melhor processo de escalabilidade à solução existente.

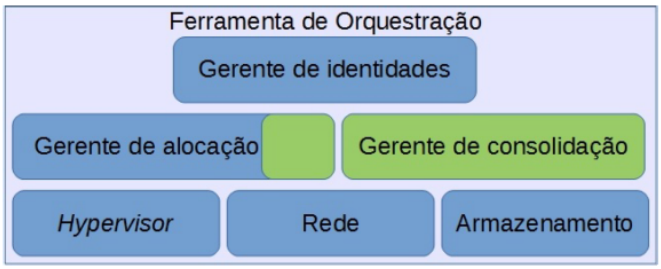

Figura 1. Componentes de uma ferramenta de orquestração expandida pelo *framework*, retirado de [Bräscher 2015].

Os agentes inteligentes usados por uma ferramenta de orquestração podem ser requisitados em vários segmentos desta. A partir da inclusão da plataforma de SMA, pode-se unificar o acesso a eles através da criação de uma interface visível à ferramenta de orquestração, que por sua vez se conecta à plataforma SMA e seus agentes. Uma abordagem desse tipo adiciona um componente a mais na arquitetura da ferramenta de orquestração, porém mantem a lógica dos agentes desacoplada. Para alcançar esse objetivo, este trabalho propõe uma mudança nos componentes da ferramenta de orquestração afim de adicionar um novo módulo, o gerente de SMA. Este novo módulo tem o objetivo de servir de fachada para a plataforma de SMA, garantindo acesso para os nodos de controle da ferramenta de orquestração.

2.1. Arquitetura do Gerente de SMA

Segundo a Figura 1, o uso do *framework* proposto por [Bräscher 2015] incrementa o número de elementos básicos de uma ferramenta de orquestração de 5 para 6. A primeira mudança proposta por este trabalho é renomear o gerente de consolidação para agente de gerência, devido ao fato de que o gerente de consolidação em [Bräscher 2015] não se restringe apenas a executar a consolidação, mas depende apenas das heurísticas apontadas a ele. Sendo assim, este agente trabalha de forma a realizar essas atribuições dentro de um *cluster*, tal como distribuição de cargas entre servidores.

Existe também a necessidade de um novo módulo com o propósito de manter a comunicação da ferramenta de orquestração com a plataforma de SMA. Este módulo, chamado de gerente de SMA pode ser visto na Figura 2, onde estão representados os elementos da ferramenta de orquestração proposto por este trabalho, destacando em um gradiente mais escuro os módulos adicionados e modificados.

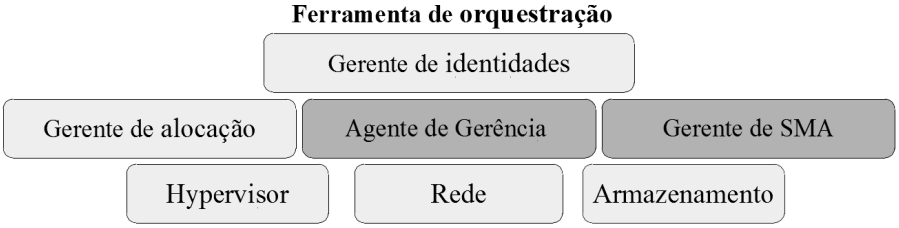

Figura 2. Componentes de uma ferramenta de orquestração incrementados pelo gerente de SMA.

A atribuição do novo componente é gerenciar a plataforma de SMA, garantido o ciclo de vida de todos os componentes distribuídos que compõem a plataforma de SMA. Desta forma, a ferramenta de orquestração pode manter-se alheia a esses processos e assumir que enquanto ela própria estiver operando, a plataforma de SMA está instanciada. Para fazer esta garantia, são adicionadas duas abstrações ao componente de gerência de SMA:

1. Plataforma de SMA: é a abstração dos estados, componentes ativos e aspectos da plataforma de SMA do ambiente;
2. Gerente de plataforma: uma entidade que responsabiliza-se por garantir a atividade, acessibilidade e tolerância a faltas da plataforma de SMA.

