Processamento de Work Units pelo Motor de Execução da Plataforma de Integração Guaraná

Alessandra Lucero Silva, Fabricia Roos-Frantz, Rafael Z. Frantz

Departamento de Ciências Exatas e Engenharia, DCEEng, Unijuí, 98700-000, Ijuí, RS

alelucero182@gmail.com, fabriciar@gmail.com, rzfrantz@gmail.com

Abstract. *The present work intends to understand how the processing of work units is impacted by increasing the amount of threads available to the runtime system of the Guaraná platform. For that, we analyzed experimental data regarding the amount of work units processed by the Guaraná runtime system. By performing statistical tests for software engineering data sets, it has been found that adding more threads to a certain amount, more work units are processed. However, by continuing to add threads, fewer work units are processed until processing stabilizes.*

Resumo. *O presente trabalho pretende compreender como é impactado o processamento de work units ao aumentar a quantidade de threads disponíveis ao motor de execução da plataforma Guaraná. Para tanto, foram analisados dados experimentais referentes à quantidade de work units processadas pelo motor de execução do Guaraná. Realizando testes estatísticos voltados para conjuntos de dados de Engenharia de Software, verificou-se que adicionar mais threads até certa quantidade, mais work units são processadas. No entanto, ao continuar adicionando threads, menos work units são processadas, até que o processamento se estabiliza.*

1. Introdução

O campo de pesquisa conhecido como Integração de Aplicações Empresariais (EAI) preocupa-se com a criação de mecanismos adequados para resolver o problema de integrar aplicações do ecossistema de software das empresas [Ritter et al 2017]. Uma solução de EAI busca fazer estas aplicações operarem entre si, sincronizando dados ou criando novas funcionalidades, sem que as aplicações envolvidas tenham suas antigas funcionalidades prejudicadas. Criar uma solução de EAI não é uma tarefa trivial, pois geralmente as aplicações envolvidas não foram desenvolvidas considerando uma possível integração com outra aplicação [Frantz et al 2011] e [Linthicum 2000].

As plataformas de integração são softwares desenvolvidos para dar suporte à integração de aplicações, possibilitando o desenvolvimento e a execução de soluções de integração. A ferramenta de uma plataforma de integração que executa a solução de integração é denominada motor de execução. Para que as mensagens sejam processadas, o motor de execução cria work units associadas a cada mensagem a ser processada utilizando recursos computacionais, e o motor as executa baseado no seu modelo de implementação [Frantz et al 2016].

Sobre outros trabalhos que pesquisam acerca de plataformas de integração podem ser citados o trabalho de Freire et al (2019) que avaliam a performance de 4 plataformas de integração open-source (dentre elas o Guaraná) considerando as dimensões *message processing*, *hotspot detection* e *fairness execution* e tem como objetivo ranquear as

plataformas analisadas conforme seu desempenho e Haugg et al (2019) que propôs uma metodologia e uma ferramenta para gerar modelos de simulação baseado em teoria das filas a fim de obter a quantidade ideal de threads para executar uma solução de integração que otimizam o desempenho, e utilizou a plataforma Guaraná para comparar os dados reais da plataforma com os dados da simulação.

Neste trabalho pretende-se responder a seguinte questão de pesquisa: qual o impacto do aumento do número de threads disponíveis para o motor de execução da plataforma Guaraná no número de work units processadas ao executar uma solução de integração? A hipótese é que quanto mais threads o motor pode utilizar, maior será a capacidade de processamento de work units pelo motor de execução. Para verificar esta hipótese, se analisou um conjunto de dados experimentais relativos ao número de work units processadas pelo motor de execução da plataforma Guaraná ao executar a solução de integração Café, com diferentes quantidades de threads disponíveis ao motor.

O restante do artigo está organizado da seguinte forma: na Seção 2, discute-se sobre EAI, plataformas de integração e funcionamento dos motores de execução; na Seção 3 é apresentado o experimento; na Seção 4 são apresentados os testes estatísticos realizados sobre os dados experimentais; na Seção 5 são apresentadas as interpretações sobre os testes estatísticos realizados e, por último, na Seção 6 são apresentadas as considerações finais.

