Aspectos de Decisão de Projeto visando Desempenho Eficiente em Sistemas de Gerenciamento Autonômico para Infraestruturas que Suportam ATIS

Felipe Silvano Perini e Marcia Pasin

[1] Laboratório de Sistemas de Computação (LSC)
Centro de Tecnologia – Universidade Federal de Santa Maria (UFSM)
{*fsperini, marcia*}*@inf.ufsm.br*

1. Introdução

Sistemas que oferecem informação ao passageiro, coletivamente conhecidos como ATIS (*Advanced Traveler Information System*), estão cada vez mais populares. Entretanto, soluções de infraestrutura computacional atuais não são projetadas para tratar com dinamismo e complexidade inerentes nesses sistemas. De fato, gerenciamento manual é uma prática recorrente. Além da possibilidade de introdução de erros no sistema, esta atividade é trabalhosa, complicada, e pode conduzir o sistema a baixo desempenho e indisponibilidade. Há carência de infraestruturas computacionais elásticas, capazes de suportar diferentes adversidades simultaneamente (falhas, picos de carga e ociosidade). O resultado desta carência é traduzido em indisponibilidade e tempo inadequado de resposta.

Uma solução para esses problemas é gerenciar servidores da infraestrutura computacional através de um serviço autonômico [3], com a propriedade de auto-gerenciamento [2, 6, 5]. Isto significa que o sistema se auto-adapta de acordo com a situação atual, anterior (através da análise de um comportamento histórico) ou futura (usando mecanismos de previsão). Vantagens do uso de computação autonômica incluem mínima ou ausência de intervenção humana, e serviço contínuo e eficiente apesar da ocorrência de falhas e demandas escalares. Entretanto, a implementação de mecanimos de computação autonômica não é trivial e pode onerar o sistema devido ao processamento requerido para estratégias de decisão e coleta de métricas.

Neste artigo destacamos decisões de implementação visando desempenho eficiente para um sistema autonômico que adapta uma infraestrutura distribuída de suporte a um ATIS. A adaptação objetiva associação adequada entre economia de recursos e cumprimento de serviço a clientes. O trabalho está sendo desenvolvido no LSC/UFSM com apoio do Projeto RS-SOC 10/0049-7 Edital PRONEX FAPERGS/CNPq 2009-2012.

2. ATIS: aplicação-alvo

Um tipo especial de ATIS é um sistema que oferece informação prévia ao passageiro de transporte público urbano. Usuários contactam este sistema através de um *browser* em um PC ou através de *smart phones* ou PDAs para obter informação sobre rotas e posição atual de ônibus. Além de consulta de usuários, a infraestrutura computacional que suporta este ATIS deve processar dados recebidos sobre os ônibus em circulação e suas rotas.

Na infraestrutura física, um micro-controlador instalado em cada ônibus recolhe dados e os envia para o ATIS via tecnologia celular (GSM / GPRS). Quando a mensagem

chega ao servidor, os dados são utilizados para atualização de banco de dados com o novo *status* da frota. A Figura 1 mostra um possível esquema desta infraestrutura.

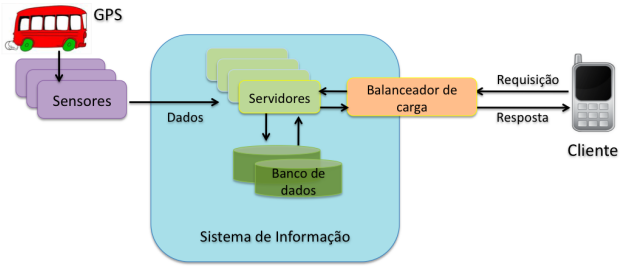

Figura 1. Visão geral do sistema de informação de transporte ao passageiro

Um serviço de balanceamento de carga distribui requisições de clientes para o conjunto de servidores usando uma política pré-estabelecida. Cada servidor de aplicação é associado a um servidor de banco de dados que armazena uma réplica da tabela com dados dos ônibus.

