Arquitetura de Escalonamento Ortogonal de Tempo-Real para garantias de QoS em Servidores Web

Maycon L. M. Peixoto, Rogerio Tott, Michelle Nery, Francisco J. Monaco

[1]Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
Caixa Postal 668 – São Carlos – SP – Brazil

{maycon, tott, nery, monaco}@icmc.usp.br

Abstract. *Despite the significant body of results in Quality of Service (QoS) for Web-Servers, many real-world problems are not easily supported. While the current approaches limit to provide relative QoS through service differentiation, this work presents and compares three models aiming at providing absolute QoS to Web Server on a heterogeneous cluster by means of an Orthogonal Scheduling Architecture. Performance evaluation of the Orthogonal Architecture demonstrates that it performs well in providing absolute QoS in face of instantaneous changes in the workloads. Results show that the combination of EBS (Exigency Based Scheduling) as the queue discipline with the resource discipline proposed in this work outperforms the other studied.*

Resumo. *Em relação aos significativos resultados em Qualidade de Serviço (QoS) para servidores Web, existem ainda muitos problemas não resolvidos. Enquanto as abordagens atuais se limitam a prover QoS relativa através de diferenciação de serviço, este projeto apresenta e compara três modelos que têm por objetivo prover QoS absoluta para um array de servidores Web heterogêneos por meio de uma arquitetura de escalonamento ortogonal. A avaliação de desempenho da arquitetura ortogonal demonstra que a mesma obtém um bom desempenho na provisão de QoS absoluta com relação a mudanças instantâneas das cargas de trabalho no ambiente Web. Os resultados demonstram que a combinação da EBS na política de fila com a disciplina de recurso proposta neste trabalho é superior às outras combinações examinadas.*

1. Introdução

Ao longo dos últimos anos a *Web* tem agregado funcionalidades em atendimento às necessidades de um crescente número de utilizações inovadoras e tornado-se uma das mais populares e importantes aplicações da Internet. Por outro lado, tem sido submetida a mudanças substanciais na infra-estrutura de comunicação e de negócios para atender a todos esses requisitos de serviços. O incremento de pesquisas para prover qualidade de serviço em aplicações destinadas a servidores Web se dá com maior rigor nas aplicações que possuem uma relação temporal para atendimento de um serviço, por exemplo, aplicações de *e-commerce, e-banking*, *VoIP*, tele medicina, TV digital, entre outras aplicações de Internet. Nessas situações torna-se difícil para um servidor Web prover um serviço em um tempo específico dadas as condições de seu ambiente temporalmente não-determinístico [Rashid et al. 2005] [Traldi et al. 2006] [Ye et al. 2005].

Tipicamente um servidor Web recebe numerosos pedidos de serviços, não sendo capaz de gerenciar todos ao mesmo tempo. Por isso, normalmente é usado um *buffer* ou uma fila para armazenar os pedidos que chegam e ficam esperando por um serviço. Essas requisições são mantidas exatamente na ordem em que chegam ao sistema. O servidor Web recebe a primeira requisição da fila e a atende. Este é o modelo comumente utilizado, denominado escalonamento *FIFO* (primeiro a chegar primeiro a ser servido) [Ye et al. 2005]. Com essa abordagem não é possível atender a nenhuma restrição temporal para os clientes. Por exemplo, a ocorrência de tráfego de rajada tende a causar atrasos em aplicações sensíveis ao tempo, além de oferecerem um pobre desempenho em situações de sobrecarga. Por isso, a política *FIFO* não é especialmente adequada para aplicações que requerem *QoS*[1] também em nível de aplicação [Traldi et al. 2006].

No escopo das estratégias para prover *QoS* em servidores Web encontra-se a predição estimada de demanda e correspondente alocação de recursos. Porém, além de ser difícil prever com precisão a demanda, superestimar recursos para tratar curtos períodos de sobrecarga pode ser economicamente ineficiente. Uma medida alternativa para contornar esse problema é incluir mecanismos e algoritmos que gerenciam os recursos do servidor para implementar prioridades no serviço prestado.