Sendo o gerente de plataforma uma parte integrante da ferramenta, este deve servir para comunicar a plataforma de SMA com o restante do ambiente. Dessa maneira, a ferramenta de orquestração e a plataforma de SMA podem manter-se desconexas, de forma que a ferramenta de orquestração não saiba que há agentes operando em seus recursos. A escolha de não acoplar a plataforma diretamente é uma maneira de garantir a modularidade do gerente de SMA. Sendo assim, quando outros segmentos da ferramenta de orquestração sentirem a necessidade de sociedades de agentes, estes poderão ser adicionados a plataforma sem grandes mudanças na ferramenta de orquestração. Caso também seja constatado que há a necessidade de trocar a própria tecnologia da plataforma de SMA, esta mudança refletirá em alterações apenas no gerente de SMA. A Figura 3 expande a visão deste componente, ilustrando a conexão de suas entidades com o restante da ferramenta de orquestração.

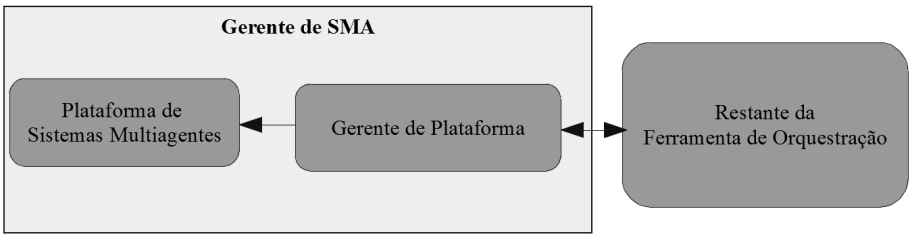

Figura 3. Relações entre componentes do gerente de SMA e elementos de uma ferramenta de orquestração.

2.1.1. Gerente de Plataforma

A plataforma multiagente usada nesse trabalho é a ferramenta JADE, escolhida por implementar o padrão FIPA, permitir que a plataforma seja distribuida e por oferecer tolerância a faltas através de redundância de seus nodos, além do fato de permitir a instancia de agentes reativos e BDI. A plataforma JADE funciona como uma central de comando para os agentes, sendo os processos de criação, comunicação e gerência das atividades desses realizado pela própria API da plataforma.

O JADE é composto por *containers* que precisam ser instanciados em uma JVM. Para facilitar a integração com a ferramenta de orquestração, os *containers* JADE são instanciados em uma *Virtual Machine* (VM - Máquina Virtual) dedicada a funções de sistema dentro do ambiente de CN sob o controle de alguma ferramenta de orquestração. A instalação do JADE, assim como de todas suas dependências nessa VM deve ocorrer de forma automática, sem ser necessária a interferência de um administrador. Para isso, são utilizadas funções implementadas na ferramenta de orquestração extendida em [Bräscher 2015] para instanciar uma máquina virtual contendo uma *Java Virtual Machine*, necessária para o funcionamento do JADE.

O gerente de plataforma é a entidade responsável por fazer os pedidos de recursos que são necessários para executar o JADE, implementando desde a obtenção de VMs de sistema até a instalação dos requisitos do JADE. Para que a criação de um *container* seja justificada o gerente de plataforma deve autonomamente perceber o estado atual do ambiente afim de decidir sua próxima ação. Para que isto seja possível, esse módulo deve

comunicar-se com os *containers* JADE periodicamente. Dependendo de quais *containers* estão ativos, pode ser percebebido a necessidade da criação de novos nodos.

Durante o processo de percepção do ambiente de nuvem, o gerente de plataforma deve encontrar *containers* que não estão funcionando adequadamente. Cabe a esse módulo detectar inconsistências na plataforma e corrigi-las. Através da destruição de nodos, as máquinas virtuais associadas a estes ficam sem propósito. Sendo assim, estas VMs são desalocadas através da ferramenta de orquestração. Pode-se argumentar que tais VMs podem ser usadas para receber novas instâncias de *containers* da plataforma. Sendo assim o ciclo de vida do gerente de plataforma pode ser visualizado na Figura 4.