Este ATIS foi parcialmente implementado em [1], e atualmente está sendo estendido para permitir todas as funcionalidades aqui descritas. Quando posto em operação, um cenário que provavelmente irá surgir é que tal sistema está sujeito à demanda elástica: picos de carga, com milhares de usuários solicitando informação ao mesmo tempo (por exemplo, na saída de uma partida de futebol), contrastando com certos períodos onde a demanda decresce drasticamente (na madrugada, por exemplo). O sistema, então, precisa se adaptar sob estas condições para economizar recursos e prover um serviço eficiente mesmo quando ocorre sobrecarga.

3. Proposta de gerenciamento autonômico

O sistema autonômico que adapta a infraestrutura de servidores do ATIS é formado por um conjunto de serviços: balanceamento de carga, reconfiguração de servidores, detecção de falhas, detecção de ociosidade, reintegração de servidor após falha, transferêcia de estado. O serviço de detecção de falhas exclui de um grupo de servidores nodos suspeitos de falha. Requisições recebidas pelo balanceador de carga somente são propagadas para servidores operacionais. O serviço de reconfiguração de servidores ajusta automaticamente o número de servidores operacionais de acordo com a carga atual do sistema (considerando a ocorrência de ociosidade e picos de carga).

Este sistema foi implementado em linguagem Java usando o sistema de comunicação de grupo JGroups como ferramenta de suporte aos serviços autonômicos. Os serviços são construídos como módulos independentes e foram implementados de forma descentralizada para evitar ponto único de falha e gargalos. Mais detalhes são descritos em [4].

Para avaliar a implementação, foi realizado um teste inicial considerando o serviço de reconfiguração em um cenário livre de falha. O cenário inicial possui um servidor executando em uma máquina processando requisições de clientes. A medida que a quantidade de requisições aumenta, a quantidade de servidores servindo a aplicação aumenta

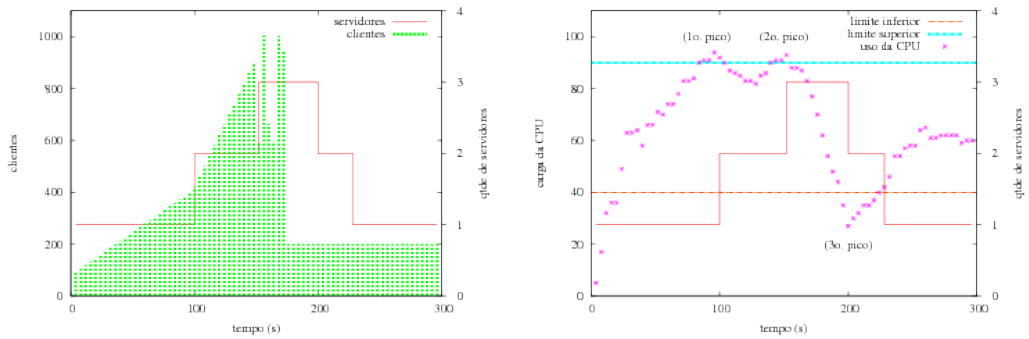

Figura 2. Reconfiguração da infraestrutura devido à mudança de carga

também. A figura 2 ilustra este cenário de teste através de gráficos mostrando requisições de clientes e taxa de ocupação da CPU.

A carga de requisições é aumentada para sobrecarregar a configuração inicial. Ocorre a inicialização automática de outro servidor, se a taxa de ocupação de CPU nos nodos operacionais atingiu o limite máximo de 90%. A tarefa de inicialização de servidor se repete até que o número máximo de nodos da infraestrutura é atingido (três nodos). Analogamente, se a carga de requisições diminuir (no experimento foi reduzida e mantida a 200 clientes ativos), a ocupação da CPU também reduz e quando atinge 40%, servidores são desalocados deste sistema.

4. Aspectos focados em desempenho eficiente

Durante a construção do sistema autonômico, decisões de projeto foram tomadas tendo em vista desempenho eficiente. Tradicionalmente sistemas autonômicos são um *software* complexo e não queremos que este serviço prejudique demasiadamente o desempenho esperado da aplicação-alvo. Por questões de espaço, seguem breves considerações sobre nossas decisões de implementação.

Comunicação unidirecional de métricas. O gerenciamento autonômico é feito com base em métricas coletadas na infraestrutura de servidores. A coleta de métricas usa chamadas do tipo *heartbeat*, onde servidores enviam periodicamente informações sobre valores a um processo monitor que avalia as métricas. As métricas são coletadas usando séries temporais, levando em conta não somente dados atuais, mas um conjunto de dados históricos. Para não onerar o sistema com troca de mensagens e gargalos, esses valores são coletados e computados localmente em cada nodo que compõe a infraestrutura.

