Análise Comparativa de Desempenho e Arquitetura dos Frameworks Multiagentes MASPY e SPADE

Guilherme S. Cerdeira[1], Alexandre L. L. Mellado[1], Rafael C. Cardoso[2], André P. Borges[1], Gleifer V. Alves[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Ponta Grossa, PR, Brasil

[2]University of Aberdeen
Aberdeen, United Kingdom

{gcerdeira,mellado}@alunos.utfpr.edu.br

rafael.cardoso@abdn.ac.uk, {apborges,gleifer}@utfpr.edu.br

Abstract. *The choice of a framework for Multi-Agent System (MAS) development is a critical decision, marked by a significant trade-off between distributed platforms like SPADE and centralized ones like MASPY. This paper addresses the lack of quantitative benchmarks between these two architectures. To this end, a comparative performance analysis was conducted using the Contract Net Protocol as a case study, measuring execution time in scenarios with up to 200 agents. The results demonstrated MASPY's significant performance superiority, proving to be up to 2.7 times faster than SPADE in a single-machine environment. Additionally, MASPY's BDI approach resulted in a more concise implementation. It is concluded that the choice of framework is not a matter of absolute superiority, but of suitability for the project's architecture: MASPY is ideal for prototyping and high-performance centralized systems, whereas SPADE is indispensable for distributed, interoperable, and scalable applications.*

Resumo. *A escolha de um framework para o desenvolvimento de Sistemas Multiagentes (SMAs) é uma decisão crucial, marcada por um trade-off significativo entre plataformas distribuídas como SPADE e centralizadas como MASPY. Este artigo aborda a lacuna de benchmarks quantitativos entre essas duas arquiteturas. Para isso, foi realizada uma análise comparativa de desempenho utilizando o Protocolo Contract Net como estudo de caso, medindo o tempo de execução em cenários com até 200 agentes. Os resultados demonstraram uma superioridade de desempenho expressiva do MASPY, que se mostrou até 2.7 vezes mais rápido que o SPADE em ambiente de máquina única. Adicionalmente, a abordagem BDI do MASPY resultou em uma implementação mais concisa. Conclui-se que a escolha do framework não é uma questão de superioridade, mas de adequação à arquitetura do projeto.*

1. Introdução

O desenvolvimento de Sistemas Multiagentes (SMAs) é facilitado por frameworks que abstraem a complexidade da comunicação, coordenação e raciocínio dos agentes [Wooldridge 2009]. Essas plataformas fornecem estrutura para a modelagem e execução destes agentes, por meio de diversos paradigmas.

Neste contexto, destacam-se duas plataformas Python com filosofias distintas: SPADE (Smart Python Agent Development Environment) [Jimenez-Fernandez et al. 2020], um framework consolidado que adota os padrões FIPA e utiliza o protocolo XMPP para garantir a interoperabilidade em ambientes distribuídos; do outro, o MASPY [Mellado et al. 2023], um framework moderno inspirado na arquitetura BDI (Belief-Desire-Intention) [Rao and Georgeff 1991], otimizado para a programação declarativa do raciocínio simbólico em ambientes centralizados de alto desempenho.

A escolha entre a escalabilidade de SPADE e o desempenho local de MASPY carece de benchmarks quantitativos na literatura. Este artigo aborda diretamente essa lacuna ao apresentar uma análise comparativa entre os dois frameworks. Utilizando o Protocolo Contract Net [Smith 1980] como um estudo de caso para a avaliação comparativa. Este protocolo foi especificamente escolhido por ser um paradigma canônico de interação em SMAs, exigindo um conjunto de capacidades fundamentais de qualquer plataforma de agentes: comunicação um-para-muitos (o leilão), resposta um-para-um (as propostas), agregação de estado (o recebimento das propostas) e tomada de decisão baseada em regras (a escolha do vencedor). Assim, o Contract Net serve como um benchmark eficaz para estressar tanto a infraestrutura de comunicação do SPADE quanto o motor de raciocínio do MASPY.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha as ferramentas e a metodologia empregada, incluindo a descrição dos frameworks e do protocolo de teste. A Seção 4 exibe os resultados quantitativos da análise de desempenho, enquanto a Seção 5 discute as implicações desses resultados sob uma perspectiva arquitetônica e de usabilidade. Por fim, a Seção 6 apresenta as conclusões do trabalho e aponta direções para pesquisas futuras.

