Controle de Admissão e Diferenciação de Serviços em Clusters de Servidores Web

A. Serra[*1], D. Gaïti[2], K. Cardoso[1,4], G. Barroso[3], R. Ramos[4]

1 Institut National des Télécommunications
9 rue Charles Fourier – 91011 - Evry Cedex, France

2 LM2S – Université de Technologie de Troyes
12 rue Marie Curie – 10.010 - Troyes Cedex, France

3 Universidade Federal do Ceará
Av. da Universidade, 2853 – Benfica - 60020-181 - Fortaleza - CE-Brasil

4 Centro Federal de Educação Tecnológica do Ceará
Av. 13 de Maio 2081 Fátima - 60040-531 Fortaleza Ce Brasil

antonio.serra@int-evry.fr; dominique.gaiti@utt.fr; gcb@fisica.ufc.br, {klecius; ronaldo}@cefet-ce.br

Abstract. *The number of users of services deployed on the Internet has been increasing continually. This has been causing overload in a great amount of Web servers reducing QoS (Quality of Service) offered by service providers. Furthermore, users with different QoS aspirations have been forced to share available processing resources in a same way even they can pay more. This paper presents WS-DSAC, a diffserv-based admission control and load balancing mechanism conceived to improve QoS on Web server clusters. WS-DSAC has three main aims: to balance the imposed load, to guarantee different QoS levels and to use available resources in an effective way. We also present evaluation experiments showing the mechanism is able to perform these objectives.*

Resumo. *O número de usuários de serviços ofertados na Web vem aumentando continuamente provocando sobrecarga em um grande número de servidores reduzindo a QoS (Qualidade de Serviço) oferecida. Além disso, clientes com expectativas diferentes de QoS são forçados a compartilhar de forma igual os recursos de processamento disponíveis. Neste artigo, é apresentado um mecanismo de controle de admissão e de balanceamento de cargas concebido para aumentar a QoS em clusters de servidores Web. O mecanismo, denominado WS-DSAC tem três objetivos principais: balancear a carga imposta ao sistema, permitir a diferenciação da QoS oferecida aos clientes e utilizar de forma eficaz os recursos disponíveis. Experimentos são também apresentados mostrando que o mecanismo WS-DSAC é capaz de realizar os objetivos propostos.*

---

[*] Antonio Serra é financiado pela CAPES-Brasil.

1. Introdução

O número de usuários de serviços oferecidos na World Wide Web (WWW) tem crescido continuamente. Ao disponibilizar uma aplicação na Web, um fornecedor de serviços pode fazer face a uma quantidade inesperadamente grande de acessos. Isto pode provocar a sobrecarga, ou até mesmo um "crash", do servidor Web que provê o serviço, reduzindo conseqüentemente a qualidade do serviço (QoS) oferecida.

Levando em consideração que o ponto de sobrecarga (ponto de estrangulamento) vem se localizando cada vez mais nos recursos de processamento, devido ao aumento do conteúdo dinâmico (CGIs, Servlets, JSPs,…) oferecido na Internet, duas possíveis soluções para lidar com este problema e aumentar a QoS do usuário final são: a distribuição equilibrada da carga imposta pelos serviços em um certo número de computadores, ou seja, utilização de clusters com balanceamento de carga; e o uso de mecanismos de controle de admissão para evitar a sobrecarga ou possível "crash" de servidores [Cherkasova02] [Abdelzaher02].

Por outro lado, usuários com expectativas diferentes em relação a QoS desejada, mesmo podendo pagar mais caro pelos serviços utilizados, são forçados a compartilhar de forma igual os recursos de processamento disponibilizados para o provimento de um serviço na WWW. No nível de redes baseadas no protocolo IP, o diffserv (differentiated services) [Carpenter02] permite a diferenciação de serviços agrupando-os em classes e garantindo a cada classe certos parâmetros de QoS (perdas, retardo, variação do retardo). No nível de aplicação, para permitir que provedores de serviços possam diferenciar a QoS oferecida aos seus clientes, é necessária a utilização de mecanismos que permitam o uso diferenciado dos recursos de processamento disponíveis.

Neste artigo é apresentado um mecanismo de controle de admissão e de balanceamento de cargas concebido para clusters de servidores Web. O mecanismo permite a diferenciação de serviços através do agrupamento de requisições em classes diferentes de serviços. Um parâmetro de QoS denominado "Coeficiente de Reatividade" [Olejnik02] (diretamente relacionado ao tempo de resposta do cliente) é associado a cada classe de serviços pré-estabelecida. O mecanismo tem como objetivos principais: balancear a carga imposta pelas requisições atendidas, garantir a QoS estabelecida para cada classe de serviços e utilizar de forma eficaz os recursos de processamento disponíveis. Também são apresentados experimentos que mostram que o mecanismo é capaz de realizar estes objetivos.

Este artigo esta organizado da seguinte forma: na seção 2, é apresentado o contexto no qual este trabalho está inserido. Na seção 3, são discutidos aspectos ligados a flexibilidade e escalabilidade do mecanismo. Na seção 4, o mecanismo WS-DSAC é formalmente descrito. Na seção 5, são apresentados resultados de experimentos que mostram que o mecanismo é capaz de realizar os objetivos propostos. Finalmente, na seção 6, conclusões e perspectivas de trabalhos futuros são discutidas.

2. Contexto do Trabalho

O mecanismo WS-DSAC foi projetado para realizar o controle de admissão e o balanceamento de cargas em uma plataforma distribuída apresentada em [Serra04a]. A plataforma oferece diferentes níveis de QoS baseando-se na diferenciação de serviços.

