Análise de Desempenho do Mecanismo de Balanceamento de Cargas em *Clusters* de Servidores *Web*

Isabel C. R. Oliveira[1], **Francisca A. P. Pinto**[1],
Carla K. M. Marques[2], **Giovanni C. Barroso**[1]

[1]Departamento de Engenharia de Teleinformática
Universidade Federal do Ceará (UFC)
Caixa Postal 6007 – 60455-760 – Fortaleza – CE – Brasil

[2]Departamento de Informática
Universidade Estadual do R. G do Norte (UERN)
CEP 59.610-210 – Mossoró – RN – Brasil

{isabelregio,aparecidapradop,carla.katarina}@gmail.com
{gcb}@fisica.ufc.br

Abstract. *The number of users of services offered on the Web is increasing, causing overloading the servers, reducing the quality of service (QoS). To deal with these problems, solutions-based clusters are used. This article presents the modeling and analysis of a mechanism for admission control and load balancing in a Web cluster system using colored Petri nets. Analysis were applied to some performance metrics. The results indicate that the platform was efficient, achieving the proposed objectives.*

Resumo. *O número de usuários de serviços ofertados na Web vem aumentando, causando sobrecarga nos servidores, reduzindo a Qualidade de Serviço (QoS). Para lidar com esses problemas, soluções baseadas em clusters são utilizadas. Este artigo apresenta a modelagem e a análise de um mecanismo de controle de admissão e balanceamento de cargas em um sistema de clusters Web, utilizando redes de Petri coloridas. Para análise foram aplicadas algumas métricas de desempenho. Os resultados indicam que a plataforma se mostrou eficiente, atingindo os objetivos propostos.*

palavra-chave: *Balanceamento de cargas*, *Clusters* e *Redes de Petri coloridas*.

1. Introdução

Atualmente, os serviços *Web* têm atraído grande atenção para pesquisas. Em algumas situações, o aumento na demanda desses serviços resulta numa queda inaceitável na disponibilidade do sistema. Dentre outras, uma solução atraente para contornar esse problema é a utilização de *clusters* [Tanenbaum and Steen 2009] *Web* que ofereçam maior poder de processamento para manter a disponibilidade do sistema em limites aceitáveis, mesmo com o aumento significativo de carga de serviços.

A partir da proposta de [Serra et al. 2005], o foco deste trabalho é a modelagem de um sistema que consiste em dois *clusters* homogêneos, onde é aplicado um mecanismo de balanceamento de cargas baseado em diferenciação de serviços. Num sistema

de gerenciamento de recursos, um balanceamento de carga eficiente pode aumentar de 10 para 35% o desempenho de um *cluster* [Clusterbuilder 2011]. Para análise do sistema, realizam-se várias simulações com o intuito de avaliar a capacidade do mecanismo em cumprir os seus objetivos. A modelagem foi realizada em redes de Petri Coloridas (RPC) [Jensen and Kristensen 2009], utilizando a ferramenta CPNTools, onde são investigadas algumas de suas propriedades comportamentais, como também a sua escalabilidade.

As principais contribuições deste trabalho são apresentadas a seguir. A modelagem do sistema em RPC. Em seguida, a partir dos resultados das simulações feitas, é apresentada uma análise de desempenho do sistema, utilizando as seguintes métricas de desempenho: tempo de resposta, quantidade de requisições atendidas e a taxa de utilização dos *clusters*.

O artigo está organizado da seguinte forma: uma descrição dos trabalhos relacionados é apresentada na na Seção 2. Na Seção 3 são mostrados aspectos relevantes da plataforma WS-DSAC, sobre a qual é feita a modelagem. A Especificação formal em RPC do sistema é apresentada na Seção 4. Na Seção 5 são apresentadas as métricas usadas para avaliação do sistema; na Seção 6 os resultados das simulações e análise de desempenho do sistema são apresentados, e finalmente na Seção 7 são apresentadas as conclusões e perspectivas futuras.

2. Trabalhos Relacionados

Os *clusters* de servidores *Web* e aspectos relacionados (por exemplo balanceamento de carga) tem sido objeto de muitos estudos em computação. A seguir são apresentados alguns trabalhos relacionados ao longo dos últimos anos.

