Evento: XXIII Jornada de Pesquisa

DESAFIOS NOS MOTORES DAS PLATAFORMAS DE INTEGRAÇÃO DE APLICAÇÕES EMPRESARIAIS[1]
CHALLENGES IN THE RUNTIME SYSTEM OF ENTERPRISE APPLICATION INTEGRATION PLATFORMS

Fernando Parahyba[2], Fabricia Roos-Frantz[3]

[1] Pesquisa desenvolvida no Mestrado em Modelagem Matemática, pertencendo ao Grupo de Pesquisa em Computação Aplicada (GCA)
[2] Bolsista CNPq, Aluno do Curso de Mestrado em Modelagem Matemática da Unijuí
[3] Professora Doutora do Departamento de Ciências Exatas e Engenharias

Resumo

As empresas buscam alternativas tecnológicas que proporcionem competitividade para seus processos de negócios. Dentre essas, existem as plataformas de integração que integram um conjunto de aplicações, o ecossistema de software. As plataformas de integração dão suporte para implementar soluções de integração, que fazem com que as aplicações funcionem de forma sincronizada, permitindo o reuso de dados e funcionalidades do ecossistema de forma rápida e segura. As plataformas de integração geralmente fornecem uma linguagem de domínio específico, um kit de ferramentas de desenvolvimento e um motor de execução. Há uma constante preocupação com a melhoria das plataformas de integração, a fim de que elas proporcionem a combinação de menor tempo de execução das unidades de trabalho com a economia do consumo de recursos computacionais. A maneira como o motor de execução pode executar as unidades de trabalho tem um impacto direto no desempenho de uma solução e este é um dos desafios enfrentados pelas plataformas de integração. Identificou-se na literatura que os motores de integração das plataformas open-source adotam duas possibilidades, pool Global de Threads, sendo a mais usada, e pools locais. O problema dos motores se acentuam no caso de uma grande carga inicial de mensagens de entrada na solução, gerando mais fila e maior tempo de processamento, necessitando de mais Threads. Determinar o número de Threads, em cada pool, que minimiza o tempo médio de processamento de uma mensagem em uma solução de integração, é uma tarefa difícil de ser realizada manualmente. Esse artigo apresenta uma revisão teórica sobre o que se tem feito para tratar desse problema. Um algoritmo baseado na técnica Particle Swarm Optimization (PSO), que atribui as unidades de trabalho para as Threads em pools locais, a fim de otimizar o desempenho da solução de integração, um software que transforma os modelos conceituais e por meio de simulação consegue precisar o número de Threads, num pool Global, com uma melhor segurança e faz uma análise de possíveis trabalhos futuros.
Palavras-chave: Integração de aplicações empresarias; Motor de execução; Optimização.

Abstract

Companies seek technological alternatives that provide competitiveness for their business

processes. Among these, there are integration platforms that integrate a set of applications, the software ecosystem. Integration platforms support integration deployments that make applications work in a synchronized way, allowing reuse of data and ecosystem functionality quickly and securely. Integration platforms typically provide a specific domain language, development toolkit, and runtime system. There is a constant concern for the improvement of the integration platforms, so that they provide the combination of shorter execution times of the work units with the economy of the consumption of computational resources. The way the execution runtime system can run the work units has a direct impact on the performance of a solution and this is one of the challenges faced by the integration platforms. It has been identified in the literature that the engines of integration of the open-source platforms adopt two possibilities, Global pool of Threads, being the most used, and local pools. The problem of the engines is accentuated in the case of a large initial load of incoming messages in the solution, generating more queue and longer processing time, requiring more Threads. Determining the number of threads in each pool that minimizes the average processing time of a message in an integration solution is a difficult task to do manually. This paper presents a theoretical review of what has been done to address this problem. An algorithm based on the Particle Swarm Optimization (PSO), which assigns work units to the threads in local pools, in order to optimize the performance of the integration solution, software that transforms conceptual models and through simulation may need the number of Threads, in a Global pool, with better security and makes an analysis of possible future work.
Keywords: Enterprise Application Integration; Runtime System; Optimization.