Para buscar confiabilidade nesse tipo de ambiente imprevisível, diversos algoritmos de escalonamento adaptativos têm sido desenvolvidos nos últimos anos por [Vasiliou and Lutfiyya 2000] [Zhang et al. 2002] [Abdelzaher et al. 2003] [Teixeira et al. 2005] [Traldi et al. 2006] [Barbato et al. 2006]. O modelo de melhor esforço baseado em diferenciação de serviço [Almeida et al. 1998] [Kang et al. 2003] [Lee et al. 2003] [Stankovic et al. 1998] em servidores Web prioriza os clientes dependendo de sua classe mas não provê qualquer garantia em relação ao tempo em que a requisição será atendida.

A *QoS* pode ser definida em termos relativos ou absolutos. Diferenciação de serviço geralmente está relacionada à *QoS* relativa, e é a abordagem mais utilizadas em servidores Web [Eggert and Heidemann 1999] [Franklin 1993] [Henriksson et al. 2004] [Kang et al. 2003] [Lu et al. 2001], uma vez que garante apenas que a classe mais alta[2] receberá um serviço melhor (ou no mínimo não pior) que qualquer classe mais baixa, baseando-se em algum critério para distinção dos serviços oferecidos. Por outro lado, a confiabilidade é tipicamente especificada em termos de valores absolutos, por exemplo: *na média o cliente será atendido em 10 ms* ou *nenhuma requisição do cliente atrasará mais do que 15 ms* [Vasiliou and Lutfiyya 2000].

QoS Absoluta acresce substancialmente o nível de complexidade dos mecanismos destinados a prover garantias temporais em ambientes não-determinísticos. Nesse modelo uma classe não possui maior prioridade do que outra, mas, ambas devem cumprir o contrato estipulado previamente. Implementar *QoS* absoluta dessa forma implica em considerar o servidor Web um sistema de tempo-real. Para isso, existem diversos algoritmos de escalonamento projetados para atender restrições temporais.

Dentre os algoritmos clássicos, têm-se o *EDF* (*Earliest Deadline First*) [Liu and Layland 1973], que trata primeiro as requisições que estão mais próximas de perder o seu prazo de conclusão, e possui bons resultados para determinados cenários em

[1]Quality of Service.
[2]Classe mais alta: de maior prioridade; classe mais baixa: de menor prioridade.

ambientes uniprocesados. Entretanto, utilizado de maneira isolada em um ambiente multiprocessado, degrada rapidamente o desempenho em condições de sobrecarga do sistema [Stankovic et al. 1998]. Por isso, outros algoritmos de escalonamento têm sido propostos [Bampis and Kononov 2001] [Ramamritham et al. 1990] para resolver esse problema de escalonamento em ambiente multiprocessado, o qual é caracterizado como NP-difícil [Stankovic et al. 1998] [Liebeherr et al. 1995] [Ramamritham et al. 1990]. Há assim, um vasto campo de investigação sobre abordagens de escalonamento de tempo-real para ambientes distribuídos, principalmente sobre conjunto de recursos heterogêneos.

1.1. Motivação e Objetivos

Fundada na crescente relevância dada à *Web*, no domínio dos sistemas de informação, a motivação deste trabalho está concentrada na importância de aprimoramento de técnicas de gerência e manipulação de recursos com o intuito de oferecer garantias de *QoS* e atender as exigências de novas aplicações.

No contexto do predomínio de trabalhos que abordam a garantia de *QoS* para os sistemas uniprocessados, o objetivo deste trabalho é desenvolver e avaliar uma abordagem para oferecimento de suporte à *QoS* absoluta também para o modelo de sistemas multiprocessados, contribuindo, assim, para o fornecimento de suporte de qualidade de serviço em escala maior. Este trabalho propõe um modelo ortogonal que utiliza uma disciplina de recursos original aliada à disciplina de fila *EBS*, onde o escalonador da fila está na horizontal e o escalonador de recursos na vertical, atuando assim, nessa visão, de forma ortogonal. Para avaliar as propriedades desse modelo foi simulado e comparado com os algoritmos clássicos mais amplamentes empregados.

2. Arquitetura Ortogonal

Para avaliar o desempenho dos algoritmos desenvolvidos foi utilizado o modelo de rede de filas do servidor Web distribuído, sistema de fila M/M/4, limitando-se às suas principais características. Para solucionar o problema envolvido utilizou-se a extensão funcional *SMPL* escrita na linguagem de programação C. SMPL é uma extensão funcional baseada em simulação orientada a eventos.

Assim, dois tipos de escalonamento são executados: escalonamento na fila (*job*), e escalonamento de recursos (processador); constituindo este segundo problema, uma contribuição deste trabalho em relação ao estudo desenvolvido por [Casagrande et al. 2007] e demais abordagens clássicas.