Em [Papazachos and Karatza 2011] é proposta uma análise das políticas de escalonamento de *gang* AFCFS e LGFS (sem migração) e AFCFSwM e LGFSwM (com migração) em um ambiente de *clusters* multi-core homogêneos e heterogêneos. Os resultados das análises apresentaram que os algoritmos (AFCFSwM e LGFSwM) adaptados com a migração em ambiente heterogêneo mostraram menor tempo médio de resposta. Em [Gao et al. 2010] é proposta uma arquitetura de balanceamento de cargas para *clusters Web* suportando diferenciação de serviços. Porém na estratégia adotada no trabalho, o conceito de classes nativas, que envolvem prioridades em ambiente de saturação, como é o caso deste trabalho, não é usado. Em [Marques et al. 2011] apresenta uma solução para aumentar a disponibilidade de serviços em *clusters Web* baseada em sistemas multiagentes. São apresentadas a arquitetura, a especificação formal em redes de Petri e resultados experimentais da implementação da solução.

3. Plataforma WS-DSAC

O WS-DSAC [Serra et al. 2005] é uma plataforma de controle de admissão e de balanceamento de cargas concebida para *clusters* em Web, que se propõe a fazer a realocação de recursos, com a possibilidade de diferenciação de classes de serviços. Seus objetivos principais são: 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.

A plataforma WS-DSAC (Figura 1) é composta pelos seguintes elementos básicos: *Class Switch* (CS), *Cluster Gateways* (CG) e Servidores *Web*. O CS é res-

ponsável pela classificação e pelo controle de admissão de novas requisições de clientes. Ele recebe requisições HTTP, identifica a classe do serviço e, utilizando a plataforma WS-DSAC, envia cada requisição para um CG específico, o qual escolhe o servidor *Web* menos carregado para processar a requisição enviada pelo CS. A plataforma oferece diferentes níveis de QoS, um nível diferente para cada classe de serviço. Esses serviços são instalados em um certo número de servidores *Web* e podem ser compostos por serviços *Web* e objetos distribuídos. Requisições que chegam podem pertencer a diferentes classes de serviço. O administrador da plataforma associa a cada classe de serviço uma carga máxima que pode ser atingida pelo seu CG. Em cada servidor *Web* é usado o parâmetro Coeficiente Reatividade (RC) (diretamente relacionado ao tempo de resposta ao cliente), que mensura a carga atual do servidor. Esta técnica mede o tempo de espera de uma tarefa para uso dos recursos da CPU, sendo este tempo é diretamente proporcional à carga do servidor.

Em cada CG (Figura 1), o componente *Cluster Resource Manager* (CRM) periodicamente solicita informações da carga de cada servidor *Web* pertencente ao domínio do *cluster*. Ele estima a carga do *cluster* baseado nesta informação. O *Request Gateway* (RQ) distribui cada requisição HTTP que chega para um servidor *Web* específico baseado nas informações de carga do CRM. Dentro do *cluster* de uma classe de serviços específica, as mensagens trocadas entre os objetos distribuídos são também redirecionadas pelo componente *Message Object Gateway* (MOG), que utiliza a mesma informação de carga fornecida pelo CRM para manter o balanceamento de cargas dentro do domínio da classe. No CS o componente *Global Resource Manager* (GRM) solicita e mantém informações de carga de cada *cluster*. Quando uma requisição HTTP é recebida, o CS identifica a classe da requisição e a redireciona para o *cluster* apropriado. Caso a QoS especificada para a classe não possa ser fornecida, ele devolve uma mensagem recusando a requisição ao cliente.

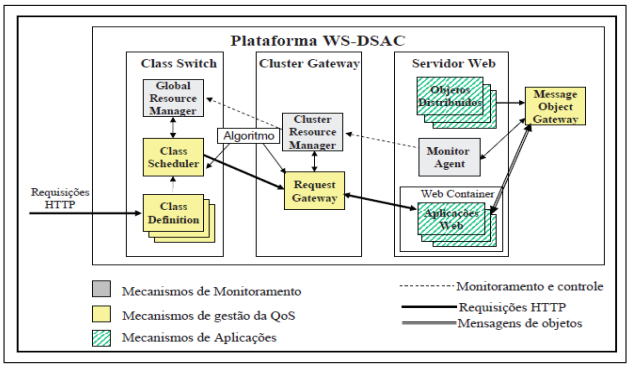

Figura 1. Componentes da plataforma WS-DSAC

A realocação de recursos entre as classes de serviços se baseia em uma mudança no modo de funcionamento de cada CG. Em um determinado intervalo de tempo, cada *cluster* pode estar em um de três modos possíveis: **compartilhado**, **exclusivo** ou **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 sua classe nativa (a classe de serviço para o qual foi designado o *cluster*) podendo atender requisições de diferentes tipos de classes. Quando o cluster passa ao modo **exclusivo**, ele só aceita requisições da classe nativa, significando que os níveis de carga chegaram a um patamar onde, aceitar requisições de outras classes,