1. Introdução

A utilização de aplicações é fundamental para dar suporte aos processos de negócios das empresas. Normalmente, elas possuem em seu ecossistema de software (MESSERSCHMITT; SZYPERSKI et al., 2005) diversas aplicações desenvolvidas por ela ou adquiridas no mercado. Desta forma, o ecossistema de software torna-se heterogêneo, com aplicações que possivelmente foram desenvolvidas utilizando diferentes linguagens de programação, de diferentes épocas, e, geralmente, não foram projetadas para trabalhar de forma conjunta (FRANTZ; CORCHUELO; ROOS-FRANTZ, 2016). Em sua maioria, os processos de negócio envolvem dados e funcionalidades presentes em aplicações distintas, o que requer a colaboração destas aplicações.

A área de pesquisa denominada integração de aplicações empresariais (EAI) (HOHPE; WOOLF, 2004) busca oferecer metodologias, técnicas e ferramentas possibilitando com que diferentes aplicações, que não foram desenvolvidas com o propósito de trabalhar em cooperação, possam vir a fazê-lo. De acordo com Linthicum (LINTHICUM, 2000), EAI é uma combinação de tecnologias com a função de dar suporte aos processos de negócio, por meio do desenvolvimento de soluções de integração. Uma solução de integração orquestra, de forma exógena, um conjunto de aplicações, sem a percepção de que estão sendo integradas, e sem introduzir dependência nas aplicações com a solução, permitindo o reuso de dados e funcionalidades presentes no ecossistema.

Evento: XXIII Jornada de Pesquisa

Existem diversas plataformas de integração das quais dá-se destaque a Camel (IBSEN; ANSTEY, 2010), Spring Integration (FISHER et al., 2012), Mule (DOSSOT; D'EMIC; ROMERO, 2014) e Guaraná (FRANTZ; CORCHUELO; ROOS-FRANTZ, 2016). Essas plataformas foram construídas com base nos padrões de integração documentados por Hohpe e Woolf (2004), que se tornaram referência na comunidade EAI. geralmente, essas plataformas de integração fornecem uma linguagem de domínio específico, um kit de ferramentas de desenvolvimento, e um motor de execução, em algumas encontra-se uma ferramenta para monitoramento. A linguagem de domínio específico está voltada para a elaboração de modelos conceituais para a solução de integração, com um nível de abstração próximo ao domínio do problema.

O kit de desenvolvimento é um conjunto de ferramentas de software que permite a implementação da solução, transformando o modelo conceitual à código executável. O ambiente para realização de testes permite provar partes individuais ou toda a solução de integração, com o objetivo de identificar e eliminar possíveis defeitos em sua implementação. O motor de execução proporciona todo o suporte necessário à execução dessas soluções de integração. Por essa razão, o desempenho dos motores de execução está diretamente relacionado com o desempenho das soluções de integração, quanto melhor for o desempenho do motor de execução, melhor será o desempenho da solução de integração.

A arquitetura dos motores de execução das plataformas open-source estudadas está organizada por meio do uso de uma fila do tipo fist-in-first-out (FIFO). Utiliza-se essa estrutura para anotar as tarefas da solução de integração que estão aptas a serem executadas, e a sua execução é realizada por um conjunto de Threads previamente configuradas no motor de execução. Os motores de execução destas plataformas estão implementados com a tecnologia Java (GOSLING, 2000), que proporciona uma Application Programming Interface (API) denominada Executor para criação e alocação de um único pool de Threads ou múltiplos pools, chamados respectivamente de pool Global e pools locais. O dimensionamento do tamanho deste(s) pool(s) é uma atividade realizada pelos engenheiros de software.

