Análise da Escalabilidade de Middleware Para Internet das Coisas em Ambientes Centralizados e Distribuídos

Bruno Schmidt Marques, Rodrigo da Rosa Righi

[1]Universidade Vale do Rio dos Sinos (UNISINOS), São Leopoldo – RS – Brasil

contato@brunomarques.com.br, rrrighi@unisinos.br

Abstract. *Um dos middlewares mais utilizados para gerenciamento das requisições RFID normalmente é instalado em uma única máquina servidora. Isto pode limitar a sua eficiência e até mesmo sua aplicabilidade. A execução deste middleware em um ambiente de computação distribuída pode resolver este problema. Assim, neste trabalho é proposto um arcabouço de avaliação de desempenho em IoT chamado TET (Time-based Evaluation Test).*

1. Introdução

O Fosstrak[1] é um dos *middlewares* mais usados para prover o EPCIS (*Electronic Product Code Information System*), conectando as leituras de *tags* RFIDs com as aplicações de usuário. Ele funciona sobre a arquitetura cliente-servidor, que está sujeita a problemas de escalabilidade. A medida que o número de requisições aumenta, a performance pode ficar comprometida por limitações de *hardware*, pois o número de clientes muitas vezes não é conhecido de antemão. A execução do Fosstrak em um ambiente de nuvem (*cloud computing*) pode resolver o problema de escalabilidade, mas é necessário que se determine exatamente quando a arquitetura distribuída é vantajosa em relação a arquitetura centralizada. Assim será possível prover elasticidade à aplicação, isto é, a capacidade de aumentar ou diminuir os recursos conforme necessário, sem comprometer a performance. Este trabalho apresenta TET - *Time-based Evaluation Test* - que objetiva analisar a escalabilidade do *middleware Fosstrak* usando a métrica tempo. O resultado do trabalho é a indicação de quais módulos do *Fosstrak* são mais onerosos e sob quais condições.

2. Sistema TET - Time-Based Evaluation Test

O sistema projetado para a análise está descrito na figura 1. As tags RFID se comunicam apenas com o leitor, que se por sua vez transmite as informações ao *middleware*, que está em um servidor dedicado. O *middleware*, por trabalhar com o padrão EPCGlobal, deve não apenas receber e filtrar as informações de acordo com as regras de negócio, mas também armazená-las em um banco de dados, permitindo consultas no futuro. Estas requisições partem do cliente, um outro computador que executa a ferramenta de testes. Esta ferramenta não só faz as requisições ao servidor como também mede e armazena os tempos de resposta do *middleware* em um banco de dados local. A medição dos tempos ocorre no cliente que envia as requisições ao *middleware*, a fim de minimizar os efeitos da análise na performance dos servidores. Ela deve ser feita imediatamente antes do envio das requisições e imediatamente após o recebimento das respostas a estas requisições. A diferença entre os tempos será o tempo total de execução do teste.

[1]http://code.google.com/p/fosstrak/

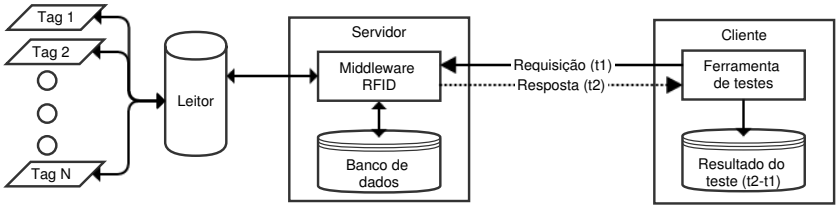


Figura 1. Arcabouço de testes para o middleware de Internet das Coisas

O *middleware* que gerencia as requisições e as informações do leitor é o Fosstrak 1.2. O *Fosstrak* não é um componente único, mas sim um conjunto de vários softwares com funções específicas que devem ser instalados e configurados para juntos promoverem todas as funções necessárias a implementação do EPCGlobal. Existem três módulos básicos dentro do *Fosstrak*: ALE (*Application Level Events*), EPCIS (*Electronic Product Code Information Services*) e *Capturing Application*. Também existem ferramentas satélites que são necessárias para configuração, como o *fc-webservice*, utilizado para configurar o ALE. O ALE é o módulo que se comunica com os leitores e recebe as informações dos marcadores. Ele filtra e agrega as informações conforme definido pelo *Capturing Application*. Uma vez que as informações tenham sido filtradas e agregadas, elas são encaminhadas ao EPCIS. Esse encaminhamento é feito pelo *Capturing Application*, um módulo no qual é possível utilizar a linguagem de programação Drools para definir as regras de negócio. Por fim, o EPCIS é o módulo que gerencia o repositório de informações. Ele recebe as informações do ALE e as armazena em um banco de dados MySQL. Ele também recebe e responde as requisições dos clientes *web*. Os testes foram executados em duas arquiteturas: (i) centralizado, onde todos os módulos do *middleware* concentram-se em um único computador, e (ii) distribuído, onde parte dos módulos é executada em um segundo computador, idêntico ao primeiro em termos de *hardware* e sistema operacional. A figura 2 ilustra a implementação da arquitetura distribuída.