pode implicar na rejeição de sessões da classe nativa. Quando o *cluster* passa ao modo **saturado** ele não aceita nenhuma nova requisição, ou seja, 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 dinâmicos do *cluster* (recalculados a cada intervalo de tempo), R_{emk} e ρ_k, e dois estáticos que são R_{max} e R_{ac}, onde: R_{emk} é o valor limite do RC que pode ser alcançado pelo cluster de uma classe, e a partir deste valor, o *cluster* passa a trabalhar em modo **exclusivo**, enquanto o RC está abaixo deste valor ele trabalha em modo **compartilhado**; ρ_k representa a carga estimada do *cluster* para o período de tempo seguinte e determina uma margem de segurança para que os contratos de QoS estabelecidos com a classe nativa sejam respeitados; R_{max} contém o valor máximo que pode atingir o R_{emk}; R_{ac} é o limite estabelecido para o *cluster* trabalhar no modo **exclusivo**.

A plataforma funciona da seguinte forma: quando uma requisição chega à plataforma o CS identifica qual é a sua classe. Dado que a requisição pertence à classe *i* e que o *cluster m* é o menos carregado, o CS executa o algoritmo a seguir: se o valor de ρ_{ki} for menor ou igual ao valor de R_{max} o *cluster* trabalha no modo **compartilhado**; se o valor de ρ_{ki} alcançar o valor de R_{max}, o *cluster* modifica o seu modo de trabalho para **exclusivo**; se o valor de ρ_{ki} alcançar o valor de R_{ac}, o *cluster* modifica o seu modo de trabalho para **saturado** e passa a não mais atender a nenhuma requisição. Na seção seguinte será apresentada a modelagem do sistema em RPC.

4. Modelagem de *clusters Web* com o WS-DSAC

Nesta seção é apresentada a modelagem do sistema descrito na Seção 3 e ilustrada na Figura 1. A modelagem foi feita em redes de Petri Coloridas [Jensen and Kristensen 2009], utilizando-se a ferramenta CPNTools. Na Figura 2 é ilustrada a página hierárquica do modelo em RPC, que é composta por páginas e subpáginas, e neste caso totalizando nove, e que dá uma visão em alto nível da plataforma. A página Principal representa o funcio-

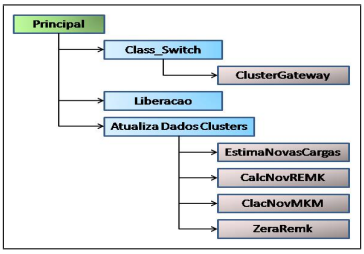

Figura 2. Hierarquia de Páginas da Modelagem em RPC

namento geral da plataforma e é através dela que são executadas as outras subpáginas (transições de substituição): *Class_Switch*, *Liberacao* e *Atualiza Dados Clusters*. A subpágina *Class_Switch* modela o controle de admissão da plataforma. Ela é composta pela subpágina *ClusterGateway*, que por sua vez modela o balanceamento de cargas e a diferenciação de serviços realizados pelo WS-DSAC. A subpágina *Liberacao* modela a liberação de requisições e cargas dos servidores, após o processamento das requisições.

A subpágina *AtualizaDadosClusters* tem a função de atualizar os dados usados pela plataforma. Esta página é composta pelas subpáginas: *EstimaNovasCargas*, *CalcNovREMK*, *CalcNovMKM* e *ZeraRemk*. As funções que cada uma delas exerce são: *Es-*

timaNovasCargas é a subpágina responsável pelo cálculo das novas cargas médias dos *clusters*; Em *CalcNovREMK* é recalculado o novo R_{emk} (Seção 3); A subpágina *CalcNovMKM* recalcula o modo de trabalho de um *cluster* (Seção 3); A subpágina *ZeraRemk* tem a função de zerar o parâmetro R_{emk}, se necessário. A primeira página do modelo (Principal) é apresentada na Figura 3.