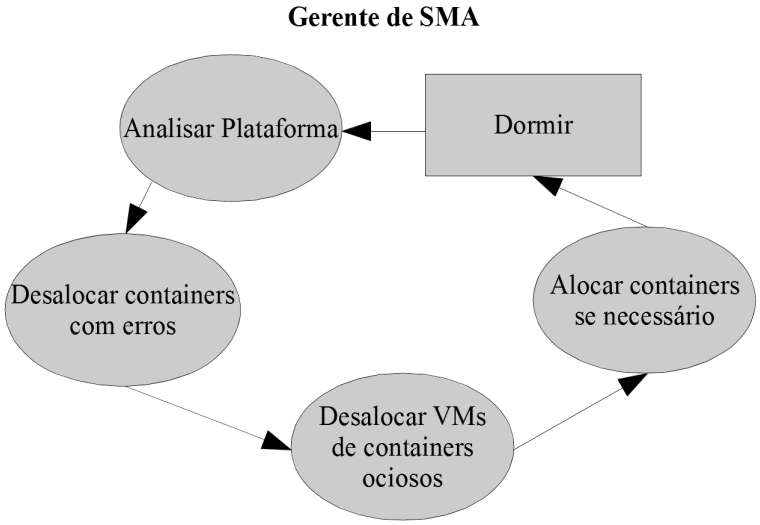

Figura 4. Ciclo de funcionamento do gerente de plataforma.

Com a gerência da plataforma garantida através dos componentes do gerente de SMA, é obtida a estabilidade e acessibilidade a plataforma de SMA. Sendo assim, resta apenas o controle do ciclo de vida dos agentes e modos para criação destes. Uma vez que a ferramenta de orquestração não deve perceber os agentes, esta não há de instanciar nenhum. A responsabilidade por detectar que agentes são necessários dentro do âmbito da ferramenta de orquestração, criá-los e destruí-los cabe a um agente que reside dentro da plataforma SMA, nomeado de agente inspetor.

2.2. Agentes do Sistema de Gerência

Uma vez que o funcionamento da plataforma de SMA está garantido pelo componente de gerência, os agentes modelados na arquitetura correspondente já podem ser instanciados e começar a atuar no ambiente. A ferramenta de orquestração permite o sensoriamento do seu ambiente para seus nodos de gerência, fato este que garante aos agentes a capacidade de observar os recursos do ambiente.

Embora seja garantido aos agentes um ambiente para atuarem e uma arquitetura para estarem contidos, estes ainda precisam ser instanciados. Esse processo, na tecnologia JADE usada neste trabalho, deve ser feito em tempo de execução ou através de parâmetros na inicialização de um dado *container*. No âmbito desta proposta ambos sistemas de inicialização dos agentes são empregados para garantir que tanto o componente

de gerência de agentes quanto o restante da ferramenta de orquestração não precisem ativamente preocupar-se com a inicialização dos agentes.

A modelagem dos agentes elaborados neste trabalho foi realizada utilizando a *Prometheus Modeling Tool* que utiliza a notação apresentada na Figura 5:

Figura 5. Legenda relativa aos diagramas dos modelos de agentes.

- Ação: Mecanismo que permite que o agente interaja com o ambiente em que se econtra. Pode representar funções atuadoras ou interações com artefatos;
- Capacidade: Abstração de uma funcionalidade do sistema. É composta por planos, eventos do ambiente e dados;
- Dados: Representação de um item ou estrutura de dados utilizado pelo sistema.;
- Mensagem: Mensagem trocada entre agentes;
- Plano: Sequência de eventos e ações que resultam na execução de um objetivo;
- Percepção: Informação oriunda de algum ponto do ambiente que pode ser interpretada pelo agente.

2.2.1. Agente Inspetor da Plataforma

O agente inspetor da plataforma deve existir enquanto houver uma instância da arquitetura de SMA ativa. Dada essa característica, este inspetor é inicializado junto aos *main containers* da plataforma através de parâmetros na inicialização dessa. Deste modo, o agente inspetor possui seu próprio ciclo de vida garantido pelo fato de que os nodos principais da plataforma contém redundância, mantendo-se ativos com cópias para atuar como nodo principal na falha do *main container* corrente. O inspetor da plataforma tem como responsabilidade tanto instânciar os agentes para efetuarem a gerência do ambiente de computação em nuvem, quanto realizar o processo de gerenciamento destes. Caso seja notado que um determinado agente não seja mais necessário, é responsabilidade do inspetor terminar sua execução.

Sendo esse o único agente de meta gerência, seus objetivos diferem do restante da sociedade em relação ao domínio de sua atuação (único agente que sensoria e tem suas ações direcionadas a própria plataforma de SMA). Para simplificar a visualização do restante da sociedade de agentes, este agente é modelado de forma separada (como um sistema disjunto do restante da sociedade de agentes), de modo que todas as atribuições, percepções e funcionalidades desse sub-sistema competem a ele apenas. Seguindo a metodologia *Prometheus*, baseada em teoria de agentes, podemos decompor este sistema em funcionalidades, percepções, objetivos, ações e protocolos de comunicação [Padgham and Winikoff 2003].