O processo de desenvolvimento de uma solução de integração pode ser abstraído nas seguintes etapas: modelagem conceitual, configuração do modelo, implantação e avaliação. Na etapa de modelagem é criado o modelo conceitual da solução de integração e projetado o fluxo de integração. De forma simplificada, configurar o motor de execução significa definir o número de Threads que devem ser utilizadas pelo motor de execução para executar uma solução de integração. Esta configuração é fundamental para garantir um bom desempenho e também atingir os requisitos de qualidade de serviço que o usuário deseja. A fase de avaliação, que pode ser feita por meio da ferramenta de monitoramento, se a plataforma possuir, ou ainda numa etapa posterior já observando o desempenho real da solução e fatores analisados pelo engenheiro de software.

A configuração do número de Threads que serão disponibilizadas para a execução do motor de execução depende de cada solução de integração e é definido a partir do conhecimento empírico

dos engenheiros de software. Esta abordagem traz consigo riscos, pois o não dimensionamento correto do número de Threads tem impacto direto no desempenho dos motores de execução. Um baixo número de Threads pode levar ao acúmulo expressivo de tarefas, enquanto um número muito elevado representa um desperdício de recursoscomputacionais, podendo também causar um acúmulo de tarefas na fila, já que o tempo para o gerenciamento das Threads gastas pelo sistema operacional aumenta consideravelmente (PUSUKURI; GUPTA; BHUYAN, 2011).

As unidades de trabalho da solução de integração são executadas por meio de recursos computacionais presentes no motor de execução, dentre os quais destaca-se as Threads, unidades básicas de utilização da CPU. As Threads são usadas na programação multithreads, visando principalmente possibilitar a simultaneidade das tarefas contidas nessas aplicações, fazendo com que quando algumas delas estiverem bloqueadas, outras possam estar executando tarefas (TANENBAUM, 2009). Nesse tipo de programação, a criação de Threads de maneira ilimitada e sem um controle pode impactar o desempenho da execução da solução de integração.

Aliado ao fato de que um dimensionamento errado de Threads afetam o desempenho do motor e consequentemente da solução de integração, está a escolha entre alocá-las num pool Global ou em pools locais, cabe destacar que em muitas plataformas estudadas, não é possível escolher entre as duas, pois a condição que existe é a de pool Global, podendo apenas determinar o número de Threads. Testar a possibilidade, pouco explorada de pools locais, nas plataformas de integração, mostra-se necessária porque o que se tem é que quando de uma carga inicial de mensagens muito alta, o pool global tem dificuldade de manter um tempo médio de processamento adequado, causando degradação da performance. Existem alguns trabalhos já identificados na literatura na tentativa melhorar o desempenho dos motores, essa revisão teórica procura apresentá-las e propor alguns trabalhos futuros, dentre os já existentes destaca-se um algoritmo que visa minimizar o tempo médio de processamento (SELLARO et al., 2017), e outro que otimiza o dimensionamento de Threads, por meio de simulação (HAUGG; FRANTZ, 2018), de modo que minimiza o tempo total de execução de uma mensagem na a solução de integração.

2. Referencial Teórico

Em sistemas computacionais, quando há um grande acúmulo de processos esperando para serem executados na fila, podem surgir gargalos de desempenho. É necessário, o dimensionamento do sistema para tratar o surgimento destas filas. A área que estuda o surgimento e o melhor tratamento é a teoria das filas (ADAN; RESING, 2002). Um sistema de filas é composto por, clientes, processo de chegada, processo de atendimento, número de atendentes, fila e disciplina da fila (HILLIER; LIEBERMAN, 2013). Os clientes são provenientes de uma população, quando a população é muito grande, ela é definida como infinita, conforme mostra figura 1. O processo de chegada é definido pelo comportamento das chegadas de clientes no sistema, podendo ser: determinístico ou estocástico. Quando o número de chegadas e o momento onde os clientes chegam ao sistema é conhecido, o comportamento é determinístico. Quando as chegadas apresentam comportamento aleatório, o comportamento é estocástico. Este comportamento pode

ser representado através de uma distribuição de probabilidade (CAMERON; TRIVEDI, 1986).