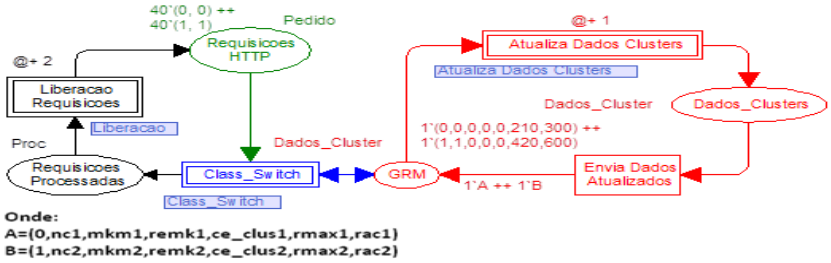

Figura 3. Página Principal da Modelagem em RPC

A rede da Figura 3 modela as seguintes atividades: existindo requisições HTTP a serem processadas, (fichas no lugar *RequisicoesHTTP*), a transição de substituição *Class_Switch* é ativada (modelando a execução de requisições); após as requisições serem processadas, elas serão liberadas, como também as cargas nos servidores. Isto é feito através da ativação da transição de substituição *LiberacaoRequisicoes*. Em intervalos de tempo pré-definidos, a transição de substituição *Atualiza Dados Clusters* é ativada, atualizando os parâmetros usados pela plataforma WS-DSAC, e os devolvendo novamente para serem usados pelo sistema.

A rede da Figura 4 representa a página *Class_Switch*, e modela o funcionamento da plataforma do seguinte modo: requisições chegando ao *Class_Switch* ativarão a transição *Class Scheduler*, que é responsável pelo controle de admissão do sistema de novas requisições; essa chegada de requisições foi modelada usando-se a distribuição de Poisson, representada pela variável *qtde_ped* na guarda da transição *Class Scheduler*. A partir de informações de cargas dos *clusters* (consulta ao lugar GRM), o menos carregado é selecionado, para atender a requisição. Após um *cluster* ser selecionado ele é testado para se saber se apesar de ser o menos carregado ele pode atender aquela requisição, porque se ele estiver em modo **exclusivo** só poderá atender requisições pertencentes à sua classe nativa.

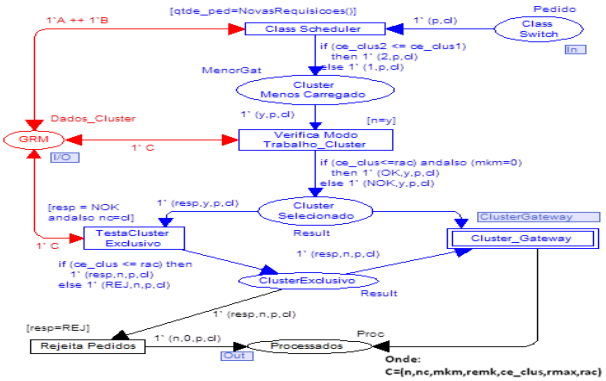

Figura 4. Página *Class_Switch*

Após o teste, uma das duas transições ocorrerá: a transição *Testa Cluster Exclusivo*, caso ele esteja nesse modo de trabalho, ou a transição de substituição *Cluster_Gateway*, caso ele esteja compartilhado. Caso o *cluster* esteja exclusivo ainda é ve-

rificado se a requisição pertence à mesma classe nativa dele e se ele ainda pode atender alguma requisição (transição *TestaClusterExclusivo*). Se esse teste é afirmativo, a requisição irá para a transição *Cluster_Gateway* que executará as alterações de carga no servidor do *cluster* em questão; caso contrário, resposta *resp=REJ*, a requisição será rejeitada.

Na Figura 5, é apresentada a página *Atualiza Dados Clusters*, que é ativada na rede da Figura 3. Nela é modelada a atualização dos parâmetros de cada *cluster* que são usados pelo WS-DSAC para fazer as escolhas mais acertadas. Esta atividade começa com os parâmetros sendo retirados do lugar *Global Resource Manager* - GRM (Figura 1) para serem atualizados, começando com o cálculo das novas cargas estimadas de cada *cluster* na transição *EstimacaoNovasCargas*, para em seguida o novo R_{emk} ser recalculado na transição *CalculoNovo_REMK*. A seguir, na transição *CalculoNovo_MKM* o novo modo de trabalho do *cluster* é determinado e por último o valor de R_{emk} é novamente testado e zerado se necessário na transição *Zera_Remk*. Após isto os parâmetros dos *clusters* retornam ao lugar *GRM* de origem através do lugar de saída *SaidaNovosParametros*, ficando à disposição do sistema novamente.