Os eventos trazem informações diretas e completas para o agente, o qual consegue deliberar sobre as implicações diretas de seus acontecimentos. As percepções, no entanto, precisam ser analisadas e ponderadas [Padgham and Winikoff 2003]. Sendo assim os eventos e percepções relacionados com esse agente são os que seguem:

1. Evento da destruição de *container*: ocorre quando um nodo da plataforma é destruido, terminando ações de agentes que lá residiam;
2. Evento de criação de *container*: ocorre quando um novo nodo da plataforma é criado, podendo conter novos agentes;
3. Percepção de mudança nas necessidades do sistema: ocorre ao ser percebido que a ferramenta de orquestração mudou suas heurísticas;
4. Percepção de ócio: percebido quando a tarefa executada por um grupo de agentes está usando mais recursos do que precisa para sua completude;
5. Percepção de subdesempenho: percebido quando uma tarefa executada por um grupo de agentes não está tendo sucesso de se concluir em uma taxa desejada.

As ações do inspetor são listadas a seguir:

1. Destruir um agente: terminar a execução de um agente de algum dado tipo;
2. Criar um agente: inicializar um agente dentro da plataforma de SMA;
3. Interpretar necessidades da ferramenta de orquestração: deriva um modelo de requisitos para ser implementado pelo inspetor sobre a vontade da ferramenta de orquestração;
4. Verificar espaço para novos agentes: confere se há local para criar novos agentes no ambiente ou se a plataforma está lotada;
5. Analisar tempo de execução de tarefa: mede o tempo que uma atividade está demorando para ser realizada;

A Figura 6 demonstra o documento chamado de *system overview* da metodologia *prometheus* para este sistema.

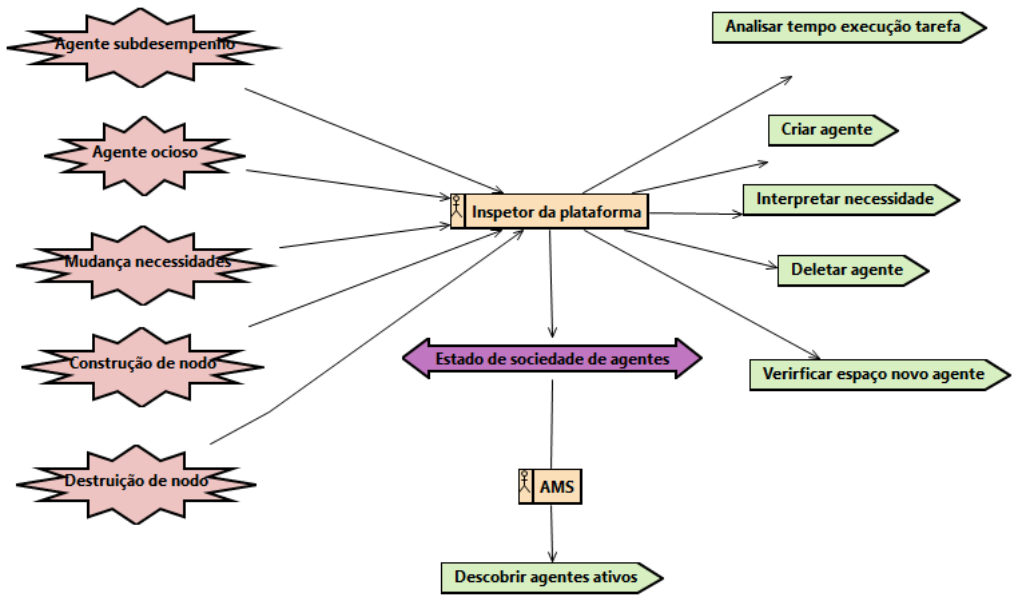

Figura 6. *System overview diagram* referente ao agente inspetor da plataforma.

2.3. Modelo dos Agentes de Gerência do Ambiente de Nuvem

Nesta seção serão descritos os modelos dos três agentes responsáveis por implementar as funções propostas em [Bräscher 2015].