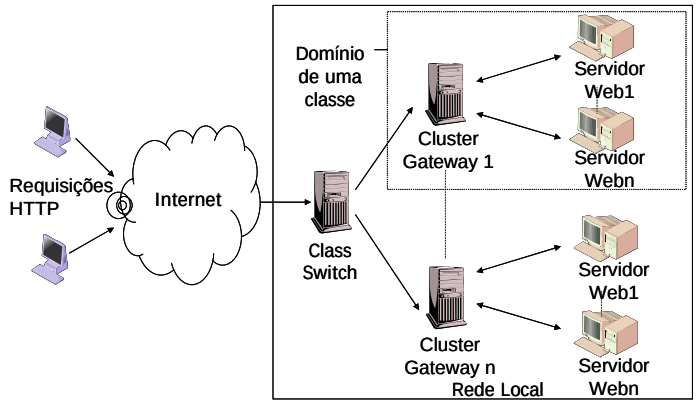


Figura 1 – Visão Geral da Plataforma.

2.1 Visão Geral da Plataforma

A plataforma (Figura 1) é composta por um conjunto de elementos básicos: "Class Switch", "Cluster Gateways" e "Servidores Web". O "Class Switch" é responsável pela classificação e pelo controle de admissão de novas requisições. Ele recebe requisições HTTP, identifica a classe de serviços e, utilizando o mecanismo WS-DSAC, envia cada requisição para um "Cluster Gateway" específico. O "Cluster Gateway" escolhe o servidor menos carregado para processar a requisição enviada pelo "Class Switch".

Os serviços são instalados em um certo número de servidores Web. Estes serviços podem ser compostos por serviços Web e objetos distribuídos. A plataforma oferece diferentes níveis de QoS, um nível diferente para cada classe de serviços. Requisições que chegam podem pertencer à diferentes classes de serviços. O administrador da plataforma associa a cada classe de serviços uma carga máxima que pode ser atingida pelo seu "class cluster".

2.2 Implementação da Plataforma

Um protótipo da plataforma proposta foi implementado utilizando-se a tecnologia Java. Os componentes da plataforma foram modelados como objetos distribuídos Java e serviços Web. O RMI (Remote Method Invocation) Java foi utilizado para dar suporte à comunicação entre os objetos distribuídos e o Tomcat 5 Servlet/JSP foi usado como container para os serviços Internet.

Os elementos da plataforma são classificados como: Mecanismos de Monitoramento; Mecanismos de Controle de Admissão e Balanceamento de Cargas e Serviços Administrativos.

Os mecanismos de monitoramento são compostos por três elementos básicos: o MA (Monitor Agent), o CRM (Cluster Resource Manager) e o GRM (Global Resource Manager) (Figura 2). O MA monitora e estima a carga de cada servidor Web e o CRM monitora e estima a carga do cluster de uma classe. O GRM solicita e mantém a informação de carga de cada CRM, ou seja, de cada cluster.

O MA possui três funções básicas. A primeira é de registrar o servidor Web em um domínio específico, ou seja, em um cluster de uma determinada classe de serviços.

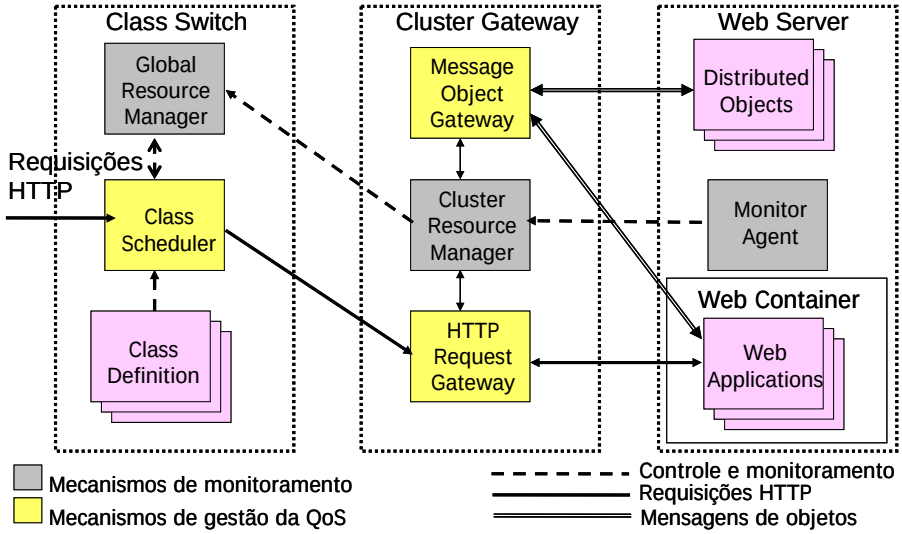


Figura 2 – Comunicação entre os elementos da plataforma.

Ele envia uma mensagem solicitando o registro para o CRM responsável pelo domínio da classe pedindo para adicioná-lo ao domínio.

A segunda função do MA é de monitorar a carga do servidor Web. O MA usa uma técnica de monitoramento baseada em um "Coeficiente de Reatividade" [Olejnik02]. Esta técnica dá uma idéia da tendência de carga do servidor estimando o tempo de espera de uma tarefa pela CPU. O MA possui um "thread" que passa o controle da CPU para um outro processo e espera ser novamente "acordado" pelo sistema. O tempo de espera do "thread" depende diretamente da carga do servidor. Para eliminar variações momentâneas da carga esta operação é repetida diversas vezes. Finalmente, é calculada a média do tempo de espera pela CPU (o "coeficiente de reatividade") que está diretamente relacionada com a carga do servidor. Na subseção 5.2 deste artigo, alguns experimentos realizados que mostram essa relação entre o "coeficiente de reatividade" e o tempo de resposta do cliente são apresentados.