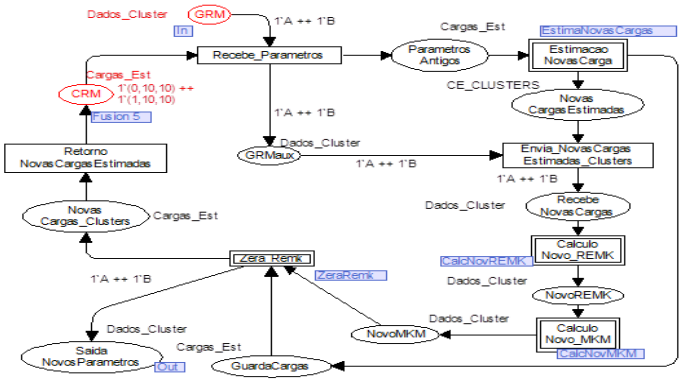

Figura 5. Página Atualiza Dados Clusters

5. Análise de Desempenho

Neste trabalho, foram aplicadas as métricas de desempenho, tempo de resposta em segundos, quantidade de requisições atendidas e taxa de utilização dos *clusters*, para analisar o comportamento da plataforma WS-DSAC em diferentes situações.

5.1. Métricas de Desempenho

A métrica **tempo médio de resposta** (TR_q) (em segundos), definido por [Papazachos and Karatza 2011], é dado pela equação: $TR_q = \frac{1}{m}$ **x** $\sum_{q=1}^{m} tr_q$. O TR_q é o tempo médio de resposta das requisições, tr_q representa o tempo de resposta de uma requisição (intervalo de tempo compreendido desde a chegada da requisição até a sua execução), *q* representa uma requisição e o *m* é o número total de requisições processadas pelo sistema. No entanto, foi considerado que não existe tempo de espera da requisição para entrar no sistema.

A métrica **quantidade de requisições atendidas** foi usada com o objetivo de estudar a capacidade dos *clusters*, em atender requisições em diversas situações. Os resultados são apresentados na Seção 6. E a **taxa de utilização dos *clusters*** se refere ao

comportamento de cargas dos *clusters* durante a simulação, de acordo com os limites estabelecidos. Neste trabalho, o tempo de processamento das requisições é exponencialmente distribuído com a média $1/\lambda$ (10 para requisições classe *0* e 20 para requisições classe *1*). A escolha por esta distribuição foi devido à sua principal característica que é a ausência de memória, e é muito utilizada na modelagem de tempos devido à sua alta variabilidade [Triola 2009]. Este é o nosso caso.

5.2. Parâmetros de entrada

Em cada resultado apresentou-se um valor médio derivado de dez simulações, por sua vez cada uma executada até dez mil passos, feitas em três cenários: **cenário 1** (só requisições de tipo classe 0 são enviadas ao sistema); **cenário 2** (só requisições de tipo classe 1 são enviadas ao sistema); e **cenário 3** (requisições pertencentes aos dois tipos de classes (*0* e 1) são enviadas ao sistema). Para cada cenário, foram executadas dez simulações sendo cada uma dessas, por sua vez executada até dez mil passos. Para as simulações, dois domínios diferentes de classes de requisições foram definidos na plataforma (classes *0* e *1*), cujos parâmetros com seus respectivos valores, são apresentados na Tabela 1, e estão descritos na Seção 3.

Tabela 1. *Parâmetros das classes de serviços*

	Classe *0*	Classe *1*
R_{ac}	300	600
R_{max}	210	420

Nas simulações, foi usada uma arquitetura contendo dois *clusters* (*clusters 0* e *1*), cada um possuindo dois servidores, sendo atribuída a cada *cluster* uma classe nativa de serviços, ficando o *cluster* 0 associado à classe *0* e o *cluster 1* à classe *1*. A chegada das requisições foi implementada segundo a distribuição de Poisson, rede da Figura 3, que é uma distribuição discreta de probabilidade, aplicável a ocorrências de evento em um intervalo especificado [Triola 2009]. No caso deste trabalho, ela se adéqua convenientemente, já que existem requisições que chegam ao sistema em um determinado intervalo de tempo de simulação. Os valores das taxas de chegada (μ) das requisições utilizadas no sistema são: 0.2, 0.4, 0.6, 0.8, 1.0.