2.3.1. Agente de Busca do Cluster

O agente de busca de *cluster* é um agente reativo, segundo a definição proposta em [Wooldridge et al. 1995], que classifica uma arquitetura de agente como reativa quando este não possui um modelo simbólico do mundo a sua volta e não faz uso de nenhum tipo de raciocínio simbólico. O agente tem o objetivo de monitorar o ambiente a procura de *clusters* que necessitam ser trabalhados, para então apresentá-los de forma que o agente diligente possa trabalhar sobre os *clusters*. Essa listagem é feita através de um *blackboard*, instanciado na plataforma JADE. Como ilustrado na Figura 7, o agente de busca terá conhecimento apenas dos *clusters* ativos do ambiente e, uma vez que um novo *cluster* for descoberto ou algum *cluster* existente tiver sido trabalhado pela última vez a muito tempo, ele deve escrever no *blackboard* a necessidade de trabalhar novamente sobre aquele *cluster*.

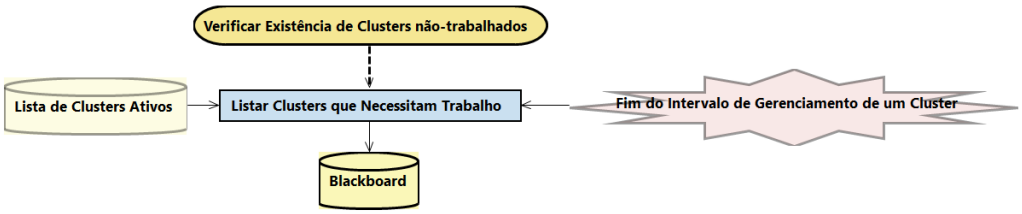

Figura 7. Modelo do agente de busca de cluster.

2.3.2. Agente Diligente do Cluster

O agente diligente do *cluster* é responsável por realizar os processos de gerência nos *clusters* que foram identificados como aptos para receber tais processos. Ao encontrar um *cluster* listado no *blackboard*, o agente deve aplicar as heurísticas definidas pelo administrador do ambiente para determinar a prioridade para a realização do trabalho sobre os seus *hosts*. Uma vez definida a prioridade dos *hosts* a serem trabalhados, o agente diligente solicita, em ordem, que as *VMs* dos *hosts* de menor prioridade sejam mapeados aos *hosts* de maior prioridade, na medida em que estes possuírem recursos suficientes disponíveis. Uma vez que um *host* não possui mais *VMs* mapeadas a ele, este *host* pode ser desativado. Como explicitado na Figura 8, o agente espera o término do trabalho para poder marcar o *cluster* como trabalhado por um tempo determinado pelo administrador e atuar sobre o próximo *cluster*.

2.3.3. Agente de Ativação de Recursos

O agente de ativação de recursos tem como objetivo manter a disponibilidade de recursos suficientes para o bom funcionamento do ambiente, sendo o responsável por verificar periodicamente a necessidade de disponibilizar mais recursos físicos para o ambiente. Ele é o responsável pela ativação de *hosts* que podem vir a serem desativados. Ao ser instanciado, é estabelecido um intervalo de tempo entre verificações, ao final do qual o agente compara a quantidade de recursos sendo utilizados com a quantidade de recursos