Escrita sem sincronização. Servidores da aplicação-alvo processam dois tipos de transações: *read only* e escrita-cega. Transações do tipo *read only* não alteram o banco de dados e são geradas por requisições de usuários. Elas sempre têm uma mensagem de resposta associada. Transações de escrita-cega são, de fato, dados gerados por sensores instalados nos ônibus para atualizar o banco de dados sem qualquer operação de leitura. A aplicação-alvo precisa ser replicada para garantir tolerâcia a falhas, mas não requer garantias extremas de consitência. Protocolos de replicação de consistência fraca não oneram demasiadamente o desempenho do sistema, e são usados para replicar operações de escrita-cega na base de dados. Um aprimoramento que pode ser feito na implementação

atual é a aglutinação de várias mensagens de escrita em uma única operação sobre a base de dados. Cada mensagem que é recebida pelo sistema é armazenada em um vetor, em *cache*. Quando este vetor está completo, um conjunto de atualizações é processado de uma só vez.

Uso de servidores sem estado. Servidores requeridos pela aplicação-alvo são *multi-thread stateless* (sem estado), porque requisições de clientes são do tipo *read only*. Uma grande vantagem de servidores *stateless* é que a implementação e recuperação em caso de falha é mais fácil que servidores *stateful* pois não precisa recuperar estado conversacional. Além disso, devido ao fato de não precisar manter o estado, sob operação livre de falhas, o serviço para os clientes fica mais leve e é mais facilmente escalável. Do ponto de vista do gerenciamento autonômico, a implementação também é facilitada pelo fato de não precisar transferir estado conversacional em caso de adição de servidor na infraestrutura.

Escalonamento *round robin*. A estratégia de balanceamento de carga atual segue a política *round-robin*, onde a distribuição de tarefas obedece a ordem estabelecida em uma lista de servidores operacionais mantida pelo JGroups. Esta política pode ser modificada para permitir alternativas mais sofisticadas, que considerem informações de contexto da aplicação ou por uma abordagem que analise a carga atual individual de cada servidor para que ele receba uma nova tarefa.

5. Conclusões e trabalhos futuros

Neste trabalho, apontamos decisões de implementação para a construção de um sistema de auto-gerenciamento focado em adaptação de servidores. Suas principais características são prestação de serviços relacionados com aplicações sujeitas a falhas e demandas elásticas, e implementação de serviços através de algoritmos leves para não sobrecarregar o sistema. Um mecanismo de séries temporais é usado para computar métricas para evitar a ativação ou desativação desnecessária dos servidores da infraestrutura. Trabalhos futuros incluem a melhoria do módulo de reconfiguração com uma abordagem preditiva e a análise do impacto da execução do sistema autonômico sobre a aplicação-alvo.

Referências

[1] Bastos, R. e Jaques, P. (2010) ANTARES: Um Sistema Web de Consulta de Rotas de Ônibus como Serviço Público. Revista Bras. Computação Aplicada, v.2, pp. 41–56.

[2] Bouchenak, S., De Palma, N., Hagimont, D. e Taton, C. (2006) Autonomic Management of Clustered Applications, In: Proc. Cluster 2006. September 2006.

[3] Kephart, J. O. e Chess, D. M. (2003) The Vision of Autonomic Computing, In: IEEE Computer, vol. 36. pp. 41–50.

[4] Pasin, M., Perini, F. S. e Bazzan, A. L. C. (2011) Towards a Self-managed Distributed Infrastructure to an Advanced Traveler Information System, In: Anais do Autosoft / CBSoft 2011. São Paulo - Brasil. pp. 33–39.

[5] Roblitz, T. et al. (2004) Autonomic Management of Large Clusters and Their Integration into the Grid, In: Journal of Grid Computing - GRID, vol. 2, no. 3, pp. 247–260.

[6] Saleem, S. K. e Chen, S.-C. (2008) Towards a Self-configurable Weather Research and Forecasting System, In: Proceedings of ICAC 2008, pp. 195–196.