Um Sistema de Controle para Web Farms Baseado em Agentes de Software

Fernando Yuichi Tanaka
Universidade de São Paulo
tanaka@icmc.usp.br

Dilvan de Abreu Moreira
Universidade de São Paulo
dilvan@computer.org

Abstract

Recently, clusters of inexpensive computers are gaining popularity, especially among corporations and universities, as Web servers. Mainly because they offer a very attractive cost/benefit ratio. However, these clusters (known as Web Farms) are more complex, difficult to manage and, in the case of high available systems, need intensive monitoring to maintain their services always online. This article presents an architecture for monitoring and controlling Web Farms based on software agent technology. This architecture uses software agents to implement an autonomic system capable of monitoring and maintaining a collection of services offered by a Web Farm with reduced human intervention. A prototype system is being used in a 6 nodes 12 CPUs Web Farm running the Jboss J2EE Application Server.

1. Introdução

A popularização da Internet e os atuais avanços no desenvolvimento de diversos equipamentos computacionais, associados à redução dos custos de aquisição desses equipamentos, vêm propiciando a proliferação de *clusters* de computadores. Um *cluster* de alta disponibilidade, como uma *web farm* [11], possui como característica principal uma exposição transparente e quase contínua de dados, recursos e serviços aos seus clientes. Uma forma de se obter essa alta disponibilidade de dados, recursos e serviços em um *cluster* é ter um sistema autônomo que monitore e controle as variáveis do *cluster*. Nesse contexto, este artigo apresenta uma arquitetura e implementação de um sistema que utiliza a tecnologia de agentes de *software* para monitorar e controlar os estados e os serviços de um *cluster* (do tipo *web farm*), de modo a deixar os seus serviços sempre disponíveis.

Inicialmente na seção 2 são discutidos os agentes de *software*. Na seção 3, é apresentada uma breve descrição de *clusters* de computadores. Na seção 4, são descritos a computação autônoma e seus benefícios. Na seção 5, é apresentada a arquitetura do sistema. Na seção 6 são tecidos alguns comentários sobre trabalhos relacionados e na seção 7 são colocadas algumas considerações finais sobre o trabalho.

2. Agentes de software

No contexto computacional, os agentes foram propostos para facilitar a criação de softwares capazes de interoperar, ou seja, trocar informações e serviços com outros programas e, dessa forma, resolver problemas complexos [3].

Apesar do termo agente ser amplamente utilizado, ele não tem uma definição consensual, talvez por cobrir diversas áreas de investigação e desenvolvimento. Dessa forma, cada grupo de pesquisadores que trabalha com agentes de *software* procura explicar a sua própria utilização da palavra "agente", sendo que muitos abordam o termo levando em consideração algumas propriedades encontradas na literatura, tais como autonomia, habilidade social, pró-atividade e reatividade [9].

O termo agente, neste trabalho, é utilizado para se referir a uma entidade que funciona continuamente e de modo autônomo, em um ambiente onde podem existir outros processos e agentes [10].

3. Cluster de computadores

A idéia inicial dos *clusters* de computadores foi desenvolvida na década de 1960, mas só se popularizou nos últimos anos devido principalmente à redução dos custos dos equipamentos computacionais e à popularização da Internet. Um *cluster* de computadores pode ser definido como um tipo de sistema de computação distribuído ou paralelo que consiste de um grupo de máquinas e sistemas de software independentes e interconectados, incluindo equipamentos montados em rede, sistema operacional e bibliotecas de comunicação [8], que trabalham juntos para transparentemente prover serviços aos usuários.

Os *clusters* de computadores permitem uma grande flexibilidade de configuração, como, por exemplo, aumentar o número de nós e a capacidade de memória por nó para elevar o poder de processamento. Para tal, pode-se utilizar tecnologias padrão com custos relativamente baixos, permitindo um alto retorno do investimento neste tipo de sistema.

A implementação desse sistema foi desenvolvida em um *cluster* do tipo *web farm* com máquinas *desktops* de pouca confiabilidade. Esse *cluster* possui seis nós, um *director* (ou *web switch)* primário e um *director* secundário, como mostra a Figura 1. O *director* secundário faz o monitoramento do *director* primário, através do *heartbeat*, para prover redundância.