2. Trabalhos Relacionados

A avaliação e comparação de frameworks para o desenvolvimento de Sistemas Multiagentes é um tópico recorrente na literatura, visando guiar desenvolvedores na escolha da ferramenta mais adequada para seus projetos. Diversos trabalhos se dedicaram a comparar plataformas consolidadas, frequentemente focando em benchmarks de desempenho sob protocolos de interação específicos. Por exemplo, Radhakrishnan et al. [Radhakrishnan et al. 2018] realizaram uma análise comparativa entre JADE e SPADE.

Dentro do ecossistema de agentes BDI, o framework Jason [Bordini et al. 2007], baseado na linguagem AgentSpeak, é frequentemente citado como uma das principais implementações do modelo, sendo amplamente utilizado em pesquisas acadêmicas. Da mesma forma, no universo de plataformas compatíveis com os padrões FIPA, o JADE (Java Agent DEvelopment Framework) [Bellifemine et al. 2007] representa uma das mais influentes e utilizadas, servindo como base para inúmeros trabalhos e comparações.

Apesar da existência desses estudos, observa-se uma lacuna na literatura no que tange a comparação direta entre um framework BDI moderno e leve em Python, como o MASPY, e uma plataforma FIPA consolidada na mesma linguagem, como o SPADE. Os trabalhos existentes tendem a comparar frameworks de famílias semelhantes (BDI vs. BDI) ou plataformas em linguagens diferentes (Java vs. Python). O presente artigo

contribui para preencher essa lacuna, oferecendo um benchmark de desempenho e uma análise arquitetônica que contrasta diretamente as duas filosofias de projeto no ecossistema Python.

3. Ferramentas e Metodologia

Como breve introdução de ambas plataformas, SPADE (Smart Python Agent Development Environment) [Jimenez-Fernandez et al. 2020] é um framework projetado sobre os padrões da FIPA para o desenvolvimento de sistemas multiagentes distribuidos. Sua arquitetura é fundamentada no protocolo de comunicação XMPP, o que significa que cada agente opera apartir de um identificador único (JID). O desenvolvimento em SPADE é centrado no conceito de comportamentos, como "CyclicBehaviour"ou "OneshotBehaviour", que rodam de forma concorrente e assíncrona. Essa estrutura realça a superioridade em termos de escalabilidade de aplicações executadas em diferentes máquinas.

Já MASPY [Mellado et al. 2023] é um framework composto por quatro classes para abstrações do paradigma BDI, agentes, ambiente, comunicação e administrador. A classe de agentes contém o necessário para gerenciar o conhecimento e raciocínio deste; a classe de ambiente fornece um contexto para agentes perceberem e interagirem sobre um mesmo ambiente; a classe de comunicação permite que estes agentes troquem informação entre si e o administrador organiza e executa o sistema.

Em suma, as diferenças significativas residem, por um lado, na escalabilidade horizontal do SPADE [Jimenez-Fernandez et al. 2020], obtida através de sua arquitetura de rede distribuída (XMPP) e modelo de programação procedural assíncrono. Em contraste, MASPY [Mellado et al. 2023] destaca-se pela eficiência de execução local e pela expressividade do raciocínio cognitivo, frutos de sua arquitetura centralizada e paradigma declarativo.

Todos os experimentos foram conduzidos em um ambiente de máquina única para garantir a consistência e a reprodutibilidade dos testes. Embora o uso do Subsistema do Windows para Linux (WSL) introduza uma camada de virtualização, ambos os frameworks foram submetidos exatamente às mesmas condições de execução. O foco do estudo reside na comparação de desempenho relativo entre as plataformas, e não na medição de um desempenho absoluto. Dessa forma, qualquer custo computacional constante introduzido pelo ambiente de execução afeta ambas as ferramentas de maneira uniforme, preservando a validade da análise comparativa sobre qual arquitetura é mais eficiente neste cenário específico.

As ferramentas utilizadas foram selecionadas para criar um ambiente de teste controlado. A plataforma de hardware consistiu em um computador com processador Intel Core i5-10300H, 8 GB de RAM, operando com o sistema operacional Debian GNU/Linux 12 (bookworm) utilizando WSL (Windows Subsystem for Linux).

O ambiente de software foi construído sobre a linguagem de programação Python versão 3.13.5. As versões dos frameworks de agentes analisados foram SPADE versão 4.1.0 e MASPY versão 0.6.4.