O processo de atendimento segue o mesmo conceito do processo de chegada, pode ser determinístico ou estocástico. Porém, geralmente ocorre de forma estocástica e também por uma distribuição de probabilidade. O número de servidor representa a quantidade de servidores disponíveis para realizar o atendimento aos clientes que estão aguardando na fila. A quantidade de servidores pode ser finito ou infinito. A fila é onde os clientes aguardam até que algum atendente esteja disponível. A disciplina da fila define a ordem pela qual os clientes são selecionados da fila para o atendimento. Esta disciplina normalmente segue o modelo FIFO,ou seja, o atendimento é realizado de acordo com a ordem de chegada (PRADO, 1999).

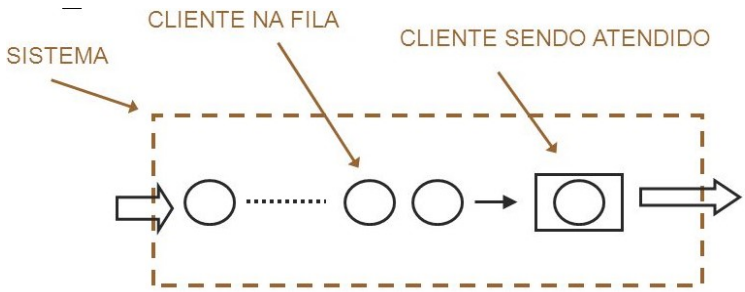

Figura 1 – Comportamento de uma fila.

Uma das análises e simulações mais elaboradas é a proposta por Hohpe e Woolf, que descreve como os pedidos dos clientes são processados em uma cafeteria. O processo começa por um cliente solicitando um pedido para o caixa, que então registra o pedido no sistema e o adiciona a uma fila de pedidos. Um pedido pode incluir bebidas quentes e frias, que são preparadas por diferentes baristas. Quando todas as bebidas que correspondem à mesma ordem foram preparadas, elas serão entregues pelo garçom. Cada pedido tem uma bandeja associada a ela, que é usada para entregar ao cliente, ou seja um mesmo garçom realizou o pedido este o efetuará a entrega do pedido encerrando assim o pedido, observe a Figura 2.

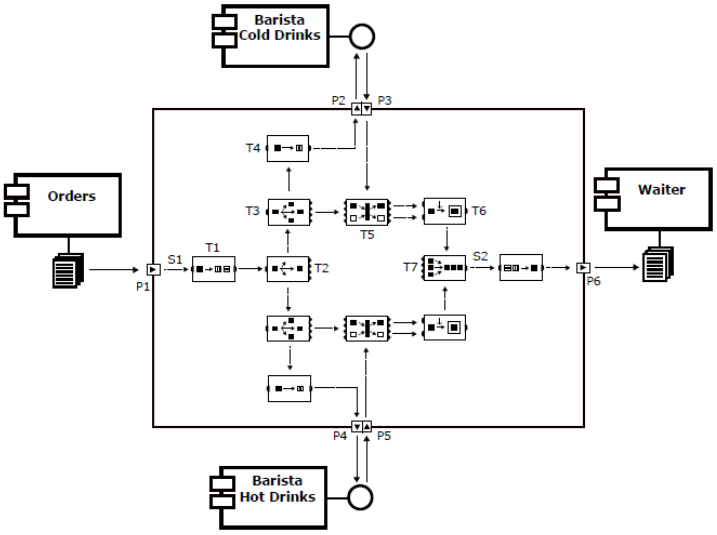

Figura 2 – Exemplo de uma solução de integração, o problema do Café.

Para resolver problemas como o do café, de integrar aplicações é que surge as plataformas. E no contexto de pesquisa aqui abordado, a maioria delas oferece esta disciplina FIFO combinado com um modelo de arquitetura Pipes and Filters (ALEXANDER, 1977), conforme a Figura 3 onde Pipes são os canais de mensagens e os Filters as unidades de trabalho atômicas que implementam um padrão de integração concreto, processa dados encapsulados em mensagens, sendo que a próxima unidade de trabalho só pode ser executada assim que a anterior já ter sido, além de possuir um motor de execução.