2.1. Modelo de Recursos

Neste trabalho considera-se um *array* de servidores modelado como um conjunto de processadores paralelos ou nós, *K = (K1, K2, ..., Kn)*, formado por uma arquitetura heterogênea. Foi utilizado n igual a quatro.

O modelo considera que o tempo de escalonamento (tomada de decisão e despacho) é nulo, por ser este insignificante comparado à ordem de grandeza dos outros tempos que definem a dinâmica do processo, tais como de processamento das requisições e tempo de espera em fila.

2.2. Modelo de Tarefas

O modelo inicial de tarefas assume que todas as tarefas são independentes, a execução ocorre sem restrição de precedência, sem preempção e sem recirculação. O modelo adotado é caracterizado por tarefas aperiódicas com o *deadline* calculado dinamicamente, ou seja, a cada ciclo de execução, os *jobs* são atualizados segundo os seus *deadlines* em função de outros fatores relevantes, tais como a carga imposta a cada processador do cluster. Para cada tarefa i, existe um nível de QoS N correspondente, que é cumprido de acordo com o contrato previamente estipulado.

Os atributos de cada tarefa que ocorrem no texto são descritos da seguinte forma:

- τ_i^p é o tempo de término (estimado) do *job* i no nó p;
- C_i^p é o custo (carga de trabalho ou tempo de processamento) do *job* i no nó p;
- T_r^p é o tempo restante necessário para que o nó p seja liberado;
- $T_i^p(f)$ é o tempo de processamento em relação aos atributos (D_i, F_i) do *job* i no nó p, onde, D_i é o *deadline* e F_i é o fator de carga;
- $\sum C_n^p$ representa o custo total dos jobs que aguardam na estrutura da fila virtual;
- $C_j^p(t)$ é o tempo restante de processamento do *job* j atualmente em execução no nó p;

2.3. Controle de Admissão

Foi proposto um mecanismo de controle de admissão tal como em [Teixeira et al. 2005]. Esse mecanismo é sensível a mudanças na carga de trabalho oferecida ao sistema, o qual foi implementado em todas as políticas elaboradas.

O mecanismo de controle de admissão tem como métrica de sobrecarga a utilização média do cluster de servidores Web. O cálculo é realizado através de uma média exponencialmente ponderada. O algoritmo funciona da seguinte forma: a utilização atual do cluster é medida a cada nova requisição que chega ao servidor e este valor é combinado com dados históricos a fim de obter a média exponencialmente ponderada, ME_{cor}. Caso a mesma esteja acima de um limiar pré-estabelecido (*THRESHOLD*) adotado nesse projeto igual a $0,77$, então o controle irá recusar quaisquer novas requisições, independentemente de sua classe, até que ME_{cor} caia para níveis aceitáveis, isto é, abaixo do limiar determinado. O cálculo da média exponencialmente ponderada é definido pela equação (1).

$$ME_{cor} = ((1 - p) \cdot ME_{ant}) + (p \cdot ME_{atual}) \quad (1)$$

ME_{ant} é o valor anterior da média, é o histórico. Enquanto ME_{atual} é o valor atual observado para a utilização do cálculo. O peso p tem a função de um coeficiente de sensibilidade das mudanças de carga de trabalho que chegam ao sistema, à medida que:

- $p \longrightarrow 1$: o mecanismo de controle torna-se bastante sensível a mudanças na utilização do sistema.
- $p \longrightarrow 0$: o mecanismo de controle reage mais lentamente a mudanças na utilização do sistema. Nesse projeto, adotou-se essa configuração, sendo p igual a $0,001$.

2.4. Satisfação do Cliente

Considera-se que um usuário estará satisfeito com o serviço a ele oferecido quando obtiver uma alta porcentagem de requisições atendidas em média abaixo do limiar de qualidade contratado. Logo, analiticamente, o escalonador pode tomar como índice de satisfação do usuário a equação (2), em que são relacionadas a quantidade total de requisições submetidas pelo i-ésimo usuário (R_i) e o número de vezes em que a média do tempo de resposta de sistema dessas requisições ficou abaixo do limiar contratado (N_i).