Para realizar a comparação de desempenho, foi implementado uma versão simplificada do FIPA Contract Net Protocol [Smith 1980], um mecanismo de interação padrão para alocação de tarefas em sistemas multiagentes. A escolha deste protocolo se justi-

fica por sua capacidade de testar de forma integrada a comunicação, a coordenação e o gerenciamento de estado dos agentes. As execuções deste sistema em ambos, SPADE e MASPY, foram feitas na mesma máquina localmente. O fluxo de interação do protocolo segue as seguintes etapas:

1. Chamada por Propostas (CFP): Um agente *Initiator* envia uma mensagem de CFP a um grupo de agentes *Participant*, anunciando uma tarefa a ser executada.
2. Proposta: Cada agente *Participant* avalia o CFP e responde com uma mensagem de *propose*, contendo o custo para realizar a tarefa (neste estudo, um valor inteiro aleatório entre 20 e 20000), ou com uma mensagem de *refuse*, caso não possa ou não queira participar.
3. Seleção e Contratação: O agente *Initiator* coleta as propostas recebidas. Após receber todas as respostas, ele seleciona a proposta de menor custo como vencedora.
4. Notificação: O agente *Initiator* envia uma mensagem de *accept-proposal* ao agente vencedor e mensagens de *reject-proposal* a todos os outros que enviaram propostas.

4. Resultados

Nesta seção são apresentados os resultados da análise comparativa de desempenho entre os frameworks MASPY e SPADE. Os dados foram coletados executando o cenário do Protocolo Contract Net com uma carga crescente de agentes, de 20 a 200 participantes, conforme detalhado na metodologia.

Tabela 1. Tempo de execução médio (em segundos) por número de agentes.

Nº de Agentes	Tempo Médio (MASPY) [s]	Tempo Médio (SPADE) [s]
20	0.5404	6.4030
40	1.4513	10.4639
60	2.2589	13.3896
80	4.6146	25.2059
100	7.5856	30.2689
120	9.2213	30.4956
140	12.9443	33.0728
160	15.5344	40.1788
180	19.0409	50.8997
200	22.3292	59.3107

A Tabela 1 resume os dados de tempo de execução médio coletados para cada plataforma. Para garantir a robustez estatística, cada cenário foi executado três vezes, e a média dos resultados foi registrada. Em seguida, a Figura 1 ilustra visualmente esses dados, facilitando a comparação da escalabilidade de cada framework.

Para validar quantitativamente a linearidade da relação entre o aumento do número de agentes e o tempo de execução, foi calculado o coeficiente de correlação de Pearson (r). Para o MASPY, obteve-se um valor de $r = 0.9867$, e para o SPADE, um valor de $r = 0.9812$. Ambos os coeficientes, por serem muito próximos de +1, indicam uma correlação linear positiva muito forte. Este resultado confirma estatisticamente a observação visual

do gráfico, validando que o tempo de execução aumenta de forma previsível e linear com a adição de mais agentes para ambas as plataformas, sob as condições deste experimento.

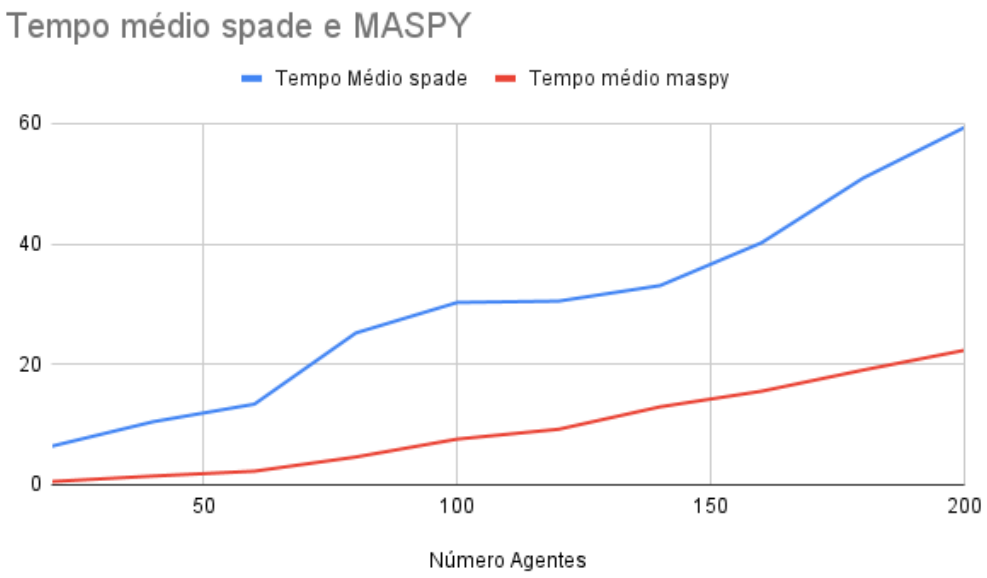

Figura 1. Comparativo de desempenho: Tempo de execução em segundos (eixo y) vs. Número de agentes (eixo x).