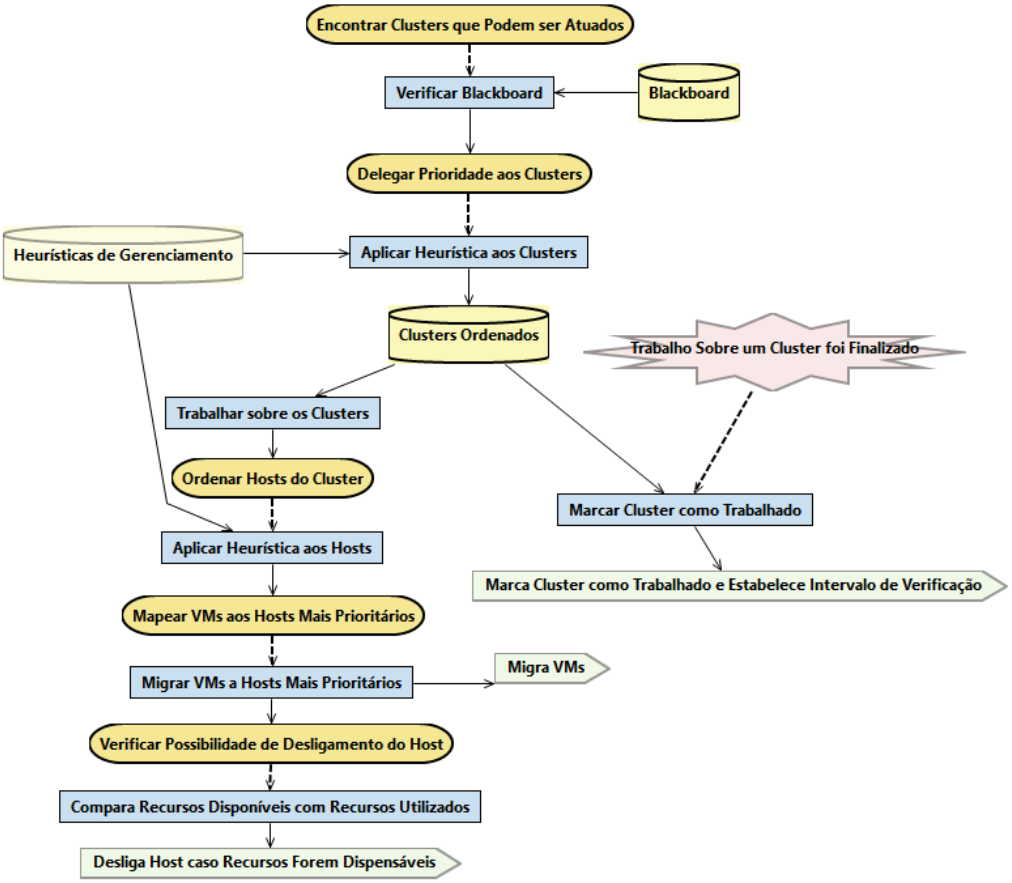

Figura 8. Modelo do agente diligente do cluster.

ativos e, baseado nas heurísticas, decide se devem ser ativados novos recursos e, sendo necessário, a quantidade a ser ativada. A Figura 9 explicita o modelo do agente.

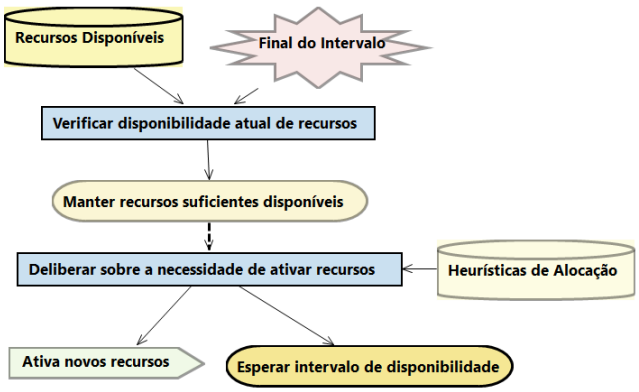

Figura 9. Modelo do agente de alocação.

Vale notar que o agente de ativação, por conta própria, não consegue garantir integralmente a disponibilidade de recursos físicos, já que podem haver casos de uma demanda inesperada por recursos enquanto o intervalo de verificação ainda estiver correndo. Para remediar estas situações, um gerente de recursos é injetado na plataforma

de orquestração do ambiente extendida por [Bräscher 2015], e atua sob demanda para disponibilizar recursos físicos, quando for tentada a alocação de uma *VM* e não forem encontrados recursos suficientes.

3. Desenvolvimento

Este trabalho foi executado sobre a versão 4.6.0 do *Apache Cloudstack*. Todo o processo de desenvolvimento ocorre dentro do escopo criado pelo *framework* de [Bräscher 2015] na versão 1.0.2. O arcabouço criado pelo *framework* é composto por um grupo de *plugins* que formam um módulo adicional no *Apache Cloudstack*, chamado de *autonomiccs platform*.

Para manter a hierarquia criada pela adição do *framework* de [Bräscher 2015] e a sua comunicação com a ferramenta de orquestração, este trabalho teve como objetivos tanto a criação de um módulo para instanciar e gerenciar a plataforma de sistema multiagentes, quanto modificar os *plugins* do *framework* de forma a dissociar os serviços criados que são inerentes ao *Apache Cloudstack* da lógica de tomada de decisões, afim de transforma-las nos agentes definidos e apresentados na seção 2.2.