Uma requisição é bem atendida quando seu tempo médio de resposta de sistema observada (real) for menor ou igual ao tempo médio de resposta somado a um desvio padrão aceitável contratados. Quanto mais próximo de R_i for o valor de N_i, maior será a satisfação proporcionada ao usuário i. Esta é uma métrica suficiente para avaliar o desempenho de um escalonador no cumprimento dos níveis de *QoS* por ele oferecidos.

$$S_i = \frac{N_i}{R_i} \tag{2}$$

Especificou-se melhorias na métrica de Satisfação do Cliente: pode ser conveniente transformar a variável S_i de *hard* para *soft*, ou seja, ao passo que uma requisição não é atendida dentro do contrato estipulado ela é julgada segundo o critério de tempo de extrapolamento (Tex_i), (Equação 3) faixa que varia de 0 a 1, que representa a porcentagem do tempo ultrapassado com referência o contrato, isso torna eficiente o emprego de políticas com menores tempos médios de resposta, ainda que esses tempos tenham sido ultrapassados.

$$S_i = \frac{N_i + (1 - Tex_i)}{R_i} \tag{3}$$

Essa métrica também proporciona um sistema de promoção para as requisições atendidas abaixo do contrato estipulado. ($Tant_i$) representa a porcentagem do tempo médio de resposta atendido abaixo da referência do limite contratado, dado pela equação 4.

$$S_i = \frac{N_i + Tant_i}{R_i} \tag{4}$$

O mecanismo de controle de admissão pode influenciar na satisfação de um dado usuário. Se alguma requisição é descartada pelo algoritmo do controle de admissão é criada uma penalização. A cada requisição descartada é decrementado o número de vezes em que a média do tempo de resposta do sistema dessas requisições ficou abaixo do limiar contratado (N_i).

Especificou-se a penalização pelo descarte do Controle de Admissão: tornar o sistema de controle de admissão de *Hard* para *Soft*. Dado a satisfação do cliente na equação 2, pode ser conveniente incluir uma taxa de descarte de requisições, equação 5. Esta taxa (d_i) representa o peso que uma requisição descartada pelo controle de admissão influencia o i-ésimo usuário, tendo seu valor variando entre 0 e 1. Isso significa que pode ser instituído por meio de um contrato qual o peso que um usuário dá para uma requisição

descartada, colaborando com o mecanismo de negociação. Enquanto fixar $d_i = 0$ significa ausência desse tipo de controle, fixar $d_i = 1$ significa que nenhuma requisição do usuário i pode ser descartada. Conforme a taxa de descarte aumenta, a satisfação do cliente diminui, como é o esperado.

$$S_i = \frac{N_i \cdot (1 - d_i)}{R_i} \tag{5}$$

3. Fluxo da Arquitetura Ortogonal

No modelo de estudo, a fila de requisições é composta por *jobs* que contêm um par de atributos (D_i, C_i), representando o *deadline* e o fator de carga[3] do job, respectivamente.

A abordagem proposta consiste em combinar ao escalonador de *jobs* um segundo escalonador, de recursos. As duas políticas de escalonamento, assim, atuam ortogonalmente (Figura 1), a primeira ordenando a fila de *jobs* e a segunda, atribuindo o *job* ao recurso para atendimento.