6. Resultados da simulação e Análise

Nesta seção são apresentados os resultados das simulações executadas usando-se as métricas descritas na Seção 5.

6.1. Tempo médio de resposta (TR_q) *versus* taxas de chegada (μ)

Na Figura 6 é apresentado o comportamento dos tempos médios de resposta (em segundos) dos *clusters* em relação às taxas de chegada, nos três cenários, com intervalo de confiança de 95%. No gráfico (Figura 6), pode-se observar que para os respectivos valores de μ (0,2; 0,4; 0,6; 0,8; 1,0) os tempos médios de resposta aumentaram, para os três cenários, de acordo com o aumento das taxas de chegada, tendo os seus maiores valores quando μ = 0,8 e 1,0. Este resultado já era esperado porque nestas duas últimas taxas de chegada, o número de requisições que chegam são os maiores, saturando assim os dois *clusters*, e conseqüentemente aumentando os tempos de atendimento das requisições.

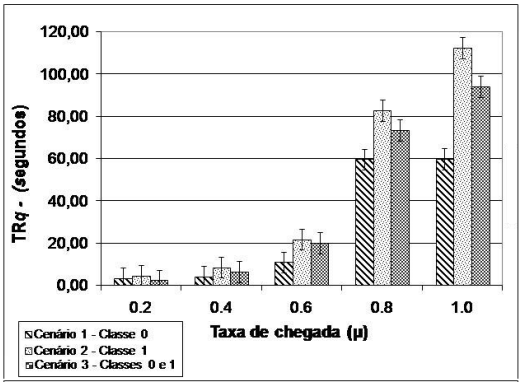

Figura 6. Tempo de Resposta *versus* Taxa de chegada (μ)

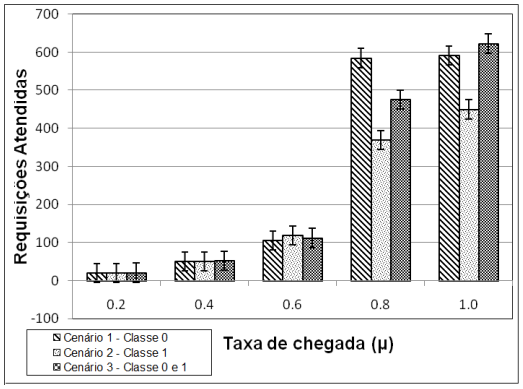

Figura 7. Requisições Atendidas *versus* Taxa de Chegada (μ)

6.2. Número médio de requisições atendidas *versus* taxas de chegada (μ)

Na Figura 7, é apresentado o comportamento das requisições atendidas em relação às taxas de chegada, com intervalo de confiança de 95%; neste caso, como no anterior, cada cenário apresentado é um valor médio derivado de dez simulações. Pode-se observar (Figura 7) que no cenário 2, para os valores de μ (0.8 e 1.0), o número de requisições atendidas é menor do que os dos outros cenários, enquanto que para a mesma situação (Figura 6) os seus tempos médios de resposta foram maiores do que os dos outros (cenário 1 e 3). Isto se deve ao fato de que como só existem requisições de classe *1* para serem atendidas, os dois *clusters* inicialmente trabalhando no modo **compartilhado**, as recebem; o *cluster 0* logo passará a trabalhar no modo **exclusivo**, pois seus limites são menores, sobrando então só o *cluster 1* para atendê-las; quando este *cluster* passar a trabalhar no modo **saturado** e o *cluster 0* ainda estiver em modo **exclusivo** nenhum dos dois *cluster* poderá receber mais nenhuma requisição, podendo ocorrer rejeições.

6.3. Taxa de utilização dos clusters *versus* tempo

Nesta seção é apresentado o comportamento das cargas dos *clusters* em diferentes situações. Na Figura 8 é ilustrada a distribuição de cargas dos dois clusters, usando o WS-DSAC, em dois momentos: no primeiro experimento, denominado Sim_1, foi simulada uma situação em que um cliente envia requisições pertencentes à classe *0*; no segundo experimento, Sim_2, o mesmo cliente envia requisições pertencentes à classe *1*.