A terceira funcionalidade básica do MA é de responder ao CRM quando este solicita periodicamente informações sobre a carga de cada servidor Web.

Cada domínio de classe ("Class Cluster Domain") executa um CRM (Cluster Resource Manager) que solicita informações de carga de cada servidor Web em intervalos constantes de tempo. Ele analisa todos os "coeficientes de reatividade" dos servidores do cluster e faz uma estimativa para a carga do cluster inteiro.

Os mecanismos de controle de admissão e balanceamento de cargas são compostos por três elementos: o CS (Class Scheduler), o HRG (HTTP Request Gateway) e o MOG (Message Object Gateway) (Figura 2). O CS é responsável pela classificação e pelo controle de admissão de requisições HTTP que chegam. O HRG envia as requisições HTTP redirecionadas pelo CS para o servidor menos carregado do cluster. O MOG é responsável pela interceptação e re-direcionamento das mensagens trocadas entre os objetos distribuídos dentro de um cluster mantendo a carga balanceada entre os diversos servidores do domínio.

O CS recebe requisições que são enviadas pelos clientes. Depois de classificar a requisição identificando a classe de serviços a qual ela pertence, ele verifica as informações de carga mantidas pelo GRM e, utilizando o mecanismo WS-DSAC, ele analisa se a QoS estabelecida para a classe da requisição pode ser fornecida. Se é possível garantir a QoS especificada, o CS redireciona a requisição para o "Cluster Gateway" eleito pelo mecanismo, senão ele retorna uma mensagem para o cliente informando que a QoS requerida para aquela classe de serviços não pode ser garantida.

O HRG (HTTP Request Gateway) recebe as requisições redirecionadas pelo CS. Depois ele decide qual servidor Web ira processar cada requisição baseando-se na informação de carga mantida pelo CRM.

A plataforma permite a coexistência de serviços Web e de objetos distribuídos que podem interagir conjuntamente para prover serviços aos clientes. Quando um objeto distribuído é instalado na plataforma, o serviço de administração utiliza a interface de comunicação distribuída do objeto para automaticamente criar seu "Message Gateway". O MOG (Message Object Gateway) intercepta as invocações de métodos remotos dentro de um cluster específico e, baseado na carga informada pelo CRM, ele redireciona as invocações para o servidor menos carregado. Desta maneira, se um serviço Web utiliza objetos distribuídos para resolver suas sub-tarefas, a carga imposta por estas sub-tarefas também será balanceada entre os servidores do cluster.

Um conjunto de serviços foi desenvolvido para permitir a administração da plataforma. Eles são agrupados em: Serviços de Instalação, Serviço de Definição de Classes e Serviços de Monitoramento dos Clusters.

Usando os serviços de instalação o administrador da plataforma pode instalar serviços Web e objetos distribuídos na plataforma. Os Serviços são replicados na plataforma e os mecanismos de balanceamento de carga são automaticamente criados.

O administrador pode definir diferentes classes de serviços, bem como as características de cada classe, utilizando o serviço de definição de classes. As características de cada classe estão diretamente relacionadas com o "coeficiente de reatividade" que é requerido para cada requisição de cliente pertencente àquela classe.

Os serviços de monitoramento de carga exibem informações estatísticas (taxa de requisições recusadas, média do coeficiente de reatividade por classe de serviços, etc.) sobre todo o sistema. Usando estas informações o administrador do sistema pode redefinir a estratégia de distribuição dos recursos entre os diversos clusters.

3. Flexibilidade e Escalabilidade da Plataforma

Um dos objetivos do mecanismo WS-DSAC é utilizar de forma eficaz os recursos de processamento disponíveis na infra-estrutura aumentando a QoS oferecida às requisições atendidas independentemente da classe de serviços a qual elas pertençam. Isto significa em outras palavras que, enquanto existirem recursos de processamento disponíveis e os parâmetros de QoS estabelecidos para cada classe de serviços forem respeitados, todos os recursos, mesmos os alocados prioritariamente para uma determinada classe, devem ser compartilhados de forma igualitária.

Para isto, uma realocação dinâmica de recursos entre as classes de serviços é necessária. Algumas soluções existentes propõem estratégias que trocam

temporariamente servidores de domínios. No entanto, por questões de flexibilidade e escalabilidade, o uso deste tipo de solução não é conveniente para a nossa plataforma.

Este tipo de solução não seria conveniente dado o fato de que a plataforma foi projetada de forma a permitir diferentes agrupamentos de servidores como podemos ver na figura 3. Para um cluster com um grande número de equipamentos, os servidores pertencentes à domínios de classes de serviços distintos podem ser agrupados fisicamente em diferentes subredes. Isto permite a utilização de um "dispacher" de nível 3 [Gan00] ("Cluster Gateway") para cada domínio, distribuindo melhor os cálculos de estimação de carga e os processamentos relativos ao redirecionamento dos pacotes das requisições. A arquitetura da plataforma permite que estas subredes estejam até mesmo geograficamente distribuídas. Outra vantagem é a minimização da latência de cada subrede provocada pela troca de mensagens de monitoramento.

Por este motivo a estratégia adotada para a realocação dinâmica de recursos entre as classes de serviços se baseia em uma mudança no modo de funcionamento de cada "class cluster". Em um determinado intervalo de tempo, recursos reservados prioritariamente para uma classe podem estar em um de três modos possíveis : "modo compartilhado" ; "modo exclusivo" ou "modo saturado".

Quando está no "modo compartilhado" o cluster da respectiva classe possui recursos disponíveis que podem ser utilizados por outras classes de serviços sem comprometer o contrato estabelecido com a classe "nativa" do cluster durante um intervalo de tempo pré-definido. Estando neste modo de trabalho o "class cluster" atende requisições de diferentes tipos de classes.