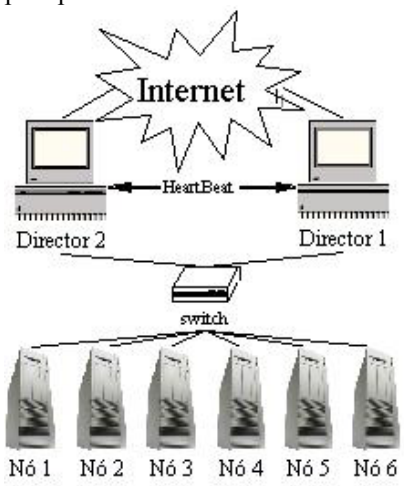

Figura 1. Arquitetura física do cluster.

4. Computação autônoma

O avanço e o crescimento da indústria de computadores vêm ocasionando o surgimento de infraestruturas que são heterogêneas e sofisticadas. Juntando-se a isso o crescimento da Internet, chegou-se a uma nova era de grande acessibilidade à informação. Esta explosão simultânea de informação e integração de novas tecnologias está levando a um aumento da complexidade no gerenciamento de sistemas. Isto ocorre até mesmo nos sistemas mais simples, aumentando os riscos para a segurança e para a geração de interrupções no sistema. Este contexto levou os pesquisadores da IBM a considerarem técnicas de gerenciamento baseadas em estratégias usadas pelo sistema nervoso autônomo, isto é, sistemas que controlam ações involuntárias do corpo [7]. Com base nesta idéia, a IBM propôs uma nova visão de computação para tratar a complexidade e heterogeneidade dos sistemas, denominada sistemas de computação autônoma ou sistemas autonômicos [7].

Um sistema autônomo pode ser definido como um sistema computacional que tem a capacidade de se auto-gerenciar e que funciona de forma independente, sem a intervenção humana [2] (ou com uma intervenção mínima). Para tal, é preciso que o projeto e a implementação dos sistemas de computação, *software*, armazenamento e suporte exibam alguns fundamentos básicos, tais como [7]: autonomia, flexibilidade, acessibilidade e transparência. A autonomia pode ainda ter as seguintes propriedades [2][12]: auto-configuração, auto-otimização, auto-tratamento e auto-proteção.

Com todos estes fundamentos e propriedades, a computação autônoma é uma solução apropriada para o gerenciamento de sistemas complexos [3].

4.1. Benefícios

A IBM fez um estudo e constatou que o custo de se treinar o pessoal para gerenciar um sistema é aproximadamente equivalente ao custo dos equipamentos utilizados e, estima-se que o custo de pessoal dobrará o de equipamentos nos próximos cinco ou seis anos [2]. Essas constatações e estimativas foram feitas com base nos sistemas atuais que não utilizam o conceito de computação autônoma. Nesse contexto, a tecnologia de auto-gerenciamento de sistemas autônomos poderá fornecer diversos benefícios, tanto para empresas quanto para usuários. Dentre esses benefícios, podem-se destacar [2][7]: redução substancial dos custos de operação e de pessoal especializado; redução da dependência da intervenção humana para manter sistemas complexos; redução da complexidade computacional para usuários finais; e redução da necessidade de manutenção do sistema.

Além dos benefícios em termos de pessoal, outros benefícios podem ser obtidos, sendo estes com relação às capacidades técnicas dos sistemas autônomos, tais como: aumento da capacidade computacional; colapso menos freqüente no sistema; aumento da estabilidade do sistema; e utilização balanceada dos processadores, de forma a evitar processadores ociosos.

5. Arquitetura do sistema proposto

O sistema proposto é composto por três tipos de agentes em cada nó do *cluster*: o Agente Controlador (AC), o Agente Monitor de Serviços (AMS) e o Agente Monitor de Nós (AMN). A função principal do AC é analisar e responder ou não às requisições dos usuários. Já o AMS tem como principal função o monitoramento e controle dos estados e serviços de um nó do *cluster*. E por fim, o AMN tem como principal função o monitoramento do funcionamento dos outros nós do *cluster*. Os AMSs e os AMNs podem estabelecer comunicações com os ACs dos *directors*.