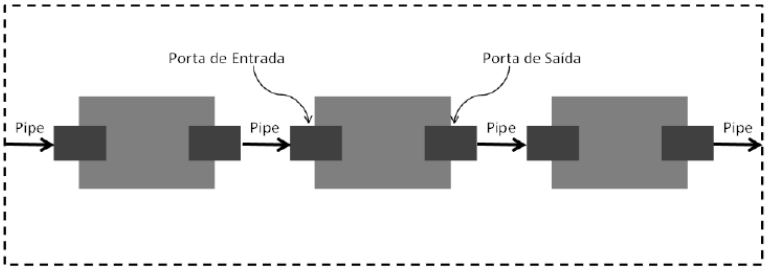

Figura 3 – Arquitetura Pipes e Filters (SELLARO et al., 2017).

O motor de execução é o responsável pela execução de uma solução de integração. Seu elemento central é o orquestrador, que é responsável pela coordenação de todas as atividades presentes em uma execução. A Figura 4 mostra o funcionamento do motor de execução da plataformas em geral, onde há uma fila na qual recebe de entrada diversas unidades de trabalho, as quais são

provenientes de uma solução de integração, configurado com um pool Global de Threads (FRANTZ; CORCHUELO; ROOS-FRANTZ, 2016). A fila, tem como função armazenar as unidades de trabalho a serem processadas, pelas Threads do pool, seguindo a disciplina FIFO (FRANTZ; QUINTERO; CORCHUELO, 2011).

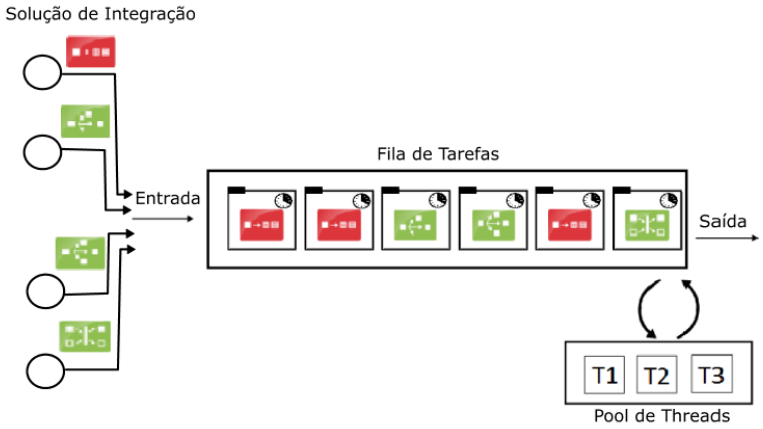

Figura 4 – Funcionamento dos Motores em Geral.

As unidades de trabalho representam a instância dela na qual está anotada com a informação de quando essa instância pode ser executada. Pool de Threads representa a quantidade de servidores que estão atendendo a fila de tarefas, podendo ser um único ou múltiplos. No caso de ser um único pool, sempre que houver a entrada de novas unidades de trabalho na também única fila, as Threads são informadas, e então, ocorre o processo de concorrência, onde apenas uma das Threads disponíveis será encarregada de realizar a execução da unidade de trabalho. A partir disto, a Thread consulta se a unidade de trabalho pode ser executada imediatamente, se isso for possível, ela retira a unidade de trabalho da fila de tarefas e realiza a execução. Caso contrário, a unidade de trabalho permanece na fila de tarefas e a Thread passa para o estado de espera, até o surgimento de uma nova entrada na fila de tarefas (FRANTZ; QUINTERO; CORCHUELO, 2011).