Quando o cluster passa ao "modo exclusivo", ele só aceita requisições da classe "nativa". Quando o servidor passa ao "modo exclusivo" significa que os níveis de cargas chegaram a um patamar onde, aceitar requisições de outras classes, pode implicar na rejeição de sessões da classe nativa. Isto dado ao fato que, se requisições de outras classes forem aceitas, recursos que são prioritariamente reservados para a classe nativa estarão sendo utilizados por sessões de outros tipos de classe.

Quando o cluster passa ao "modo saturado" ele não aceita nenhuma nova requisição. Isto significa que os recursos existentes não serão suficientes para garantir a QoS assegurada às requisições em processamento, caso o mesmo aceite novas requisições.

A mudança de modo de funcionamento é baseada em dois parâmetros do cluster: um limiar que determina o valor máximo do "coeficiente de reatividade" de uma classe

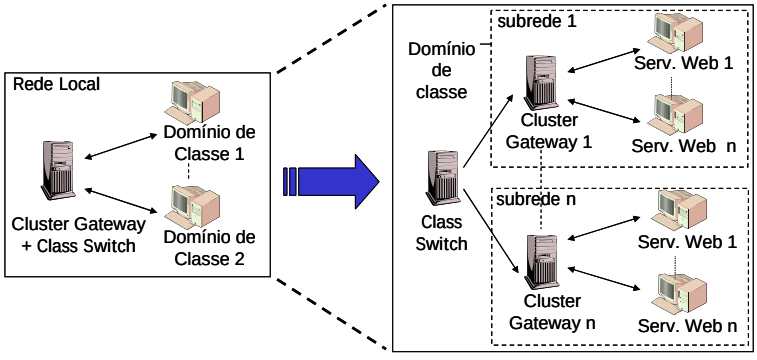


Figura 3 – Escalabilidade e Flexibilidade no uso do mecanismo WS-DSAC.

e um limiar dinâmico recalculado a cada intervalo de tempo. Este segundo limiar determina uma margem de segurança para que os contratos de QoS estabelecidos com a classe nativa sejam respeitados. Na próxima seção este mecanismo será apresentado formalmente.

4. O Mecanismo WS-DSAC

4.1 Objetivos do Mecanismo

O mecanismo WS-DSAC têm três objetivos principais:

- Realizar um balanceamento de cargas em dois níveis, entre os servidores de um cluster e entre um conjunto de domínios de clusters;
- Permitir a diferenciação de serviços em relação ao uso dos recursos de processamento disponíveis em clusters de servidores Web, garantindo assim diferentes tempos de resposta para diferentes classes de serviços;
- Utilizar de forma eficaz recursos de processamento disponíveis realocando de forma dinâmica os recursos entre classes diferentes de serviços enquanto estes não estão em níveis críticos de utilização de forma que, serviços de classes menos prioritárias, não serão sacrificados por serviços mais prioritários enquanto existirem recursos disponíveis.

4.2 Algoritmo WS-DSAC

O mecanismo WS-DSAC integra estratégias dos mecanismos discutidos em [Cherkasova02] e [Abdelzaher02]. Ele foi projetado de forma à respeitar as características de escalabilidade e flexibilidade da plataforma, discutida na seção 3 deste artigo. Formalmente o mecanismo WS-DSAC pode ser definido através dos seguintes parâmetros:

T1, T2, …, Tn,… - Uma seqüência de intervalos de tempo utilizados para tomar decisões em relação a admissão ou rejeição de requisições durante o próximo intervalo de tempo. Esta seqüência é definida pelo parâmetro "ac-interval";
N_c – Número de classes de serviços diferentes (class clusters) definidas dentro da plataforma, onde ($N_c > 1$);
N_{si} – Número de servidores alocados no domínio da classe "i", onde ($N_{si} > 0$);
R_i^{ac} – O valor limite que pode alcançar o "coeficiente de reatividade" da classe "i", onde ($1 < i <= N_c$). Este parâmetro estabelece o nível de utilização crítica do cluster da classe "i" a partir do qual o cluster passa a trabalhar em « modo saturado » ;
R_{ki}^{em} – O valor limite do "coeficiente de reatividade", recalculado a cada período k, que pode ser alcançado pelo cluster da classe "i" durante o k-ésimo período de tempo. A partir deste valor, o cluster passa a trabalhar em « modo exclusivo ». Enquanto o coeficiente esta abaixo deste valor o cluster responsável pela classe "i" pode compartilhar seus recursos de processamento com outras classes, caso contrário os recursos serão utilizados exclusivamente por requisições pertencentes a classe nativa do cluster. Este parâmetro estabelece o nível de utilização considerado crítico para o cluster da classe "i". O cálculo deste parâmetro é discutido no próximo tópico;
r_{kij} – A carga (média do "coeficiente de reatividade") do servidor Web "j" dentro do cluster da classe "i", medida durante o k-ésimo período de tempo. Onde ($1 < i <= N_c$) e ($1 <= j <= N_{si}$);

r_{ki} – A carga média dos servidores registrados no domínio da classe "i" medida durante o K-ésimo período de tempo. Ou seja:

$$r_{ki} = \left(\frac{\sum_{j=1}^{Nsi} r_{kij}}{N_{si}} \right) \quad (1)$$

ρ_{kij} – A carga estimada para o período de tempo (k+1) do servidor Web "j", pertencente ao cluster da classe "i", calculada durante o K-ésimo período de tempo a partir da função de controle de admissão f_{ac} após o intervalo k e antes que o intervalo (k+1) inicie, ou seja $\rho_{kij} = f_{ac}(k+1)$.