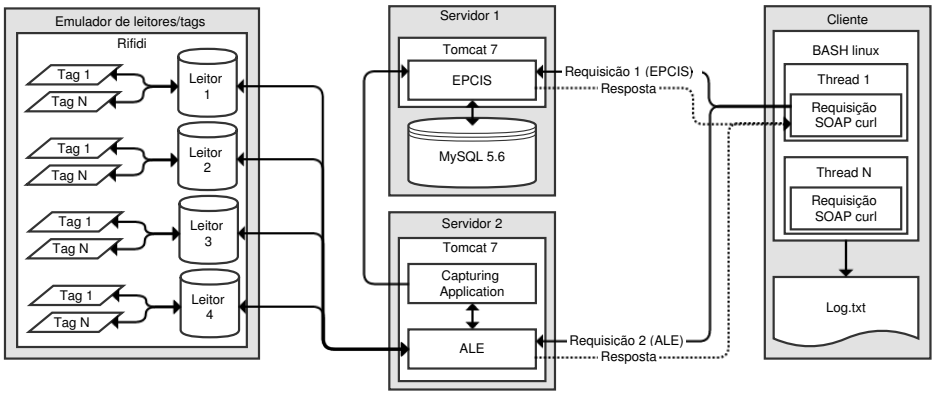


Figura 2. Implementação tecnológica em arquitetura distribuída

2.1. Metodologia de Testes

Os testes foram executados utilizando um computador que simula o comportamento de centenas de clientes solicitando informações ao *middleware*. Este computador possui um shell de interpretação BASH e o programa Curl. O shell é responsável por instanciar *threads* para realizar requisições simultaneamente e sequencialmente enquanto o Curl envia uma única requisição SOAP via rede para o servidor. O BASH mede o tempo

do relógio local antes do envio das requisições, envia os parâmetros ao Curl, aguarda a resposta de todas requisições, mede o tempo novamente e grava a diferença entre os tempos em um arquivo de *log*.

O ciclo completo de monitoramento pode ser explicado em quatro tempos: T1, T2, T3 e T4. Em T1 o cliente instancia todas as *threads* que vão solicitar informações ao *Fosstrak* através de scripts BASH. Foram instanciadas de 1 a 256 *threads* em cada arquitetura (centralizada e distribuída) para cada um dos conjuntos de carga (quatro, oito e 16 tags), onde cada *thread* enviou de uma a 64 requisições sequencialmente, graduados em escala 2^n. Em T2 as *threads* enviam as requisições ao *middleware*, e a velocidade é determinada pela tecnologia de comunicação em rede, como qualidade do *link*, conflito de pacotes, velocidade de processamento dos equipamentos de rede, etc. Em T3, o *middleware* passa a processar internamente as informações solicitadas pelas *threads*, o que pode levar muito tempo dependendo do montante de solicitações e do poder de processamento do servidor. Por fim, em T4 o *middleware* responde as requisições das *threads* e o ciclo acaba. Se o teste que estiver sendo executado for de apenas uma requisição por *thread* o bash anota o tempo de execução total (final - inicial) em um arquivo de *log*, junto de outras informações de execução. Se o teste tem mais requisições a serem executadas o bash não mede o tempo e um novo ciclo recomeça.