Como ilustrado na Figura 1, observa-se que o MASPY (representado pela linha vermelha) demonstrou um desempenho significativamente superior em todos os cenários. Seu tempo de execução apresenta um crescimento linear e contido, partindo de 0.5404 segundos para 20 agentes e atingindo 22.3292 segundos para 200 agentes. Em contrapartida, a implementação com SPADE (linha azul) exibe uma sobrecarga inicial substancialmente maior, registrando 6.4030 segundos já no cenário de 20 agentes.

A disparidade de desempenho se acentua com o aumento da carga. No cenário com 200 agentes, por exemplo, o SPADE necessitou de 59.3107 segundos, sendo aproximadamente 2.7 vezes mais lento que o MASPY. Este resultado condiz com a hipótese inicial de que a arquitetura de comunicação em memória do MASPY é mais eficiente para sistemas executados em máquina única, enquanto o modelo de comunicação em rede do SPADE introduz uma latência considerável que impacta diretamente a performance.

5. Análise dos Resultados

A superioridade de desempenho do MASPY, evidenciada nos resultados, não é um fator isolado, mas uma consequência direta de sua arquitetura de comunicação fundamentalmente diferente da utilizada pelo SPADE. A eficiência do MASPY deriva de sua comunicação em memória: todos os agentes executam em um único processo, e a troca de informações entre eles se assemelha a uma chamada de função interna, caracterizada por uma sobrecarga mínimo e pela ausência de latência de rede. Em contraste, o SPADE foi projetado para interoperabilidade e distribuição através da comunicação em rede via XMPP. Cada mensagem enviada por um agente SPADE passa por um ciclo custoso que envolve serialização para o formato XML, transmissão para um servidor intermediário,

processamento e o caminho reverso até o destinatário. Este processo inerentemente mais lento justifica a performance inferior em um ambiente de máquina única.

Essa diferença é refletida diretamente no paradigma de programação e na experiência do desenvolvedor. Ao abstrair a complexidade da comunicação, o MASPY permite que o programador adote um modelo declarativo focado na arquitetura cognitiva BDI, onde a lógica é expressa em termos de objetivos e planos (“Goals” e “Plans”). Isso resulta em um código mais conciso e conceitualmente mais próximo do raciocínio do agente. Por outro lado, o SPADE expõe a natureza da comunicação em rede, exigindo que o desenvolvedor adote um paradigma procedural e assíncrono. É necessário gerenciar explicitamente os comportamentos (“Behaviours”), lidar com primitivas de ‘async/await‘ e com a estrutura das mensagens, o que, embora conceda maior controle sobre a comunicação, aumenta a verbosidade e a complexidade do código.

Para quantificar essa afirmação, a lógica central do protocolo neste estudo foi implementada com aproximadamente **25 linhas** de código em MASPY, em contraste com as **70 linhas** necessárias em SPADE. Essa diferença se torna visualmente aparente ao comparar os trechos de código responsáveis pela mesma tarefa: o envio da chamada por propostas (CFP), como mostram os Códigos 1 e 2.

```
@pl(gain, Goal("start_cfp"))
def start_cfp(self, src):
    self.send(broadcast, achieve, Goal("send_proposal"))
    self.add(Goal("evaluate_current_proposals"))
```

Listing 1. Trecho de código do MASPY para envio do CFP

```
class SendCFPBehaviour(OneShotBehaviour):
    async def run(self):
        for jid in self.agent.Participant_jids:
            msg = Message(to=jid)
            msg.set_metadata("performative", "cfp")
            msg.set_metadata("protocol", "contract_net")
            await self.send(msg)
        self.agent.add_behaviour(
            self.agent.ReceiveProposalsBehaviour(),
            self.agent.proposal_template
        )
```

Listing 2. Trecho de código do SPADE para a mesma tarefa.

No Código 1, a intenção do agente é declarada de forma direta através do envio de um novo objetivo (“Goal”). Em contraste, o Código 2 demonstra a necessidade de gerenciar explicitamente os comportamentos (“Behaviours”), lidar com primitivas de ‘async/await‘ e construir manualmente a estrutura de cada mensagem, o que, embora conceda maior controle sobre a comunicação, valida a afirmação sobre a maior complexidade da implementação.