A função de controle de admissão $f_{ac}(k+1)$ e é dada por :

$f_{ac}(1) = R_i^{ac}$;
$f_{ac}(k+1) = (1-\alpha) * f_{ac}(k) + \alpha * r_{kij}$;

Onde α é um coeficiente entre 0 e 1, chamado coeficiente de ponderação do controle de admissão. Este coeficiente pondera o grau de responsabilidade e de estabilidade do mecanismo [Carpenter02]. Como o comportamento do tráfego de requisições Web é bastante imprevisível, nos experimentos apresentados na seção 5 nós optamos por um coeficiente de ponderação responsável utilizando um $\alpha = 1$;

ρ_{ki} – A média da carga predita para o período de tempo (k+1) dos servidores do cluster da classe "i", calculada durante o K-ésimo período de tempo, ou seja:

$$\rho_{ki} = \left(\frac{\sum_{j=1}^{Nsi} \rho_{kij}}{N_{si}} \right) \quad (2)$$

M_{ki} – O modo de trabalho do cluster da classe "i" durante o período de tempo (k+1) calculado a partir da $f_{ac\text{-}mode}$ (função de cálculo do modo de trabalho), dada por :

$$f_{ac\text{-}mode}(k) = \begin{cases} 0 \ (\text{se } \rho_{ki} <= R_{ki}^{em}) \ (\text{"compartilhado"}) \\ 1 \ (\text{se } R_{ki}^{em} < \rho_{ki} <= R_i^{ac}) \ (\text{"exclusivo"}) \\ 2 \ (\text{se } \rho_{ki} > R_i^{ac}) \ (\text{"saturado"}) \end{cases}$$

$\rho_{k\ min}$ - O valor da carga estimado do "class cluster" menos sobrecarregado durante o período de tempo (k+1). Ou seja:

$$\rho_{k\,min} = Min_{i=1}^{Nc}(\rho_{ki}) \quad (3)$$

O mecanismo funciona da seguinte forma: a cada k-ésimo período de tempo o "class cluster gateway" interroga todos os servidores do seu domínio para obter as cargas atuais (r_{kij}) e estimar as cargas do próximo período (ρ_{kij}). Em seguida, cada "class cluster" calcula seu "ρ_{ki}". Baseado nestes cálculos o modo de trabalho de cada "class cluster" durante o próximo período de tempo (k+1) é determinado (Figura 4).

Quando uma requisição chega durante um k-ésimo intervalo de tempo, o mecanismo WS-DSAC é aplicado. Primeiro o "class switch" identifica qual é a classe

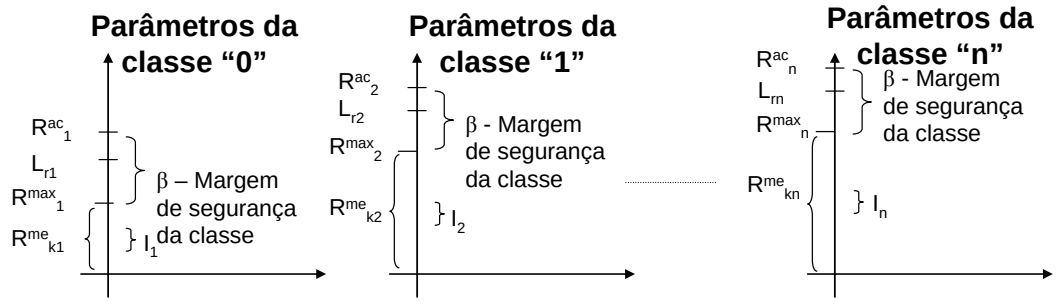


Figura 4 – Parâmetros utilizados para o cálculo do "modo de trabalho".

da requisição. Dado que a requisição pertence a classe "i" e que o cluster "m" é o menos carregado, o "class switch" executa o algoritmo a seguir:

Se ($\rho_{k\,min} <= R_i^{ac}$) e ($M_{km}= 0$) então
  Enviar a requisição para o cluster gateway "m";
Senão
  Se ($\rho_{ki} <= R_i^{ac}$) então
    Enviar a requisição para o cluster gateway da classe "i";
  Senão
    Enviar mensagem "Requisição rejeitada" para o cliente;
  Fim se
Fim se

Quando o "cluster gateway" eleito pelo algoritmo acima recebe a requisição redirecionada pelo "class switch" ele envia a mesma para o servidor menos carregado do cluster, ou seja, para o servidor que possui o menor "ρ_{kij}".

4.3 Cálculo dinâmico do R_{ki}^{em}

No mecanismo WS-DSAC, o cálculo do R_{ki}^{em} é de fundamental importância. Este parâmetro esta diretamente ligado ao uso eficaz dos recursos dentro da plataforma e a garantia dos contratos de QoS estabelecidos com cada classe. Se o seu valor é mantido em um nível muito abaixo do limite da classe nativa do cluster (R_i^{ac}), recursos que poderiam ser utilizados por outras classes dentro da plataforma ficarão indisponíveis mesmo não sendo utilizados pela classe nativa. Por outro lado, se o seu valor é mantido em um nível muito alto, sessões da classe nativa do cluster poderão ser recusadas devido ao fato de que os recursos deste cluster estão sendo utilizados por sessões de outras classes de serviços.

O R_{ki}^{em} é calculado dinamicamente a cada período Tk. Um esquema similar ao TCP slow-start, utilizado na presença de congestionamentos, é uma boa solução para nosso problema.