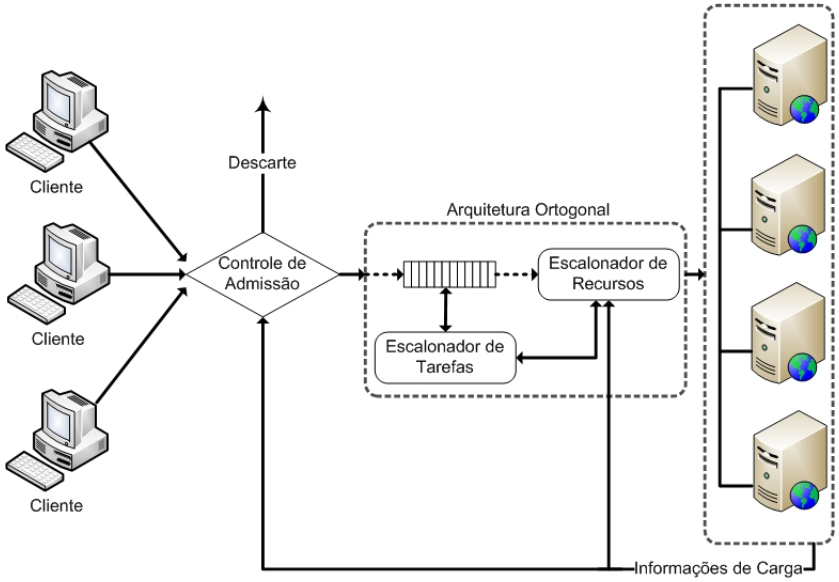


Figura 1. Modelo Ortogonal

A arquitetura de escalonamento ortogonal é descrita por meio de um escalonador que seleciona a ordem de execução de um *job* e um recurso, ou seja, para qual processador será enviado para ser executado. Esse projeto compara um conjunto de combinações de políticas de fila (EDF e EBS) e políticas de recursos MQ (*Multiple Queue*), SQ (*Single Queue*) e DSQ (*Dynamic Single Queue*). É possível assim, traçar algumas conclusões das vantagens e limitações de cada abordagem.

3.1. Política de Fila

O *escalonador de requisições* possui a estratégia inicial de manipular os tempos de fila das requisições de acordo com o parâmetro contratual de cada classe. As requisições que

[3]É uma propriedade inerente ao job associado ao seu custo computacional relativo. O custo (tempo de processamento) do job no processador p é então dado por uma função $T_p(F_i)$.

estão mais próximas ou foram descumpridas receberão maior prioridade do escalonador, ao contrário daquelas que toleram maiores tempos de fila.

Dentre os vários algoritmos clássicos, utilizou-se o *FIFO* que é um algoritmo que possui larga utilização e *overhead* de escalonamento nulo, porém possuindo nesse ambiente pobres resultados. Também foi utilizado o *SJF* (*Shortest Job First*) o qual realiza a priorização dos *jobs* mais curtos. Utilizou-se também o EDF, o qual simplesmente lida primeiro com as requisições que estão mais próximas de perder seus *deadlines* [Liu and Layland 1973]. Apesar de possuir os melhores resultados dos algoritmos clássicos, o algoritmo EDF, caso seja aplicado de forma isolada, irá exibir ainda um resultado pobre.

3.1.1. EBS - Exigency-Based Scheduling

A *EBS* [Casagrande et al. 2007], baseada em uma estratégia híbrida que leva em conta o custo de execução de uma requisição e o tempo de espera em fila, distingue-se por buscar impor ao sistema a menor demanda de recursos possível, garantindo um compromisso entre desempenho e estabilidade no atendimento de contratos individuais. Em sistemas interativos *online*, como servidores Web, a *EBS* proporciona resultados superiores às políticas convencionais.

A política atua com base na teoria de *feedback scheduling*, construindo a atualização instantânea do *deadline*, ou seja, sempre que o servidor terminar de atender uma dada requisição j do usuário u o valor de Ω_u (Tempo Médio de Resposta) será recalculado. Como mostra a equação 6, este cálculo é a média entre o antigo tempo de resposta real de u (Ω'_u) e o tempo de residência da requisição j recém atendida. O valor de $time()$ representa o tempo atual, $timeStamp_j$ o tempo de chegada da requisição j e R_u o número de requisições anteriormente submetidas por u.

$$\Omega_u = \frac{(\Omega'_u \cdot R_u) + (time() - timeStamp_j)}{R_u + 1} \tag{6}$$

À medida que as requisições chegam ao sistema e não encontram servidor disponível são transferidas para uma fila única de espera. Ao término da execução de uma requisição, o escalonador recalcula o tempo médio de resposta real daquele usuário, e em seguida busca na fila a requisição mais urgente, ou seja, aquela que tolera o menor tempo de espera dentre todas as existentes no sistema. Essa terá acesso ao servidor, e o ciclo se repete.

O *deadline* representa o quanto uma requisição ainda pode esperar na fila antes de começar a descumprir seu contrato. Para obtê-lo, isola-se na inequação 7 a variável D_j, que representa o *deadline* da requisição j. Como o interesse é obter o máximo tempo de espera aceitável pelo usuário u antes que o valor do seu tempo médio de resposta real (Ω_u) ultrapasse o seu tempo médio de resposta contratado, igualam-se os termos. Os valores R_u e T_{w_j} expressam respectivamente o número de requisições feitas pelo usuário u e o tempo de espera em fila da requisição j, até o momento.

$$\frac{(\Omega_u \cdot R_u) + T_{w_j} + D_j}{R_u + 1} \leq \Omega_{c_i} \tag{7}$$

O valor de D_j representa maior prioridade a requisição ao passo que se torna menor, pois tem maior urgência. Basicamente este é o mesmo princípio utilizado pela política de escalonamento *EDF*, porém tratada de forma mais flexível, pois visa manter valores médios de tempo de resposta de sistema, e não-preemptiva.