Finalmente, os resultados deste estudo, conduzido em um ambiente de máquina única, permitem inferir sobre os modelos de escalabilidade inerentes a cada framework. O desempenho do MASPY está diretamente atrelado aos recursos do hardware local, um

comportamento característico da escalabilidade vertical. Em contrapartida, a sobrecarga de desempenho do SPADE, mesmo em um cenário local, evidencia uma arquitetura construída para um propósito diferente: a escalabilidade horizontal. Embora os experimentos atuais não tenham medido o desempenho em um ambiente distribuído, a arquitetura baseada em rede do SPADE é intrinsecamente projetada para permitir que o sistema cresça pela adição de múltiplas máquinas, superando as limitações de um único hardware.

6. Conclusão

Este artigo realizou uma análise comparativa de desempenho, usabilidade e arquitetura entre os frameworks para sistemas multiagentes MASPY e SPADE, utilizando o Protocolo Contract Net como estudo de caso. Os resultados quantitativos demonstraram inequivocamente a superioridade de desempenho do MASPY em ambiente de máquina única, sendo até 2.7 vezes mais rápido que o SPADE em cenários com 200 agentes. A discussão aprofundada revelou que essa eficiência é uma consequência direta da arquitetura de comunicação em memória do MASPY, que contrasta com o significativa sobrecarga de rede imposto pelo modelo baseado em XMPP do SPADE. Adicionalmente, a análise qualitativa indicou que a abordagem declarativa BDI do MASPY resulta em um código mais conciso e com menor carga cognitiva para o desenvolvedor em comparação com o modelo procedural e assíncrono do SPADE.

O MASPY firma-se como a ferramenta de escolha para prototipagem, simulações e sistemas centralizados, onde sua arquitetura em memória garante máxima eficiência. O SPADE, por sua vez, é a plataforma indispensável para aplicações de produção, distribuídas e que demandam interoperabilidade, cuja arquitetura é projetada para a robustez e a escalabilidade horizontal, o que justifica a sobrecarga de comunicação observada.

Este estudo estabelece um claro ponto de partida de desempenho em ambiente local. A continuação natural desta pesquisa envolve a análise de métricas como consumo de CPU e memória que poderiam oferecer um panorama comparativo mais completo para a comunidade de desenvolvedores de sistemas multiagentes.

Como trabalho futuro também será possível a expansão desta análise com a inclusão da extensão SPADE-BDI [e Silva et al. 2018]. A utilização de uma lógica BDI em ambos os frameworks permitiria uma comparação mais direta, focada especificamente no impacto de desempenho da arquitetura de comunicação de cada plataforma.

Agradecimentos: Este trabalho tem financiamento do projeto: 444568/2024-7, CNPq/MCTI/FNDCT Nº 22/2024, *Programa Conhecimento Brasil – Apoio a Projetos em Rede com Pesquisadores Brasileiros no Exterior.*

Referências

Bellifemine, F., Caire, G., and Greenwood, D. (2007). Developing multi-agent systems with jade. *Java Programming, Wireless and Mobile Applications*, pages 231–250.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley.

e Silva, A. G. S. T., Teixeira, C. A. C., dos Santos, I. M., and Julian, V. (2018). Integração da Arquitetura BDI ao Framework SPADE para o Desenvolvimento de Agentes Cognitivos. In *Anais do XV Encontro Nacional de Inteligência Artificial e Computaci-*

onal (ENIAC 2018), pages 73–84, Porto Alegre, RS, Brasil. Sociedade Brasileira de Computação.

Jimenez-Fernandez, J. A., de la Prieta, F., Carrascosa, C., Rincon, J. A., and Julian, V. (2020). SPADE 3: An agent platform for the IoT. In *Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2020)*, pages 156–168. Springer.

Mellado, A. L. L., Borges, A. P., Alves, G. V., and Cardoso, R. C. (2023). MASPY: Um framework em python para o desenvolvimento de sistemas multiagentes BDI. In *Anais do IX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações (WESAAC 2023)*, pages 13–24.

Radhakrishnan, G., V., C., and K.L., S. (2018). Comparative study of JADE and SPADE multi agent system. *International Journal of Advanced Research*, 6(11):1035–1042.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR '91)*, pages 473–484.

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. John Wiley & Sons, 2nd edition.