Agora observando que se esse processo submetido a uma alta taxa de entrada esse único pool Global, ficará de forma significativa preso tentando resolver a entrada de novas mensagens, visto que existe uma única fila. É necessário um alto número de Threads para tentar suprir essa demanda e nem sempre é possível dada a concorrência entre elas. O motor configurado com pool Global mostra-se ineficiente quando a tempo médio de processamento e consequentemente ao desempenho (SELLARO et al., 2017). Já quando possível e optado pela escolha de pools locais de Threads acontece que dada a uma alta taxa de entrada o esquema de distribuição das unidades de trabalho para as Threads é parecido, porém existem múltiplos pools que costuma ser divididos um para cada unidade de trabalho, existirão portanto uma fila para cada unidade e Threads disponíveis apenas para execução dela, evita-se aqui um excesso de Threads para as primeiras unidades de trabalho fato que ocorre com o uso de pool Global.

3. Desafios

Diante do exposto são dois os desafios encontrados na literatura, primeiramente é qual dos modelos de pool optar, se Global ou local, nessa escolha surge então o segundo desafio, que seria o correto dimensionamento das Threads, afim de executar a solução de integração num menor tempo médio possível, mantendo-se o desempenho. Até recentemente se tinha o conhecimento do uso de pools locais, porém poucas plataformas o adotaram como a forma de execução do motor. Em um trabalho recente, Sellaro et al. (2017) propuseram um algoritmo que utiliza a meta-heurística Particle Swarn Optimization (PSO), no qual mostra-se a possibilidade de utilizar pools locais para resolver o problema encontrado no pool Global. A escolha de PSO é justificada pelos autores por se tratar de um algoritmo estocástico para otimização de funções não-lineares de alta dimensionalidade e com variáveis contínuas (BRATTON; KENNEDY, 2007). O PSO foi introduzido por Eberhart e Kennedy (1995), e foi inspirado no comportamento social de organismos biológicos, mais especificamente na habilidade de algumas espécies de animais de trabalhar em conjunto para localizar boas regiões com fontes de alimento, assim como ocorre em cardumes e em bandos de pássaros (BRATTON; KENNEDY, 2007). É baseado em um enxame de partículas, que se movem pelo espaço e se comunicam para determinar uma direção de busca ideal. Segundo Eberhart e Kennedy (1995) o PSO tem melhor desempenho computacional do que outros algoritmos e tem menos parâmetros para ajustar, facilitando sua implementação. O PSO vem sendo utilizado com sucesso na solução de problemas da ciência e da engenharia devido à sua simplicidade, eficácia e robustez (RODRIGUEZ; BUYYA, 2014).

Referendando o algoritmo proposto por Sellaro et al. (2017) ele tem indícios de ser promissor, pois observa as unidades de trabalho a serem executadas levando em consideração seu tempo de execução e taxa de entrada. Dentro de uma solução de integração, existem diversas unidades de trabalho que são executadas em menor tempo, que poderiam ser resolvidas com menos rapidez se comparado a uma mais lenta. Nesse sentido é que usando PSO é possível uma alternativa ao pool Global, dividindo as tarefas em pools locais, dessa forma cada tarefa possuiu o seu pool, devendo ter seu tamanho compatível com o seu tempo de execução, taxa e bem como um bom desempenho da solução. Umas das ambiguidades que podem ser observadas é que é necessário um algoritmo para optimizar a solução de integração quanto ao número limitado de Threads (FRANTZ et al., 2014). Porém sabe-se que até o presente momento não existia uma maneira de definir o melhor número de recursos computacionais, nem matematicamente, tampouco computacionalmente, seja para pools locais ou Global e essa necessidade mostra-se importante.

Sugere-se de que seja elaborado uma proposta, que seja capaz de ao verificar qualquer que seja, a solução de integração, bem como um fluxo mínimo de mensagens a ser consideradas, que optimize um número de Threads mais próximo do ideal. Algo que caminha nesse sentido é o proposto por Haugg e Frantz (2018). Nessa proposta é observado uma forma de sabendo previamente as características da solução de integração é possível, predizer por meio de simulação uma melhor previsão de Threads para um pool Global, num contexto usando a tecnologia Guaraná.