Em alguns casos podem existir algumas requisições urgentes com os mesmos valores de *deadline*. Apesar de apresentar a mesma urgência, as requisições desse conjunto podem impor pesos distintos ao sistema, uma vez que, além do parâmetro de serviço (requisição operacional), o custo de processamento é outro fator de impacto sobre a exigência imposta ao sistema. Minimizar esse impacto é importante, pois um sistema sob menor carga terá melhores condições para lidar com os requisitos de serviço de suas requisições.

Proporcionam-se tempos médios de espera menores do que aqueles obtidos com a *FIFO* através do uso da política de escalonamento *SJF*. Nela as requisições em espera por atendimento são organizadas em uma fila segundo o tempo de processamento (T_p). Realocar as requisições mais curtas primeiras possibilitam uma diminuição na média de tempo de resposta oferecida pelo sistema.

Na abordagem de escalonamento desenvolvida atribui-se prioridades baseado no *deadline* e no valor esperado do tempo de processamento da requisição. Como ilustra a equação (8), a prioridade de uma dada requisição j em fila, do usuário u, é dada por P_j. Quanto menor for o valor de P_j maior será a prioridade de escalonamento da requisição j, conseqüentemente as requisições que apresentarem maior urgência (menores valores de D_j) e menor custo esperado de processamento (T_{p_j}) serão classificadas como mais prioritárias.

$$\begin{aligned} P_j &= D_j \cdot T_{p_j} \\ P_j &= \left(\left(\Omega_{c_u} \cdot (R_u + 1) \right) - \left(\Omega_u \cdot R_u \right) - T_{w_j} \right) \cdot T_{p_j} \end{aligned} \tag{8}$$

Não ocorre com a *EBS* o problema de *starvation* que é apresentado pela política *SJF*, pois utiliza o *deadline* como critério de priorização. Assim, mesmo requisições mais longas não serão indefinidamente preteridas com a chegada de outras mais curtas, pois quanto mais tempo elas ficarem no sistema, menor se tornarão seus *deadlines*, o que permite serem eventualmente atendidas. Existe uma contribuição para o aumento da priorização, que são o aumento do tempo em fila (T_{w_j}) das requisições.

Em alguns casos podem existir valores negativos[4] de *deadline*, como nos cenários onde o sistema está saturado por requisições mais exigentes, o resultado do escalonamento é completamente o oposto quando aplicado às requisições que estão em descumprimento de contrato. Nesses casos é realizada uma correção da abordagem para a disciplina para manter o objetivo proposto. A correção resulta na equação (9), para as requisições mais urgentes que ainda não tiveram seus *deadlines* descumpridos, a prioridade é diretamente

[4]Um valor negativo de *deadline* significa que o contrato foi violado, i.e. o tempo que uma requisição pode aguardar na fila é menor que zero.

proporcional ao seu *deadline* e ao custo esperado de processamento. Para aquelas com descumprimento de *deadline* a prioridade é inversamente proporcional a seu custo esperado de processamento, já que quanto menor for esse custo menor será o valor de P_j resultante, e assim maior será sua prioridade de escalonamento.

$$P_j = \begin{cases} D_j \cdot T_{p_j} \text{ se } D_j \geq 0 \\ D_j \cdot \frac{1}{T_{p_j}} \text{ se } D_j < 0 \end{cases} \quad (9)$$

Dessa forma garante-se que as requisições mais urgentes, independente de terem descumprido ou não seus *deadlines*, e com menores custos esperados de processamento sejam escalonadas primeiro.

3.2. Política de Recurso

A política *EBS* é capaz de atender a contratos de QoS absoluto baseado em limite superior para média de tempo de resposta, o que configura um caso de tempo-real estocástico. A partir dessa premissa foi investigado a aplicabilidade da *EBS* demonstrado por [Casagrande et al. 2007] para o caso de um *array* de servidores. Foram aplicadas também as estratégias *FIFO, SJF e EDF* os quais não são adequadas para sistemas multiprocessados [Mok 1983] [Stankovic et al. 1995].