Os seguintes parâmetros são utilizados para o cálculo de R_{ki}^{em}:

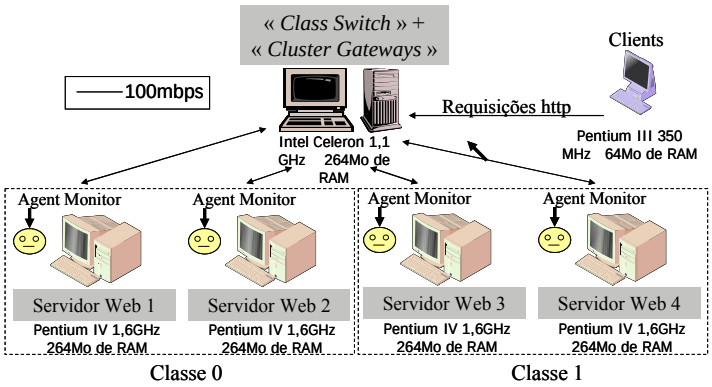


Figura 5 – Arquitetura de hardware utilizada nos experimentos.

R_i^{max} – O valor máximo que pode atingir R_{ki}^{em}. Calculado por $R_i^{max} = R_i^{ac} * (1 - \beta)$, onde β corresponde ao coeficiente de segurança da classe e $(0 < \beta <= 1)$. A escolha de um valor de β muito próximo a "0" pode comprometer o contrato de QoS estabelecido com a classe (R_i^{ac}) enquanto que valores próximos a "1" pode provocar uma subutilização dos recursos;

I_i – o incremento aplicado a R_{ki}^{em}, a cada período k, enquanto o valor de $R_{ki}^{em} <= R_i^{max}$;

L_{ri} – o valor limite para ρ_{ki} a partir do qual R_{ki}^{em} é reduzido para o valor "0";

No mecanismo, o cálculo do R_{ki}^{em} é refeito a cada período k. O valor de R_{ki}^{em} inicia em "0" e é incrementado, a cada período do valor Ii enquanto seu valor é menor que R_i^{max}. Quando o valor de ρki atinge Lri então R_{ki}^{em} passa novamente ao valor "0". Se ρki é menor ou igual a R_i^{ac} mas é maior que R_{ki}^{em} então requisições de outras classes estão começando a ameaçar a garantia de QoS contratada para a classe "i". Para evitar que sessões da classe nativa sejam recusadas porque recursos do cluster estão sendo utilizados por outras classes, somente sessões da classe "i" serão aceitas pelo cluster neste período, ou seja, o cluster vai trabalhar no "modo exclusivo" durante este período. Caso contrário, se $\rho_{ki} \leq R_{ki}^{em}$, o cluster funciona no "modo compartilhado" aceitando requisições de todos os tipos de classes.

5. Experimentos

Visando avaliar a efetividade do mecanismo vários experimentos foram realizados em um ambiente real de teste. Nesta seção, são apresentados dois destes experimentos que ilustram de maneira simples a capacidade do mecanismo em realizar os objetivos propostos. Na Figura 5, é apresentada a arquitetura de hardware utilizada para a realização dos experimentos.

5.1 Arquitetura de Hardware e Software

Um cenário heterogêneo foi utilizado (sistemas operacionais e configurações de hardware deferentes). Os equipamentos utilizados para conduzir os experimentos foram:

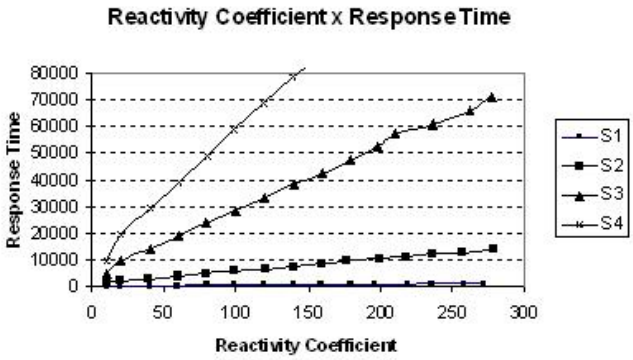


Figura 6 – Correlação "coeficiente de reatividade" e "tempo de resposta" dos serviços.

- Servidor Web 1, 2, 3 e 4, PCs genéricos contendo um processador Intel Pentium IV 1,6GHz, 264Mo de RAM, e uma placa de rede Ethernet de 100Mbs. O sistema operacional utilizado foi o Windows 2000 (Service Pack 4);
- Como "Class Switch" e "Cluster Gateway", um notebook com um processador Intel Celeron 1,1GHz, 264Mo de RAM, e uma placa de rede Ethernet de 100Mbs. O sistema operacional utilizado foi o Windows XP;
- Como cliente um PC genérico, com um processador Intel Pentium III 350 MHz, 64Mo of RAM, uma placa de rede Ethernet de 100Mbs e Windows 98.

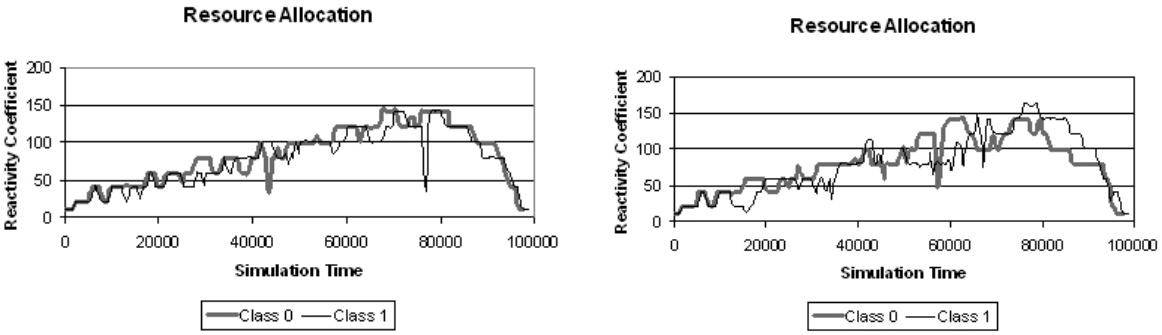
Um MA foi instalado em cada servidor web. O Tomcat 5 foi utilizado como "container" para os serviços WEB.