Destaca-se que são duas as situações, uma para otimizar as Threads num pool global e outra para os locais. São duas configurações distintas de execução dos motores e precisam de abordagens diferentes. É necessário para se ter uma boa análise, fazer simulações com o uso do algoritmo PSO e verificar se há melhora significativa do desempenho dos motores, no uso de pools locais, há pouco produzido nesse aspecto (SELLARO, 2017).

Salienta-se aqui que seria de igual forma produtiva executar algoritmos similares com outras meta-heurísticas e comparar seus resultados. Outro problema que pode ser formulado é que se sabendo um número de Threads para o pool Global, já que existem pesquisas nessa direção, é de se usar esse valor obtido como parâmetro para o algoritmo PSO ou até mesmo com uma nova meta-heurística. São dois caminhos que podem ser seguidos e vislumbram um resultado encorajador.

4. Conclusão

Atualmente com o crescimento na utilização de aplicações empresariais nas empresas, surge cada vez mais a necessidade de integração entre essas aplicações. Esta integração é realizada através de diferentes plataformas. Elas executam as soluções de integração através dos motores de execução. Eles são fundamentais para o desempenho das soluções, definir qual modo de operação será escolhido, seja pool global ou local, assim como buscar o número ideal de Threads para a execução destes motores tem grande importância. As plataformas são ferramentas de software que devem oferecer um desempenho adequado e ainda economizar os recursos computacionais.

O motor de execução das plataformas de integração é o componente responsável pela execução das soluções de integração produzidas por essas ferramentas e um dos problemas apresentados é a baixa eficiência da configuração de pool Global quando submetido a uma alta taxa de mensagens, requisitando mais Threads, porém só aumentá-las também é oneroso e caro, dimensionar bem essas Threads é um desafio, afim de chegar no número ideal. Aliado a isso existe a possibilidade em algumas plataformas de se optar pela configuração de pools locais, algo que a primeira vista pode fugir do problema encontrado no Global, mas também requer que seja dimensionado, não só a quantidade total, mas também quanto deve ser alocado em cada pool, um desafio.

Nesse artigo, referenciamos a utilização da técnica de otimização meta-heurística Particle Swarm Optimization (PSO) para modelar o problema e perca de desempenho dos motores na execução de soluções de integração. Com o problema modelado no PSO, um algoritmo que faz o mapeamento das tarefas para os recursos computacionais, buscando aquele que promove o menor tempo médio de execução com uma quantidade limitada de recursos computacionais, visando assim uma melhoria no desempenho, sem a oneração do uso dos recursos computacionais. O algoritmo considera a complexidade computacional da unidade de trabalho e a heterogeneidade na capacidade computacional dos recursos.

O algoritmo proposto nos motiva a aprofundar o estudo, realizando simulações na execução de

soluções de integração e comparando o desempenho do PSO em relação aos outros algoritmos. Possibilitando uma melhor análise se realmente optar por pools locais resolve o problema de desempenho de forma satisfatória. Para também descobrir o número ideal de Threads, este artigo sugeriu o desenvolvimento de uma proposta tendo como base a já formulada, para converter modelos conceituais em modelos de simulação de forma automática para a tecnologia Guaraná. Esta proposta seria de uma maneira expandir abrangendo a todas as plataformas.

Além disso, possibilitaria através de experimentos, descobrir o número ideal de Threads para o pool Global. Ideias preliminares mostraram que o aumento de Threads não agrega melhora no desempenho do motor de execução. Isso ocorre, pois, cada solução precisa de quantidades diferentes de Threads. Percebe-se, neste caso, a necessidade de conhecer previamente o número de Threads antes da implantação da solução de integração, algo difícil de se obter.

Como trabalhos futuros pretende-se executar parte ou todas as questões levantadas por essa revisão teórica, utilizando-se do PSO, e outra heurística para que num comparativo entre elas seja possível ver melhora e se houver qual delas é a melhor abordagem para o problema do pools locais e uma metodologia para otimizar um motor configurado com pool Global.