3.2.1. Modelo Multiple Queue

O primeiro modelo construído, o modelo *Multiple Queue* ou *Web Switch*, é formado por um componente central da arquitetura que atua como um despachante, mapeando o endereço virtual para um endereço de servidor real. As requisições são encaminhadas para o módulo *switch* que, por sua vez, os envia para um servidor qualquer do *array*, de acordo com algumas regras de escalonamento (por exemplo, políticas que levam em conta as características de carga do servidor).

3.2.2. Modelo Single Queue

Um modo trivial de transpor o referido método para o caso multiprocessado consiste em organizar a fila de requisições baseando-se nos resultados obtidos com sua heurística e enviar, então, o primeiro *job* da fila para ser executado no primeiro servidor livre do *array* (Figura 2).

Nota-se, contudo, que este simples critério de enviar o *job* para o primeiro servidor livre pode não ser a melhor solução, comparado ao caso em que aguardar por um processador mais eficaz resulte em ganho sob uma perspectiva do sistema como um todo.

3.2.3. Modelo Dynamic Single Queue

Uma primeira tentativa de melhorar esse esquema utiliza o critério *tempo de término (τ)* em cada nó do cluster e seleciona o processador que possibilita a conclusão do processamento do *job* primeiro (Figura 3).

```
1. Aguarda-se até que um nó fique livre.
2. Organiza-se a fila segundo alguma política {FIFO, SJF, EDF ou EBS}.
3. Retira-se o primeiro job da fila e o envia para o primeiro nó livre do cluster.
4. Repete passo 1.
```

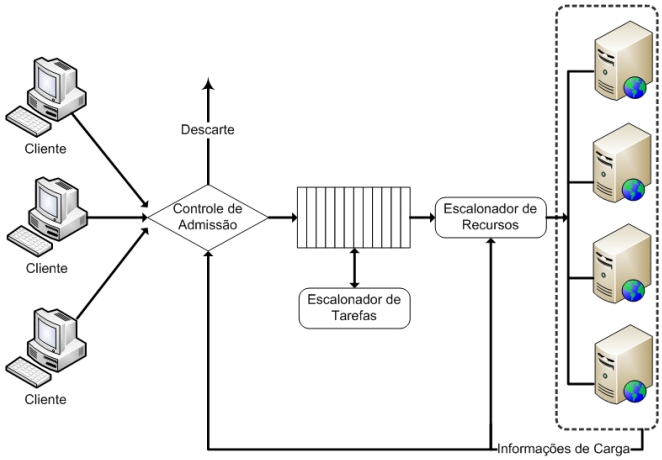


Figura 2. Modelo Single Queue.

Determina-se o tempo de término do *job* i no processador p, τ_i^p, como a soma do tempo de processamento do *job* i no processador p, C_i^p, adicionado ao tempo de espera pela liberação do processador p, T_r^p, mostrado a partir da equação 10:

$$\tau_i^p = C_i^p + T_r^p \quad (10)$$

$$C_i^p = T_i^p(f) \quad (11)$$

$$T_r^p = \sum C_n^p + C_j^p(t) \quad (12)$$

A fila virtual é apenas uma representação abstrata que serve para prover controle de gerenciamento da ordem dos *jobs* que já foram selecionados para execução, porém aguardam a vez de processamento. Não existe na realidade transposição dos elementos entre a fila principal e as fila virtuais. Acontece um gerenciamento implícito, que é a atribuição de um contador para cada servidor ($\sum C_i^p$). Esse contador armazena a carga imposta aquele servidor e a estrutura principal da fila armazena todos os índices envolvidos.

4. Planejamento de Experimentos

Foram utilizadas cargas de trabalho sintéticas com a distribuição exponencial para descrever tanto os intervalos de chegada quanto o tempo de serviço das requisições, a fim de garantir a taxa de utilização do sistema. Essa é uma distribuição amplamente utilizada para analisar sistemas de filas [MacDougall 1989]. Os comportamentos podem ser também examinados mediante as estatísticas de utilização do sistema, e de satisfação dos requisitos. Considerando que o ambiente de simulação foi configurado da seguinte maneira:

- média de taxa de chegada $\lambda = 2,35$.
- média de taxa de serviço $\mu = 4,0$.

1. Aguarda-se até que um nó fique livre.
2. Organiza-se a fila segundo alguma política {*FIFO, SJF, EDF ou EBS*}.
3. Determina-se o τ em cada nó para o primeiro *job* da fila.
4. Retira-se o *job* para ser executado no nó do cluster onde τ seja o menor.
5. Se o estado do nó for ocupado: insere-se o *job* em uma lista virtual associada ao nó e retorna-se
   ao passo 3. Se ele esta livre o *job* é executado, as filas esvaziadas e
   retorna-se ao passo 1.