5.2 Coeficiente de Reatividade, Carga e Tempo de Resposta

No mecanismo WS-DSAC um "Coeficiente de Reatividade" é utilizado como parâmetro de medida para a QoS. Em [Serra04b] podemos encontrar experimentos que mostram o comportamento do coeficiente de reatividade (CR) em função da carga imposta ao servidor. Nesta subseção são apresentados resultados de experimentos que mostram a forte correlação entre o "Coeficiente de Reatividade" e o "Tempo de resposta" de uma requisição.

Para ilustrar a variação do tempo de resposta do cliente em função da carga estimada pelo CR foram realizados os experimentos a seguir. Quatro diferentes serviços Web (S1, S2, S3, S4) foram instalados em um servidor Pentium IV, 1.6GHz, 256 Mb de RAM utilizando o Microsoft Windows 2000 (Service Pack 4) e o Tomcat 5 como "container". Em uma situação de carga ideal (servidor funcionando com a carga mínima) os serviço S1, S2, S3 e S4 possuem os respectivos tempos médios de resposta: 70ms, 938ms, 4685ms, 9659ms. Para a infra-estrutura utilizada nos experimentos, o *overhead* introduzido pelo mecanismo no tempo de resposta dos serviços foi desprezível (menor que 1ms).

Um cliente foi instalado em um equipamento Pentium IV, 1.6GHz, 256 Mb de RAM utilizando o Microsoft Windows 2000 (Service Pack 4), interligado através de uma rede local de 100 MBits. A carga do servidor Web foi incrementada em 15 níveis diferentes. Para cada tipo de serviço (S1, S2, S3, S4), 100 requisições foram enviadas



(a) (b)

Figura 7 – Alocação de recursos entre as classes em períodos de baixa carga.

pelo cliente ao servidor Web em cada momento de nível de carga diferente e o tempo médio de cada tipo de serviço em cada nível de carga foi observado (Figura 6).

De acordo com os resultados obtidos, o coeficiente de correlação entre o Coeficiente de Reatividade e o Tempo de Resposta dos serviços S1, S2, S3 e S4 foram respectivamente 0,99425023, 0,99940993, 0,99903334 e 0,99945486, o que mostra uma correlação muito forte entre as duas variáveis.

5.3 Equidade em momentos de baixa carga

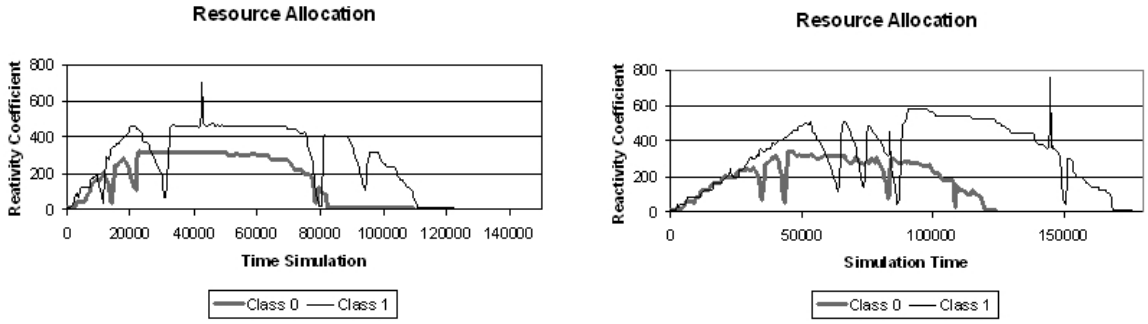
Um dos principais objetivos do mecanismo é promover uma igualdade na utilização dos recursos ("fairness") enquanto estes não estão em níveis críticos de utilização. Para mostrar esta propriedade do mecanismo os experimentos descritos a seguir foram realizados.

Dois domínios de classes diferentes foram definidos na plataforma (classes 0 e 1) cujos parâmetros estão descritos na tabela 1. Um serviço, que em uma situação de carga ideal tem um tempo de resposta médio de 4685ms, foi replicado em todos os servidores do cluster. Em um primeiro momento um cliente envia 40 requisições (uma a cada 2s) pertencentes a classe 0 (Figura 7a). E em um segundo momento o mesmo cliente envia 40 requisições (uma a cada 2s) pertencentes a classe 1 (Figura 7b).

	CLASSE 0	**Classe 1**
$\mathbf{R_{ac}}$	300ms	600ms
$\mathbf{R_{max}}$	210ms	420ms
Lr	210ms	420ms
I	10ms	10ms

Tabela 1 – Classes de Serviços

Cada uma das Figuras (7a e 7b) mostra o comportamento da carga de ambos os clusters (0 e 1). Podemos observar que em ambos os casos o mecanismo permitiu que os recursos fossem compartilhados de forma igual entre os diferentes tipos de requisição, ou seja, no primeiro momento (Figura 7a) tanto o cluster responsável pela classe 0 como o responsável pela classe 1 processaram requisições pertencentes a classe 0 e, no segundo momento (Figura 7b), ambos atenderam requisições da classe 1. Isto foi possível porque a carga dos clusters de ambas as classes não atingiram valores críticos (próximos aos limites estabelecidos) permitindo o compartilhamento dos recursos.