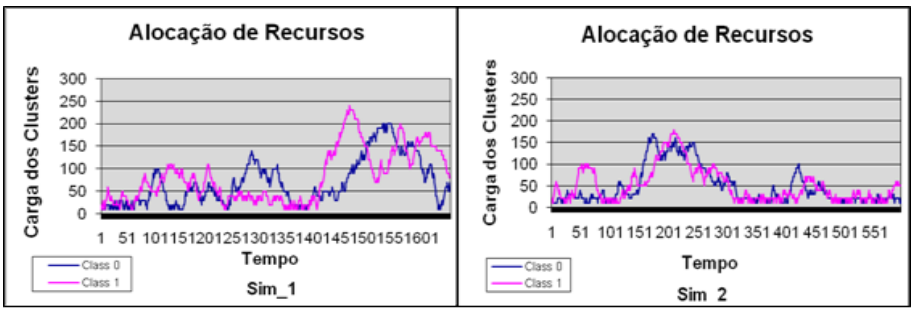

Figura 8. Alocação de recursos entre classes de serviços (baixa carga)

Vê-se, por exemplo, em Sim_1 (Figura 10), que o *cluster* 0, que possui os menores valores de R_{ac} e R_{max} (Tabela 1), não ultrapassou o seu valor de carga de 200, e a

carga do *cluster* 1 se manteve sempre abaixo de 250, ou seja, os dois *clusters* não atingiram os valores de seus R_{max}, trabalhando o tempo todo em modo **compartilhado**. O mesmo acontece nos resultados apresentados em Sim_2. Isto permite afirmar que a plataforma permitiu que recursos fossem compartilhados de forma igual entre os *clusters*, pois os dois *clusters* não trabalharam no modo **exclusivo**, em nenhum momento, ou seja, nenhum limite de parâmetros das classes foi excedido, não ocorrendo rejeições, como já era esperado.

Na Figura 9 é ilustrado o comportamento dos *clusters* em momentos de utilização crítica de carga e para isto também foram simuladas duas situações: em Sim_3, um cliente envia requisições pertencentes à classe *0* ao mesmo tempo em que outro cliente envia requisições pertencentes à classe *1*; neste caso, com o objetivo de saturar o sistema, usou-se a distribuição de Poisson com μ = 0.8. Em Sim_4 um cliente envia requisições pertencentes à classe *0*, ao mesmo tempo em que outro cliente envia também requisições pertencentes à classe *1*; neste caso, também com o objetivo de saturar o sistema mais ainda, usou-se a distribuição de Poisson com μ = 1.0.

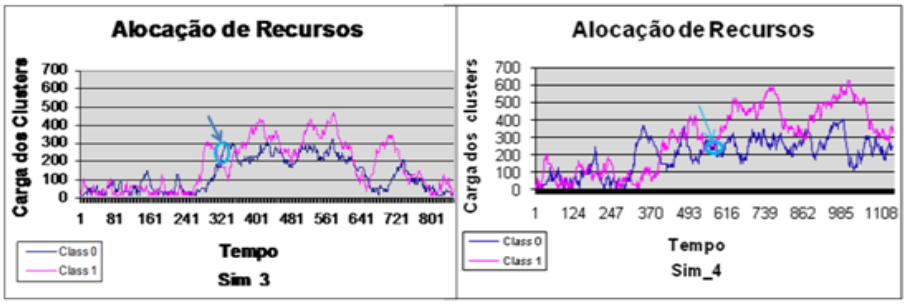

Figura 9. Alocação de recursos entre classes de serviços (alta carga)