2. Integração de Aplicações Empresariais

A plataforma de integração Guaraná é formada por uma linguagem de domínio específico, um kit de desenvolvimento, um motor de execução e uma ferramenta de monitoramento [Frantz et al 2011] e [Frantz et al 2016]. Uma linguagem de domínio específico fornece um número de conceitos que podem ser usados para elaborar a solução de um problema de um dado domínio, em outros termos, permite representar soluções de integração em modelos conceituais. Um kit de desenvolvimento oferece as ferramentas necessárias para implementação de uma solução de integração, transformando um modelo conceitual em um código executável. O motor executa uma solução de integração, já expressa em código executável, utilizando threads (recursos computacionais), de acordo com o modelo de execução que é implementado. O motor de execução é implementado conforme o modelo baseado em tarefas, que será explicado na subseção seguinte. A ferramenta de monitoramento permite detectar possíveis erros na solução e coletar estatísticas durante a execução desta solução [Frantz et al 2011] e [Frantz et al 2016].

2.1. Motor de Execução baseado em Tarefas

A arquitetura de uma solução de integração segue o estilo canais e filtros (pipes-and filters) e que, portanto, uma solução consiste num conjunto de tarefas, consideradas filtros, e um conjunto de slots, considerados canais. As tarefas recebem mensagens por um ou mais canais de entrada e as processam de acordo com um uma determinada lógica, e então geram mensagens de saída para um ou mais canais de saída. Os filtros são componentes independentes que não guardam informação de estado da solução. Dessa forma, quando uma solução é executada, se uma tarefa recebe em todos os seus canais de entrada todas a mensagens necessárias para habilitar seu processamento, o motor de execução gera uma work unit na fila de work units para esta tarefa, ou seja, gera uma work unit na fila de tarefas prontas para serem processadas. Assim, uma work unit é

formada pela identificação da tarefa correspondente associada as suas mensagens de entrada. O motor de execução aloca threads disponíveis para o processamento dessas work units utilizando a disciplina FIFO. Durante a execução de uma solução, quando uma tarefa recebe em suas entradas todas as mensagens necessárias para habilitar seu processamento, o motor gera uma work unit na fila de tarefas prontas para serem processadas. A work unit é uma tarefa associada as suas mensagens de entrada que será executada quando houver threads disponíveis para processá-la.

Para exemplificar, utilizamos uma solução de integração que modela o problema do Café, este problema é clássico na área de EAI e foi proposto por Hohpe (2005). Esta solução é composta por 12 tarefas e 6 portas; uma mensagem é considerada processada se o motor de execução processou as 20 work units produzidas para esta mensagem. O procedimento ocorre da seguinte forma: uma mensagem entra no fluxo de integração através de uma porta, e para que a mensagem seja processada e enviada para a próxima tarefa, é produzida uma work unit referente a esta porta e a esta mensagem. Logo que produzida, esta work unit é alocada em uma fila de work units, se é a primeira na fila e há threads disponíveis ela é executada.

Assim que executada é produzida uma nova work unit correspondente à próxima tarefa que precisa ser executada. Esta é enviada para a fila de work units, se é a primeira na fila e há threads disponíveis ela é executada, em outros termos, nenhuma work unit possui prioridade sore as demais. O processo termina quando todas as tarefas e portas foram executadas e a mensagem é considerada processada. Salienta-se que neste modelo de execução, baseado em tarefas, uma mesma thread não necessariamente irá processar todas as work units produzidas para uma mesma mensagem.

3. Dados Experimentais

Os dados experimentais provêm de um experimento executado em uma máquina com 16 processadores Intel Xeon CPU E5-4610 V4, com 10 cores, capacidade de execução de 20 threads, memória cache de 25 MB, 1.8 GHz, 32 GB, operando sistema Windows Server 2016 Datacenter 64 bits. A plataforma de integração experimentada é denominada Guaraná versão 1.4. Neste experimento, foi executada a solução de integração Café com distintas combinações de taxas de entrada de mensagens e threads, cada combinação repetida 25 vezes, cada uma durante 60 segundos. Neste trabalho foram analisadas as work units processadas pelo motor de execução, produzidas pela taxa de entrada de mensagens de 8000 msg/s. O conjunto de threads utilizado para verificar a hipótese é 2, 4, 6, 8, 10, 12, 14, 16, 18 e 20 threads.