Para a criação do gerente de SMA, o padrão seguido pelos agentes do *framework* de [Bräscher 2015] foi utilizado, injetando objetos no *Apache Cloudstack* através da ferramenta *Spring*. Assim garantindo a instância e execução do ciclo de percepção da plataforma. Com este gerente ativo, a criação das plataformas SMA e dos agentes nela inseridos pode ocorrer através de funções já existentes no *framework* para criar VMs e acessa-las através de SSH, injetando as dependências do JADE e inicializando-o.

Os agentes residentes na plataforma JADE que compõem a sociedade definida neste artigo foram desenvolvidos utilizando o *framework* de desenvolvimento de agentes JADE e suas funções de atuação envolvem o consumo de APIs disponibilizadas pela ferramenta contida no *framework* de [Bräscher 2015].

4. Conclusão

Embora não tenham sido realizados testes nas mudanças feitas no *framework* proposto por [Bräscher 2015], pode-se observar os efeitos que as modificações surtiram na organização e relação dos módulos do *autonomiccs plataform*. As vantagens dessa reorganização abrangem a modularização do código, a compatibilidade com o padrão FIPA para comunicação entre agentes, fazendo com que outras plataforma de SMA possam comunicar-se com o módulo de gerência da ferramenta de orquestração.

A comunicação criada entre JADE e *Apache Cloudstack* permite que desenvolvedores e pesquisadores possam ter um arcabouço para criar agentes capazes de acessar os recursos administrativos de uma ferramenta de orquestração. Além de fazer com que plataformas externas de SMA possam interoperar com os agentes do *Apache Cloudstack*. Acredita-se ainda que pela capacidade de adicionar múltiplos agentes para aplicar as heurísticas do ambiente na nuvem, sem adições extras no código, tornam essa abordagem mais escalável e eficiente. Entretanto, tendo em vista a ausência de testes para essa estensão, não há condições científicas de afirmar a eficiência e eficácia desta abordagem com sistemas multiagentes.

5. Trabalhos Futuros

Sendo assim, como trabalhos futuros, têm-se:

- Avaliar a eficiência e eficácia da abordagem de sistemas multiagentes contra a aplicação de serviços acoplados na ferramenta de orquestração proposta em [Bräscher 2015];
- Verificar a capacidade de escalabilidade dessa abordagem em comparação com outras atuais, como a proposta em [Bräscher 2015];
- Testar a adaptabilidade do SMA a heurísticas que não sejam a consolidação do ambiente em nuvem;
- Escrever um artigo científico demonstranto a solução proposta e seus resultados para ser publicado em uma conferência a ser definida.

Referências

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

Awada, U., Li, K., and Shen, Y. (2014). Energy consumption in cloud computing data centers. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, 3(3):145–162.

Bräscher, G. B. (2015). *Proposta de Um Framework Para Consolidação de Recursos em Ambientes de Computação em Nuvem*. UFSC.

De la Prieta, F., Rodriguez, S., Bajo, J., and Corchado, J. (2013). A multiagent system for resource distribution into a cloud computing environment. In Demazeau, Y., Ishida, T., Corchado, J., and Bajo, J., editors, *Advances on Practical Applications of Agents and Multi-Agent Systems*, volume 7879 of *Lecture Notes in Computer Science*, pages 37–48. Springer Berlin Heidelberg.

Hou, F., Mao, X., Wu, W., Liu, L., and Panneerselvam, J. (2014). A Cloud-Oriented Services Self-Management Approach Based on Multi-agent System Technique. *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 261–268.

Ibrahim, S., He, B., and Jin, H. (2011). Towards pay-as-you-consume cloud computing. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 370–377. IEEE.

Padgham, L. and Winikoff, M. (2003). Prometheus: A methodology for developing intelligent agents. In *Agent-oriented software engineering III*, pages 174–185. Springer.

Silveira, R. A. (2006). Introdução a sistemas multi-agente. *Universidade Federal de Santa Catarina (UFSC)*.

Weingärtner, R., Bräscher, G. B., and Westphall, C. B. (2015). Cloud resource management: A survey on forecasting and profiling models. *Journal of Network and Computer Applications*, 47:99–106.

Whitney, J. and Delforge, P. (2014). Data center efficiency assessment. *Issue paper on NRDC (The Natural Resource Defense Council)*.

Wooldridge, M., Jennings, N. R., et al. (1995). Intelligent agents: Theory and practice. *Knowledge engineering review*, 10(2):115–152.