Nesse sistema, como mostra a arquitetura esquematizada na Figura 2, o usuário acessa uma página em JSP (*Java Server Page*) de acesso e requisita a sua

autenticação. A requisição de autenticação é enviada a uma *Servlet*, que consulta uma base de dados e autentica ou não o usuário. Se o usuário for autenticado, essa *Servlet* envia uma página JSP contendo as opções que o usuário pode requisitar ao sistema, como: temperatura das CPUs de cada nó do *cluster*; rotação das ventoinhas existentes em cada nó; informações sobre disco rígido de cada nó (como espaço total, espaço utilizado, espaço em *cache* total, utilizada e livre, fabricante, modelo, etc.); informações sobre a memória de cada nó; informações de voltagem de cada nó; informações sobre as CPUs de cada nó.

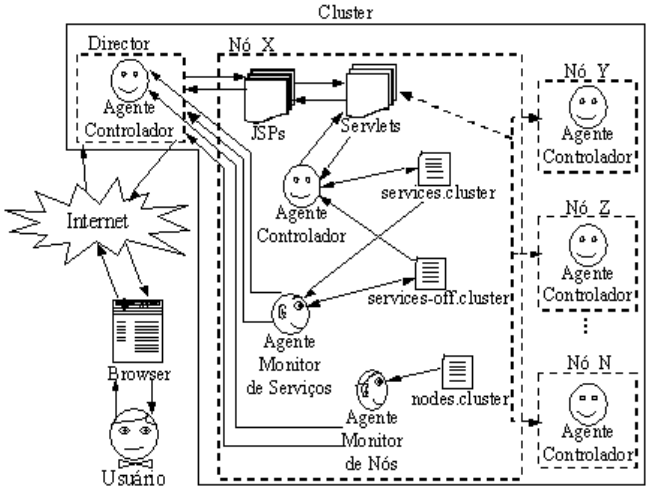

Figura 2. Arquitetura do sistema de controle.

Quando um usuário faz uma requisição, esta é enviada ao *director* (também conhecido como *web switch*), que por sua vez encaminha a requisição a um dos nós do *cluster* de acordo com um algoritmo de escalonamento. No exemplo da Figura 2, a requisição foi encaminhada para o Nó X. A seguir, a requisição chama uma *Servlet* do Nó X, que encaminha a requisição ao AC de cada nó do *cluster*. O AC de cada nó analisa a requisição e envia a informação requisitada, de acordo com sua competência, de volta a *Servlet*. A *Servlet* então envia o resultado da requisição ao usuário em forma de um documento HTML.

O AMS lê o arquivo "services.cluster", que fica no diretório /etc/, e monitora periodicamente todos os serviços que estão descritos nesse arquivo. Esse arquivo pode ser alterado pelo administrador via o sistema de gerenciamento ou manipulando diretamente através do console. Se um serviço S não estiver em funcionamento no nó N, o AMS tentar inicializar esse serviço. Se conseguir, o AMS continua o monitoramento dos outros serviços. Se não conseguir, o AMS coloca o serviço S no arquivo "services-off.cluster", que também fica no diretório /etc/, envia uma correspondência eletrônica ao administrador do *cluster* informando a parada do serviço S no nó N e se comunica com o AC dos *directors* informando que o serviço S não está mais em funcionamento no nó N. O AC dos *directors*, então, retira o serviço da lista de distribuição de requisições para que os *directors* não enviem mais requisições do serviço S para o nó N. Os serviços que estão fora de funcionamento em cada nó podem ser visualizados pelo usuário enviando uma requisição ao AC de cada nó via página *web*.

O AMN realiza um processo semelhante ao do AMS, sendo que a diferença é que ele lê quais nós estão ativos, no arquivo "nodes.cluster" e tenta contatá-los. Quando um nó não responde, o AMN avisa ao administrador (via *email*) e aos *directors,* para que não mandem mais requisições a esse nó.