Quanto às variáveis envolvidas, as independentes referem-se a taxa de chegada de mensagens, que é o fluxo de mensagens que chegam para serem processadas, e o número de threads, que são os recursos computacionais disponíveis para que o motor execute uma solução de integração. A variável dependente é o número de work units que foram produzidas para a taxa de entrada de mensagens e que que foram processadas pelo motor de execução da plataforma de integração em no máximo 60 segundos. Esse tempo é um parâmetro do experimento, correspondendo ao tempo estipulado para que o motor execute a solução.

4. Análise

Para decidir quais testes estatísticos seriam utilizados sobre os dados, primeiramente foi analisada a distribuição dos dados. Para tanto, foram construídos os gráficos de boxplot e densidade de kernel (utilizando o software RStudio [TEAM 2015]) para todas as combinações de taxas de mensagens e threads testadas. Neste trabalho os boxplots e as densidades de kernel completos foram suprimidos, são mostrados apenas os boxplots e densidade de kernel das work units executadas pelo motor de execução ao injetar 8000 msg/s, utilizando 8 threads e utilizando 16 threads. O boxplot e densidade de kernel referentes ao processamento de work units pelo motor de execução utilizando 8 threads são mostrados na figura 1a, bem como o boxplot e densidade de kernel referentes ao processamento de work units pelo motor de execução utilizando 16 threads.

Na Figura 1a, por meio do boxplot com dados do processamento com 8 threads percebe-se que há outliers na amostra de work units, que a linha central do boxplot, que se refere ao segundo quartil e à mediana está ligeiramente mais próxima do primeiro quartil. A linha superior tracejada é ligeiramente mais longa do que a linha tracejada inferior. Ao visualizar a densidade de kernel com dados do processamento com 8 threads é possível notar que há outliers pois no gráfico de densidade nota-se que a cauda é mais pesada à esquerda. Ainda neste gráfico, nota-se a possibilidade de a distribuição ser bi-modal. De fato, os dados brutos sobre work units não possuem distribuição normal. Importante ressaltar que realizar estas análises, permite ao pesquisador inferir sobre variância dos dados. Em outros termos, auxilia na verificação se os grupos possuem a mesma distribuição entre si, esta característica é denominada homocedasticidade.

No boxplot referente aos dados do processamento de work units com 16 threads percebe-se que há um outlier no conjunto de dados. A linha central no boxplot, mais próxima ao primeiro quartil, indica que os dados são assimétricos. Quanto às linhas tracejadas que partem do boxplot, vê-se que a cauda à esquerda é mais longa, do que a cauda à direita. No gráfico de densidade relativo ao processamento com 16 threads, as características mostradas pelo boxplot, confirmam-se, a presença do outlier abaixo do valor mínimo (calculado excluindo os outliers), torna a cauda à esquerda mais pesada.

Ao comparar os dois gráficos de densidade mostrados na Figura 1a percebe-se que as distribuições de ambas as amostras são distintas. Logo, os grupos de amostras são heterocedásticos entre si, não sendo possível utilizar testes estatísticos que exigem a presunção de distribuição normal e homocedasticidade.

Tendo em vista estas características dos dados, optou-se por realizar um teste estatístico de Análise de Variância (ANOVA). O teste ANOVA e o teste de comparação de médias robustos aos problemas de não-normalidade e heterocedasticidade foram executados por meio de um pacote para R desenvolvido por Patrick Mair e Rand Wilcox [MAIR, WILCOX 2016], descrito na pesquisa de Barbara Kitchenham [KITCHENHAM 2017]. Este teste estatístico é robusto aos problemas relacionados à não normalidade e à heterocedasticidade das amostras, pois as variâncias não são combinadas. Este tipo de ANOVA considera trimmed means, ou seja, calcula a média considerando uma porcentagem dos dados amostrais. Se ordenados de forma crescente, são excluídos do cálculo da média X% dos menores e maiores valores da amostra, que correspondem aos valores que podem ser considerados outliers.