(a) (b)

Figura 8 – Alocação de recursos em momentos de nível elevado de carga.

5.4 Garantia de Diferenciação de Serviços em Momentos Críticos

Para ilustrar a capacidade do mecanismo em garantir a QoS contratada para cada classe de serviços, mesmo em momentos de utilização crítica dos recursos, os experimentos a seguir foram realizados utilizando-se a mesma configuração de hardware apresentada na seção anterior.

No primeiro experimento (Figura 8a), um cliente envia 20 requisições (uma a cada 1s) pertencentes a classe "0" ao mesmo tempo que um outro cliente envia 20 requisições (uma a cada 1s) pertencentes a classe "1". No segundo (Figura 8b), um cliente envia 40 requisições (uma a cada 1s) pertencentes a classe "0" ao mesmo tempo que um outro cliente envia 40 requisições (uma a cada 1s) pertencentes a classe "1".

Podemos observar que em ambos os casos o mecanismo foi capaz de manter a carga de cada "class cluster" nos níveis especificados para cada classe (Tabela 1). Foi também observado que, no primeiro caso, o tempo médio de resposta das requisições pertencentes a classe 0 foi de 71334ms enquanto que o tempo médio de resposta das requisições pertencentes a classe "1" foi de 82284ms. No segundo caso, 70619ms para classe "0" e 90790ms para classe "1". A diferença entre os tempos de resposta das classes no primeiro caso foi menos acentuada dado ao fato que a carga máxima estabelecida para o cluster da classe "1" não foi atingida.

6. Conclusões

Este artigo apresentou um mecanismo de controle de admissão e balanceamento de cargas concebido para permitir a diferenciação de serviços em clusters de servidores Web. O mecanismo tem como objetivo principal a garantia de contratos de QoS estabelecidos para diferentes classes de serviços utilizando de forma eficaz os recursos de processamento disponíveis. Para serem utilizados de forma eficaz, os recursos alocados para cada classe de serviços podem se encontrar em três estados distintos: "compartilhados", "exclusivos" ou "saturados". Quando os recursos alocados a uma determinada classe estão no estado "compartilhados", significa que os parâmetros de QoS estabelecidos para esta classe estão longe de serem atingidos e que por isto os recursos podem ser utilizados por requisições pertencentes a outras classes. Quando o nível de carga se aproxima dos parâmetros estabelecidos, os recursos passam ao estado

"exclusivos", ou seja, só podem ser utilizados por serviços pertencentes à classe proprietária dos recursos. Quando os níveis de carga ultrapassam os limiares estabelecidos para a classe, os recursos passam ao modo "saturados". Para avaliar a eficácia do mecanismo o mesmo foi implementado e avaliado em um ambiente real de testes. Experimentos mostram que o mecanismo é capaz de distribuir de forma eqüitativa os recursos em momentos de baixa utilização ("fairness") e de assegurar os contratos de QoS estabelecidos com cada classe de serviços em momentos de sobrecarga.

Como trabalho futuro pretende-se introduzir o mecanismo em um simulador para avaliar escalabilidade, desempenho e o impacto do seu uso no tempo de resposta do usuário final.

Argumenta-se que o mecanismo apresentado é uma boa solução para o aumento da QoS de aplicações disponibilizadas na Internet bem como para a diferenciação de usuários de acordo com a necessidade e a capacidade de pagamento de cada um.

Referências

[Abdelzaher02] T.F. Abdelzaher, K.G. Shin, N. Bhatti, "Performance guarantees for Web server end-systems: a control-theoretical approach"; IEEE Transactions on Parallel and Distributed Systems, Volume: 13 , Issue: 1 , Pages:80 – 96; Jan. 2002.

[Carpenter02] B.E. Carpenter, K. Nichols, "Differentiated services in the Internet", Proceedings of the IEEE , Vol. 90, Issue: 9 , Sept. 2002, Pages:1479 – 1494, 2002.

[Cherkasova02] L. Cherkasova, P. Phaal, "Session-based admission control: a mechanism for peak load management of commercial Web sites";, IEEE Transactions on Computers, Volume: 51 , Issue: 6 , Pages:669 – 685, June 2002.

[Gan00] X. Gan and All, "LSMAC and LSNAT: Two Approaches for Cluster-based Scalable Web Servers", IEEE International Conference on Communications, Volume: 2, 18-22 June 2000 Pages:1164 - 1168 vol.2, 2000.

[Hu03] Ningning Hu; Steenkiste, P.; "Improving TCP startup performance using active measurements: algorithm and evaluation", Proceedings 11th IEEE International Conference on Network Protocols, 2003; 4-7 nov. 2003, Pag. 107-118, 2003.

[Olejnik02] R. Olejnik, A. Bouchi and B. Toursel, "An Object observation for a Java Adaptative Distributed Application platform," IEEE PARELEC'02, 22-25 Sept. 2002 pp. 171-176, 2002.

[Serra04a] A. Serra, D. Gaïti, G. Barroso, J. Boudy, "A load-balancing Distributed Platform based on Differentiated Services for a Telecare Application", IEEE International Conference on Control Aplications, 2004.

[Serra04b] Serra A., Gaïti D., Barroso G., Ramos R., Boudy J.; "Uma Plataforma Distribuída com Balanceamento de Cargas para Servidores Web Baseada na Diferenciação de Serviços", Proceedings of the SEMISH-SBC, pp. 93-105, 2004.

[Wei03] Yaya Wei, Chuang Lin, Fengyuan Ren, E. Dutkiewicz, R. Raad, "Session based differentiated quality of service admission control for Web servers", International Conference on Computer Networks and Mobile Computing - ICCNMC 2003, 20-23 Oct. 2003, Pages:112 – 116, 2003.