Avaliação de desempenho de *IO Threads* no serviço de Enqueue SAP

Clébio Dossa[1], Rodrigo Righi[1]

[1]Programa de Pós-Graduação em Computação Aplicada – Universidade Do Vale do Rio dos Sinos (UNISINOS)

clebiodossa@gmail.com, rrrighi@unisinos.br

Resumo. *O serviço de Enqueue da SAP, usado para o controle de locks, proporciona a configuração de múltiplas threads com o intuito de melhorar o desempenho. A análise executada nesta pesquisa quantifica os tempos médios para criação de locks usando distintas quantidades de threads na Enqueue. Os resultados apresentam os melhores tempos obtidos e os melhores cenários para execução do serviço.*

1. Introdução

Threading é um método eficiente para obter desempenho com chips *multi core* ou *multi CPU*. Nesse contexto, o Serviço de Enqueue da SAP[1] é crítico devido ao desempenho necessário no processamento de suas atividades. Para tal, ele pode fazer uso de recursos de múltiplas unidades de processamento. Algumas partes deste serviço podem ser paralelizadas para a melhoria de desempenho, entretanto, outras possuem um caráter estritamente sequencial. Segundo a lei de Amdahl [Amdhal 1967], a melhoria do desempenho do serviço executando tarefas em paralelo é limitada ao desempenho das tarefas executas de maneira sequencial, assim sendo, pode não existir o incremento de desempenho aumentado o paralelismo do serviço. Em outras palavras, é possível que incremento de threads possa até prejudicar o desempenho de Enqueue [Robinson 2007]. Nesse panorama, com as observações de Amdahl e baseado na descrição de Robinson, onde a maior eficiência do uso de recursos de threading depende do hardware, esta pesquisa analisa o desempenho de Enqueue com base no uso de diferentes números de *IO Threads* configuradas usando servidores com diferente número de unidade de processamento.

2. Metodologia de testes

Os testes foram realizados em duas etapas. A primeira com o serviço de Enqueue instalado em no servidor com 2 CPUs e a segunda com a Enqueue instalada em um servidor com 4 CPUs, ambos com 2 unidades de processamento lógico por CPU e com 16GB de memória RAM. Nos dois testes executou-se a geração de carga através do *load runner* através de um servidor com 4 CPUs de mesma configuração.

Para cada um dos cenários, duas baterias de testes foram executadas. A primeira execução *load runner* dispara 5000 solicitações de criação de locks em 50 processos paralelos. A segunda bateria executa 10000 solicitações de locks com 100 processos

---

[1] SAP Standalone Enqueue Service: http://help.sap.com/saphelp_nw73ehp1/helpdata/en/47/e0a62341744ac5e10000000a42189d/frameset.htm

paralelos. Está quantidade te locks ocupa uma área de 5 a 10Mb de memória na tabela de locks da Enqueue, isso representa 30% do tamanho padrão entregue pela SAP, desta forma, uma quantidade razoável de locks. Cada um dos testes foi executado 3 vezes com cada configuração de threads e os resultados médios foram considerados.

3. Resultados

O servidor com 4 CPUs apresentou resultados médios de tempo de execução menores comparado com o servidor de 2 CPUs usando uma ou duas *IO Threads* configuradas.

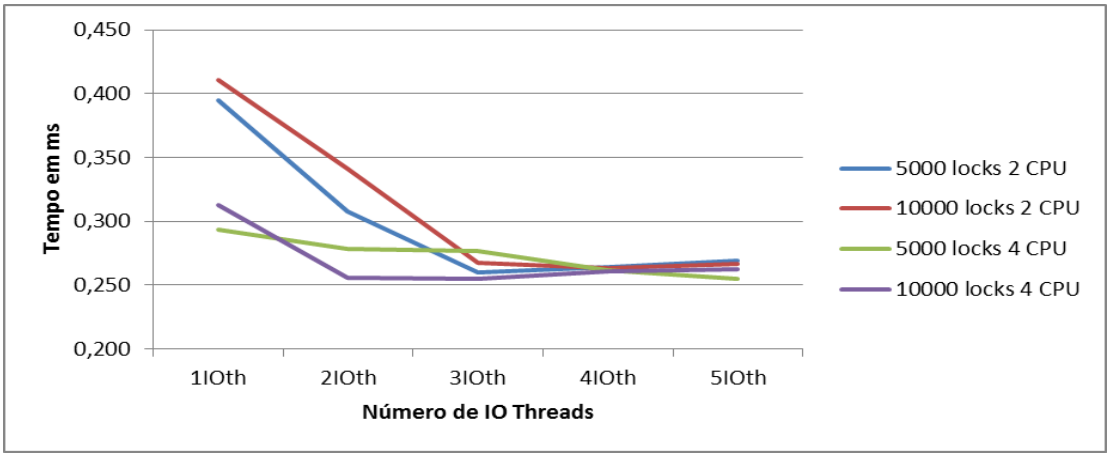

Figura 1. Tempo de criação de cada lock nos diferentes cenários

Acima de 3 ou 4 *IO Threads* não há significância em servidores com maior capacidade de processamento, conforme mostrado na Figura 1, pois os processos de natureza sequencial do serviço, como por exemplo, o *Listener* e *Work Thread* podem estar sendo o gargalo no aumento de desempenho e, desta forma, aumentando as *IO Threads* ou mesmo o número de unidades de processamento não tem um melhor desempenho. O que nota-se é a degradação do desempenho acima de 5 *IO Threads*.

4. Conclusão

A arquitetura do serviço de Enqueue, que divide suas atividades em distintas threads, se mostra eficiente em servidores multi-processados, usufruindo de forma positiva o uso de paralelismo. O melhor tempo de criação de um lock não teve grande diferença nos dois cenários. Entretanto, servidores com maior capacidade de processamento obtiveram melhor desempenho com poucas IO Threads.

Como contribuição, esta pesquisa mostra que o número de 3 ou 4 *IO Threads* tem o melhor desempenho em qualquer situação e o uso 5 ou mais *IO Threads* apresenta uma tendência a aumentar ou estagnar o tempo de criação de um lock. Este trabalho também demonstra que o melhor tempo obtido para criação de um lock é em torno de 0,255ms, informação esta de grande importância para *sizing* de sistemas SAP.

5. Referências

Amdahl, G. M. (1967) "Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities."

Robinson, Arch (2007) "Why too many threads Hurts Performance, and What to do About it" http://www.codeguru.com/cpp/sample_chapter/article.php/c13533/Why-Too-Many-Threads-Hurts-Performance-and-What-to-do-About-It.htm, Outubro 2014