Para a ANOVA apresentada neste trabalho, optou-se pelo cálculo de *trimmed mean* com 20% de exclusão. Para a execução a ANOVA robusta para a não normalidade e heterocedasticidade foi definido como hipótese nula é que disponibilizar mais threads ao motor de execução baseado em tarefas não implica em mais work units processadas, enquanto como hipótese alternativa foi definido que disponibilizar mais threads ao motor de execução baseado em tarefas permite que sejam processadas mais work units.

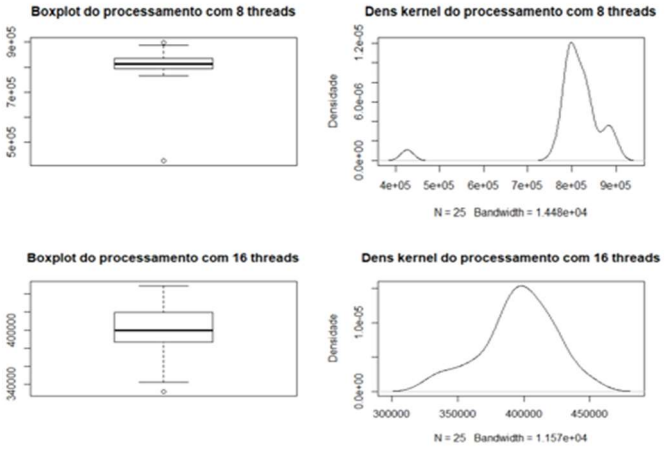

Figura 1: Visualização da distribuição dos dados

Conforme os resultados obtidos por meio da Análise de Variância, com $\alpha = 0.05$ e p = 0.001, a hipótese nula é rejeitada, pois o teste indica que há diferença estatística entre os grupos; e a hipótese alternativa é aceita. O tamanho do efeito dos tratamento é $\xi = 0.74$, ou seja médio, que conforme Mair e Wilcox (2016) pode ser interpretado por meio da escala de Cohen, disponível em Kitchenham (2017).

Como há diferença estatística entre os níveis do fator thread, foi realizada um teste post hoc que significa um teste de comparação de médias, utilizando trimmed means 20%. O resultado do teste de comparação de médias é apresentado na Tabela 1. Cada linha da Tabela 1 refere-se à comparação de trimmed means de dois níveis do fator thread. A primeira coluna mostra o nome dos níveis comparados, a segunda coluna expressa o limite inferior do intervalo de confiança do valor de p, a terceira coluna mostra o limite superior do intervalo de confiança do valor de p, e na terceira coluna, o valor de p.

5. Discussão dos Resultados

Por meio do teste de hipóteses, a hipótese nula foi rejeitada e a hipótese alternativa aceita. Então há diferença no processamento de work units pelo motor de execução que implementa o modelo baseado em tarefas utilizando distintas quantidades de threads. Por meio do teste de comparação de médias, é possível comparar quantidades de threads (níveis do fator) dois a dois, mostrado na Tabela 1. Com isso é possível inferir se ao aumentar a quantidade de threads disponíveis ao motor de execução, são processadas mais work units. Com nível de confiança 95%, quando o valor $p \leq 0.05$ e o intervalo de confiança não abrange o zero, há uma diferença estatística entre os níveis de threads.

Para auxiliar a interpretação dos resultados dos testes estatísticos, é apresentado na Figura 2 o gráfico referente a quantidade de work units processadas quando o motor dispõe de distintas quantidades de threads, à uma taxa de 8000 msg/s. Nessa figura, o eixo

x representa a variável independente threads e o eixo y representa a variável dependente work units processadas. Por meio da Tabela 1, percebe-se que o processamento de work units ao utilizar 2 threads é estatisticamente diferente de utilizar as demais quantidades de threads. Por meio do gráfico, vê-se que com 2 threads a quantidade de work units processadas é menor do que para as demais quantidades.