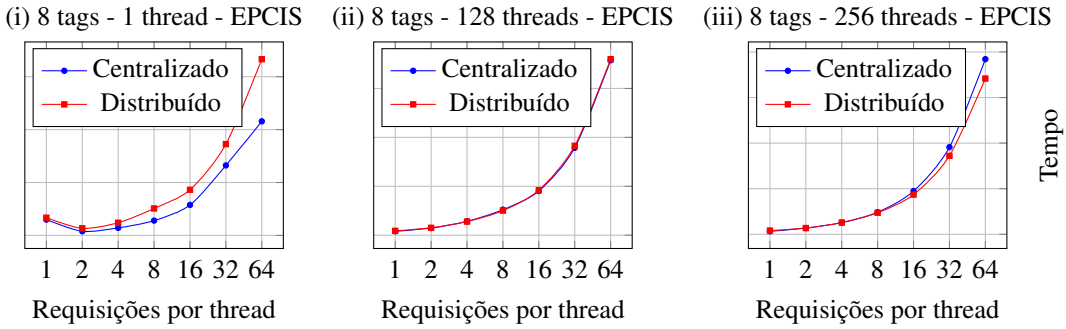
Segundo o EPCGlobal, existem dois tipos de requisições que podem ser feitos ao *middleware*: (i) ALE e (ii) EPCIS. A consulta ao ALE leva um ciclo de coleta, que neste estudo foi configurado para dois segundos, padrão do *Fosstrak*. Isso significa que uma requisição do tipo ALE pode levar até dois segundos para iniciar o processo de resposta, já que esse seria o tempo máximo até o fim do ciclo. A requisição do tipo EPCIS é feita ao repositório do *Fosstrak*, portanto não depende de nenhum ciclo e está relacionada com a performance do módulo EPCIS e do banco de dados. Baseado no estudo de [Gomes et al. 2014], que analisou o *Fosstrak* em um ambiente centralizado, foram determinados os seguintes parâmetros de testes: (a) **Número de leitores:** 4; (b) **Número de tags por leitor:** 1, 2 e 4; (c) **Número de *threads* concorrentes:** 1 a 256; (d) **Número de requisições por *thread*:** 1 a 64; (e) **Tipos de requisições:** ALE e EPCIS.

3. Resultados

Os testes com requisições ALE indicam que a arquitetura distribuída tem desempenho inferior a centralizada. Na maioria dos testes o servidor precisava de mais tempo para processar e responder as requisições na arquitetura distribuída. Isso pode estar relacionado ao fato de que todas as requisições eram respondidas simultaneamente ao fim de um ciclo de evento e a arquitetura distribuída utiliza mais recursos de rede que a arquitetura centralizada. No entanto os testes com muitas *threads* mostraram resultados aparentemente aleatórios, gerando travamentos do módulo ALE, o que indica que o servidor já estava trabalhando no seu limite nas requisições anteriores, e questões de sistema operacional como gerenciamento de memória, escalonamento do processador e transferência de dados podem ter interferido na avaliação.

As requisições EPCIS são mais rápidas que as requisições ALE e por isso são mais sensíveis ao desempenho de rede, já que a mínima interferência pode causar um atraso relativo muito grande. Inicialmente a arquitetura centralizada é mais rápida que a distribuída, porém com o aumento de *threads*/requisições por *thread* a arquitetura dis-

tribuída se tornou mais rápida, conforme testes demonstrados na figura 3. Uma possível hipótese para o comportamento obtido é de que nas cargas pequenas existem poucos dados a serem processados e a transferência de dados pela rede é a parte mais onerosa do sistema. Sendo assim, a arquitetura centralizada possuiu uma vantagem sob a distribuída, já que transfere a informação do ALE ao EPCIS localmente. A arquitetura distribuída precisa trafegar os dados entre diferentes servidores, o que é um processo mais lento que a transferência local. Porém a medida que a carga do sistema aumenta é necessário mais processamento, e então a transferência de dados deixa de ser o aspecto mais oneroso do processo. Quanto a escalabilidade, portanto, a arquitetura distribuída se mostrou mais apropriada, já que tende a ser cada vez mais rápida que a centralizada. O ponto em que a arquitetura centralizada deixou de ser a mais vantajosa pode servir de referência em futuros estudos dos sistemas RFID/IoT, onde define-se o momento exato no qual distribuir os módulos do *Fosstrak* em diferentes servidores se torna mais vantajoso.



.

Figura 3. O gráfico *(i)* mostra melhor desempenho da arquitetura centralizada. O gráfico *(ii)* mostra equilíbrio nas arquiteturas. O gráfico *(iii)* mostra melhor desempenho da arquitetura distribuída.

4. Conclusão

Esta pesquisa apresentou o método de avaliação TET, um das primeiros métodos de avaliação com foco em escalabilidade em IoT. Foram obtidos dados que indicam cientificamente quando é obtido o gargalo de execução do *Fosstrak*. Portanto, a decisão de uso ou não de computação em nuvem deixa de ser empírica. Os valores-chave de requisições podem ser usados como referência em estudos futuros, poupando tempo e dando uma direção precisa aos pesquisadores no campo de performance do *middleware*.

Referências

Gomes, M. M., Righi, R. d. R., and da Costa, C. A. (2014). Future directions for providing better IoT infrastructure. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp '14 Adjunct*, pages 51–54, New York, New York, USA. ACM Press.