6. Trabalhos Relacionados

Diversos trabalhos têm sido desenvolvidos no sentido de monitorar os vários recursos existentes em um cluster ou em redes de computadores. A seguir serão comentados alguns trabalhos que seguem essa linha de pesquisa.

Monit é um programa que faz o monitoramento e gerenciamento de processos, arquivos, diretórios e equipamentos em sistemas Unix. Ele realiza reparos e manutenções automáticas, como por exemplo, reiniciar um processo parado ou parar um processo por exceder a utilização de um recurso [4].

Mon é uma ferramenta utilizada para o monitoramento de serviços. Ele dispara mensagens de alerta quando detecta a parada de algum serviço [5].

Keepalived é um *software* ainda em desenvolvimento que faz o monitoramento dos nós de um cluster para fazer o controle da lista de distribuição de requisições do LVS (Linux Virtual Server) [1].

Como visto, tais sistemas apresentam algumas semelhanças com o sistema desenvolvido. Porém, nenhum deles apresenta o conjunto dessas funcionalidades, como exemplo o monitoramento dos recursos juntamente com o controle de distribuição de requisições do *director*.

7. Conclusão

Neste artigo apresentou-se uma arquitetura de um sistema baseado em agentes que monitora e controla os estados e os serviços de um *cluster* de computadores. Os agentes propostos foram implementados na linguagem *Java*, os serviços *web* em *Servlet* e a apresentação das informações em JSPs, de modo que o sistema é independente de plataforma.

Facilitando a visualização do estado de funcionamento do *cluster* e realizando automaticamente tarefas simples, que permitem que o sistema continue funcionando adequadamente (mesmo que partes dele falhem), este sistema autônomo diminui a necessidade de intervenção humana para a manutenção do *cluster* reduzindo seu TCO (*Total Cost of Ownership*). Ele, contudo, pode ainda atingir níveis maiores de autonomia em versões futuras.

8. Referências

[1] A. Cassen. Keepalived for LVS. Disponível em: <http://www.keepalived.org/>. Acesso em: 13 Set. 2004.

[2] B. Pereira, "Autonomic computing brings the healing touch to it", *Express Computer*, n.1, August 19, 2002.

[3] D. A. Moreira, and L.T. Walczowski, "Using software agents to generate VLSI layouts", *IEEE Intelligent Systems*, v.12, n.6, 1997, pp.26-32.

[4] J. Haukeland et al. Monit. Disponível em: <http://www.tildeslash.com/monit/>.Acesso em: 13 Set. 2004.

[5] J. Trocki. Mon – Service Monitoring Daemon. Disponível em: <http://www.kernel.org/software/mon/>. Acesso em: 13 Set. 2004.

[6] J. L. Wolf, P. S. Yu, "On balancing the load in a clustered web farm", *ACM Transactions on Internet Technology (TOIT)*, v.1, n.2, November 2001, p.231-261.

[7] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing", *IEEE Computer*, v.36 n.1, January 2003, p.41-50.

[8] M. Baker, A. Apon, R. Buyya, H. Jin, "Cluster Computing and Applications", *Encyclopedia of Computer Science and Technology*, Vol.45, Marcel Dekker, Aug. 2001.

[9] M. Wooldridge and N. Jennings, "Intelligent agents: Theory and practice", *Knowledge Engineering Review*, v.10, n.2, 1995.

[10] Shoham, Y., An overview of agent-oriented programming, In: Bradshaw, J.M., *Software Agents*, Cambridge: AAAI Press/ MIT Press, 1997, pp.271-290.

[11] V. Cardellini, E. Casalicchio, M. Colajanni and P. Yu, "The state of the art in locally distributed web-server system", *ACM Computing Surveys*, ACM Press New York, NY, USA, v.34, n.2, June 2002, pp.263-311.

[12] Z. Constantinescu, "Towards an Autonomic Distributed Computing System", *Proc. of the 14th Inter. Workshop on Database and Expert Systems Applications*, IEEE Computer Society, 2003, pp. 694-698.