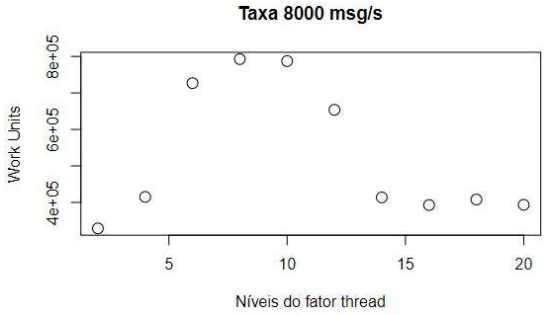

Figura 2: Work units processadas por quantidade de threads a uma taxa de 8000 msg/s

O nível 4 threads, é estatisticamente diferente dos níveis 2, 6, 8, 10, 12 e 14 threads, pois com 4 threads são processadas significantemente menos work units do que as quantidades citadas. Para as demais quantidades de threads, não há diferença estatística entre o processamento de work units. Isso significa que nem em todas as circunstâncias disponibilizar mais threads ao motor de execução, acarreta em maior processamento de work units. Esse resultado é um indício de que a hipótese alternativa, que havia sido aceita conforme o teste de hipótese, precisa ser refinada.

De acordo com o gráfico mostrado na Figura 2, ocorre o maior processamento de work units quando o motor utiliza 6, 8 e 10 threads. Por meio do teste mostrado na Tabela 1, o valor p = 0.00001 indica que há diferença estatística entre o processamento ao utilizar 6 e ao utilizar 8 threads, com 8 threads são processadas mais work units. O valor p = 0.03303 indica que há diferença estatística entre 8 e 10 threads, no entanto o intervalo de confiança IC [-59593.69; 11878.09] inclui o zero, então o tamanho do efeito não se aplica. Logo pode-se inferir que, a uma taxa de 8000 msg/s, o maior processamento de work units pelo motor de execução baseado em tarefas é quando estão disponíveis 8 threads.

Para uma taxa de 8000 msg/s, utilizar mais do que 8 threads acarreta em diminuição de quantidade de work units processadas, pois com a adição de recursos computacionais, ocorre a concorrência entre threads por causa do gerenciamento de recursos. Entre 14, 16, 18 e 20 threads não há diferença estatística entre o processamento de work units, que é aproximadamente constante e muito próximo do processamento correspondente a 4 threads.

6. Conclusão e Trabalhos Futuros

Neste trabalho foi analisado o processamento de work units produzidas e processadas pelo motor de execução, com a taxa de entrada de mensagens 8000 msg/s e distintas quantidades de threads para verificar o impacto em adicionar mais threads na capacidade de processamento do motor, medido pela quantidade de work units processadas.

Verificou-se que para a taxa de 8000 msg/s, até 8 threads, quanto mais threads são disponibilizadas ao motor de execução, mais work units são processadas. A partir dessa

quantidade, ao adicionar mais threads, menos work units são processadas, e depois o processamento de work units estabiliza-se, logo, adicionar mais threads não diminui e nem aumenta o processamento. O presente trabalho diferencia-se do trabalho de Haugg et al (2019) quanto ao seu principal objetivo que é caracterizar a performance do motor de execução, enquanto que o de Haugg preocupa-se em determinar a melhor configuração de threads para otimizar a performance do motor.

Como o conjunto de dados analisado é reduzido, as inferências sobre a performance global são limitadas. Em trabalhos futuros pretende-se estender as inferências acerca do processamento de work units pelo motor de execução baseado em tarefas analisando um conjunto maior de dados brutos, com delineamentos experimentais mais complexos e continuar utilizando testes estatísticos robustos para Engenharia de Software.