5. Agradecimentos

Este Trabalho teve o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Referências

ADAN, I.; RESING, J. Queueing theory. [S.l.]: Eindhoven University of Technology Eindhoven, 2002.
ALEXANDER, C. A pattern language: towns, buildings, construction. [S.l.]: Oxford University Press, 1977.
BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. In: IEEE. Swarm Intelligence Symposium, 2007. SIS 2007. IEEE. [S.l.], 2007. p. 120–127.
CAMERON, A. C.; TRIVEDI, P. K. Econometric models based on count data. comparisons and applications of some estimators and tests. Journal of applied econometrics, Wiley Online Library, v. 1, n. 1, p. 29–53, 1986.
DOSSOT, D.; D'EMIC, J.; ROMERO, V. Mule in action. [S.l.]: Manning Publications Co., 2014.
EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: IEEE. Micro Machine and
Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on. [S.l.], 1995. p. 39–43.
FISHER, M. et al. Spring integration in action. Manning Publications Co., 2012.
FRANTZ, R. et al. Desafios para a implantação de soluções de integração de aplicações empresariais em provedores de computação em nuvem. XIX Jornada de Pesquisa, UNIJUI, p. 1–11, 2014.

FRANTZ, R. Z.; CORCHUELO, R.; ROOS-FRANTZ, F. On the design of a maintainable software development kit to implement integration solutions. Journal of Systems and Software, Elsevier, v. 111, p. 89–104, 2016.
FRANTZ, R. Z.; QUINTERO, A. M. R.; CORCHUELO, R. A domain-specific language to design enterprise application integration solutions. International Journal of Cooperative Information Systems, World Scientific, v. 20, n. 02, p. 143–176, 2011.
GOSLING, J. The Java language specification. [S.l.]: Addison-Wesley Professional, 2000.
HAUGG, I.; FRANTZ, R. Geração automÁtica de modelos de simulação para soluções de integração. CRICTE, 2018.
HILLIER, F. S.; LIEBERMAN, G. J. Introdução à pesquisa operacional. [S.l.]: McGraw Hill Brasil, 2013.
HOHPE, G.; WOOLF, B. Enterprise integration patterns: Designing, building, and deploying messaging solutions. [S.l.]: Addison-Wesley Professional, 2004.
IBSEN, C.; ANSTEY, J. Camel in Action. [S.l.]: Manning Publications, 2010.
LINTHICUM, D. S. Enterprise application integration. [S.l.]: Addison-Wesley Professional, 2000.
MESSERSCHMITT, D. G.; SZYPERSKI, C. et al. Software ecosystem: understanding an indispensable technology and industry. MIT Press Books, The MIT Press, v. 1, 2005.
PRADO, D. S. d. Teoria das filas e da simulação. Belo Horizonte, MG: Editora de Desenvolvimento Gerencial, v. 2, 1999.
PUSUKURI, K. K.; GUPTA, R.; BHUYAN, L. N. Thread reinforcer: Dynamically determining number of threads via os level monitoring. In: IEEE. Workload Characterization (IISWC), 2011 IEEE International Symposium on. [S.l.], 2011. p. 116–125.
RODRIGUEZ, M. A.; BUYYA, R. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. IEEE Transactions on Cloud Computing, IEEE, v. 2, n. 2, p. 222–235, 2014.
SELLARO, D. F. Particle swarm optimization para agendamento de tarefas na integração de aplicações empresariais. V SFCT, v. 14, 2017.
SELLARO, D. F. et al. Task scheduling optimization on enterprise application integration platforms based on the meta-heuristic particle swarm optimization. In: ACM. Proceedings of the 31st Brazilian Symposium on Software Engineering. [S.l.], 2017. p. 273–278.
TANENBAUM, A. S. Modern operating system. [S.l.]: Pearson Education, Inc, 2009.