Observa-se na simulação Sim_3 (Figura 9) que até o tempo aproximadamente igual a 321, os dois *clusters* trabalham no modo **compartilhado**; já a partir desse ponto o *cluster 0* passa a trabalhar em modo **exclusivo**, e em modo **saturado** em alguns momentos, tempos iguais a 333, 350, 430, 576, onde ocorreram rejeições. O *cluster 1*, apesar de não indicar rejeições, trabalhou em modo **exclusivo** em alguns momentos. Isto se deveu ao fato de que, neste caso, a rede ficou sobrecarregada, provocando, assim, uma diferenciação no atendimento dos serviços requisitados. Na simulação Sim_4 (Figura 9), o *cluster 0* entrou em modo **exclusivo** bem antes do que na Sim_3, tempo aproximadamente igual a 194, e entrando em modo **saturado** (carga acima de 300) em vários momentos, onde ocorreram várias rejeições. O *cluster 1* trabalhou em modo **exclusivo**, por muito mais tempo, que nas simulações anteriores, e chegou a trabalhar no modo **saturado** (tempo aproximadamente igual a 985), onde também rejeitou requisições. Foi verificado também que a diferenciação de serviços ocorreu a partir do tempo = 591 e, a partir deste instante, os níveis de carga dos dois *clusters* se mantiveram bem mais elevados, que os da simulação anterior, ocorrendo várias rejeições, como já era esperado. A taxa de rejeição observada para este caso foi de 3,93%.

7. Conclusões e Trabalhos futuros

Este trabalho apresentou a modelagem e análise de desempenho da plataforma WS-DSAC, um serviço de controle de admissão e balanceamento de carga em *clusters Web*,

propondo-se a fazer realocação de recursos, com a possibilidade de diferenciação de classes de serviços. Com o objetivo de analisar as propriedades dessa plataforma e fazer uma avaliação da sua capacidade em cumprir os objetivos propostos, foi realizada a modelagem da plataforma em RPC usando-se a ferramenta CPN *Tools*.

Experimentos foram realizados em diferentes cenários a partir de simulações, e na análise dos resultados obtidos foram usadas às métricas: tempo médio de resposta, quantidade de requisições atendidas e taxa de utilização dos *clusters*. Os resultados indicam que o WS-DSAC é uma solução viável para o aumento de QoS em aplicações na *Internet*. Observou-se, entretanto, que os tempos médios de resposta aumentaram com o aumento das taxas de chegadas de requisições, contrariamente às quantidades de requisições atendidas, indicando a ocorrência de rejeições, principalmente quando se elevou a quantidade de requisições de classe 1.

Examinando-se os comportamentos das taxas de utilização dos *clusters*, observou-se que além do balanceamento de cargas, ocorreu também a diferenciação de serviços. Isso fica bem claro quando são simulados momentos de alta carga, onde os dois clusters continuaram atendendo às requisições, mas trabalhando em modo **exclusivo**, ou seja, só atendendo requisições pertencentes à sua classe nativa. Observou-se também que em alguns momentos os dois *clusters* excederam seus limites, onde ocorreram rejeições.

Concluindo-se, verificou-se que a plataforma pode ser uma eficiente ferramenta, para ser usada em casos que devam existir prioridades de atendimento de requisições, como, por exemplo: o seu uso para atendimento prioritário de pacientes em hospitais, ou atendimento prioritário de clientes em empresas, como bancos. Como trabalhos futuros seriam implementar uma arquitetura como alternativa a tolerância a falhas, para o caso em que aconteça algum problema no *Class Switch* e, ademais utilizar um maior número de classes de serviços no WS-DSAC.

Referências

Clusterbuilder (2011). http://www.clusterbuilder.org/pages/software/clustermiddleware/resource-manager.php - acesso: dezembro de 2011.

Gao, A., Zhou, H., Hu, Y., Mu, D., and Hu, W. (2010). Proportional delay differentiation service and load balancing in web cluster systems. *in INFOCOM*.

Jensen, K. and Kristensen, L. M. (2009). *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*. Springer-Verlag, Berlin Heidelberg.

Marques, C. K. M., de Oliveira, I. C. R., Barroso, G. C., and Serra, A. B. (2011). A dynamic architecture for reconfiguration of web servers clusters. *International Journal of Computer Networks & Communications (IJCNC)*, pages 113–131.

Papazachos, Z. C. and Karatza, H. D. (2011). Gang scheduling in multi-core clusters implementing migrations. *Future Generation Computer Systems*, 27:1153–1165.

Serra, A., Gaiti, D., Barroso, G., and J.Boudy (2005). Assuring QoS differentiation and load balancing on web servers clusters. *CCA*, pages 885–890.

Tanenbaum, A. S. and Steen, M. V. (2009). *Sistemas Distribuídos*. Pearson - Prentice Hall - 2a. Edição.

Triola, M. F. (2009). *Introdução à estatística*. ABDR - 7a. Edição.