Tabela 1: Comparação de trimmed means 20%

Comparação entre grupos	Limite Inferior	Limite Superior	Valor p
thread_10 vs. thread_12	37820.69	203839.71	0.00006
thread_10 vs. thread_14	335682.17	409803.96	0.00001
thread_10 vs. thread_16	357287.9	421776.9	0.00001
thread_10 vs. thread_18	342428.71	419838.23	0.00001
thread_10 vs. thread_2	429214.84	490965.69	0.00001
thread_10 vs. thread_20	355958.3	428274.64	0.00001
thread_10 vs. thread_4	328801.72	421284.14	0.00001
thread_10 vs. thread_6	11409.1	90886.5	0.00018
thread_10 vs. thread_8	-59593.69	11878.09	0.03303
thread_12 vs. thread_14	169204.41	334621.32	0.00001
thread_12 vs. thread_16	186671.9	350732.5	0.00001
thread_12 vs. thread_18	177142.82	343463.71	0.00001
thread_12 vs. thread_2	257266.92	421253.21	0.00001
thread_12 vs. thread_20	188714.8	353857.73	0.00001
thread_12 vs. thread_4	169189.14	339236.33	0.00001
thread_12 vs. thread_6	-153030.08	13665.28	0.0082
thread_12 vs. thread_8	-227145.57	-62230.43	0.00001
thread_14 vs. thread_16	-13902.9	47481.57	0.07339
thread_14 vs. thread_18	-29193.85	45974.65	0.45977
thread_14 vs. thread_2	58255.11	116439.29	0.00001
thread_14 vs. thread_20	-15539.16	54285.96	0.07309
thread_14 vs. thread_4	-43195.44	47795.18	0.86452
thread_14 vs. thread_6	-360264.6	-282925.93	0.00001
thread_14 vs. thread_8	-431064.39	-362137.34	0.00001
thread_16 vs. thread_18	-41454.47	24656.6	0.38962
thread_16 vs. thread_2	50396.02	90719.71	0.00001
thread_16 vs. thread_20	-26836.07	32004.21	0.76732
thread_16 vs. thread_4	-56816.69	27837.76	0.24247
thread_16 vs. thread_6	-372722.15	-304047.05	0.00001

thread_16 vs. thread_8	-442166.68	-384613.72	0.00001
thread_18 vs. thread_2	47330.01	110583.59	0.00001
thread_18 vs. thread_20	-25830.45	47796.45	0.32386
thread_18 vs. thread_4	-52721.49	40540.43	0.66164
thread_18 vs. thread_6	-370171.86	-289799.47	0.00001
thread_18 vs. thread_8	-441372.25	-368610.29	0.00001
thread_2 vs. thread_20	-95716.64	-40230.96	0.00001
thread_2 vs. thread_4	-126560.79	-43533.88	0.00001
thread_2 vs. thread_6	-441964.27	-375920.67	0.00001
thread_2 vs. thread_8	-510921.51	-456974.62	0.00001
thread_20 vs. thread_4	-61903.99	27756.92	0.20555
thread_20 vs. thread_6	-378813.19	-303124.14	0.00001
thread_20 vs. thread_8	-449480.42	-382468.11	0.00001
thread_4 vs. thread_6	-371171.19	-276619.08	0.00001
thread_4 vs. thread_8	-443501.23	-354300.23	0.00001
thread_6 vs. thread_8	-112463.28	-37547.92	0.00001

Referências

Frantz, R. Z., Corchuelo, R., and Roos-Frantz, F. (2016). On the design of a maintainable software development kit to implement integration solutions. *Journal of Systems and Software*, 111:89–104.

Frantz, R. Z., Reina Quintero, A. M., and Corchuelo, R. (2011). A domain-specific language to design enterprise application integration solutions. *International Journal of Cooperative Information Systems*, 20(02):143–176.

Freire, D. L., Frantz, R. Z., Roos-Frantz, F., and Sawicki, S. (2019). Survey on the runtime systems of enterprise application integration platforms focusing on performance. *Software: Practice and Experience*, 49(3):341–360.

Haugg, I. G., Frantz, R. Z., Roos-Frantz, F., Sawicki, S., and Zucolotto, B. (2019). Towards optimisation of the number of threads in the integration platform engines using simulation models based on queueing theory. *Revista Brasileira de Computação Aplicada*, 11(1):48–58.

Hohpe, G. (2005). Your coffee shop doesn't use two-phase commit [asynchronous messaging architecture]. *IEEE software*, 22(2):64–66.

Kitchenham, B., Madeyski, L., Budgen, D., Keung, J., Brereton, P., Charters, S., Gibbs, S., and Pohthong, A. (2017). Robust statistical methods for empirical software engineering. *Empirical Software Engineering*, 22(2):579–630.

Linthicum, D. S. (2000). *Enterprise application integration*. Addison-Wesley Professional.

Mair, P. and Wilcox, R. (2016). Robust statistical methods in r using the wrs2 package. *Behavior research methods*, pages 1–25.

Ritter, D., May, N., and Rinderle-Ma, S. (2017). Patterns for emerging application integration scenarios: A survey. *Information Systems*, 67:36–57.

Team, R. et al. (2015). Rstudio: integrated development for r. *RStudio, Inc., Boston, MA URL http://www. rstudio. com*, 42:14