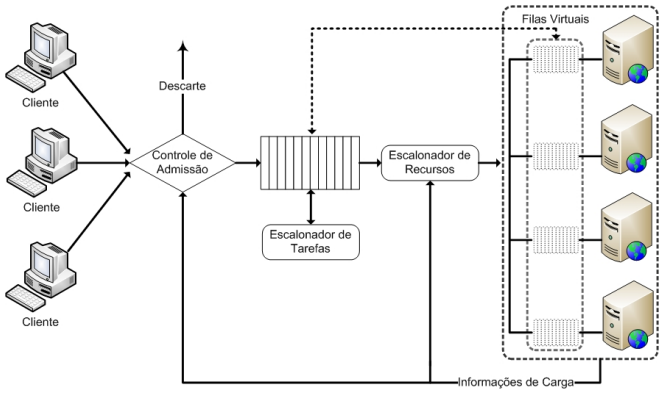


Figura 3. Modelo Dynamic Single Queue.

- cluster 4 servidores heterogêneos com potências relativas $P_1 = 1,00$, $P_2 = 0,50$, $P_3 = 0,33$ e $P_4 = 0,25$.

Obtém-se uma potência resultante do cluster de $P_R = 2,08$ e uma utilização efetiva média, segundo a equação 13, em torno de 82%.

$$\rho = \frac{\mu}{\lambda} \cdot \frac{100}{P_R} \tag{13}$$

4.1. Caracterização do Ambiente

O ambiente de simulação foi configurado com três classes de usuários {A,B e C} com diferentes variações contratuais. Os cenários utilizados neste projeto são os seguintes:

- Cenário 1: Proporção de requisições de 20%, 35% e 45% para as classes A, B e C, nessa ordem.
- Cenário 2: Proporção de requisições são iguais para todas as classes.
- Cenário 3: Proporção de requisições de 10%, 80% e 10% para as classes A, B e C, nessa ordem.
- Cenário 4: Proporção de requisições de 45%, 35% e 20% para as classes A, B e C, nessa ordem.
- Cenário 5: Proporção de requisições de 45%, 10% e 45% para as classes A, B e C, nessa ordem.

Os cenários foram assim construídos de forma a abranger a maior parte das possíveis configurações de utilizações dos usuários naquele ambiente. Uma vez definida as proporções de requisições, o valor do contrato da classe A (CtA) e da classe C (CtC) são fornecidos a partir do contrato da classe B (CtB), empiricamente medido com valor de referência de 15 ut. Durante as simulações, atribuiu-se à P os seguintes valores para as variações contratuais: 10%, 20%, 30%, 40% e 50%. Assim, segundo as equações 14 são definidos os contratos das classe A e C a serem cumpridos previamente.

$$CtA = CtB - (CtB \cdot P)$$
$$CtC = CtB + (CtB \cdot P) \quad (14)$$

A classe a qual o usuário pertence é definida através do parâmetro de tempo médio de resposta de sistema a ser garantido às requisições de um dado usuário, pois nele está incluso o tempo de fila gerenciado pelo escalonador, o que permite controlar a qualidade do atendimento. Este parâmetro, neste texto denominado tempo médio de resposta contratada pela i-ésima classe (Ω_{c_i}), é especificado previamente por um acordo entre o provedor de serviços e o cliente, e é utilizado pelo escalonador como base para atribuição de prioridades. O valor que será comparado com Ω_{c_i} é o do tempo médio de resposta real (Ω_u), que é o tempo de resposta médio de sistema atualmente oferecido para o usuário u.

4.2. Taxa de Utilização do Sistema

A Tabela 1 relaciona a utilização individual de cada servidor. Os valores são representativos de um único cenário, porém não se alteram apreciavelmente para outros cenários de carga, resultando que podem ser considerados típicos para efeito de comparação.

Tabela 1. Taxa de Utilização dos Servidos do Sistema (%)

	MQ	SQ	DSQ
Servidor	utilização	utilização	utilização
0	93,52	77,50	96,51
1	77,35	78,24	82,16
2	56,72	76,38	51,67
3	38,57	74,64	23,67

Esses valores mostram que a política *DSQ* (em combinação com a disciplina de fila *EBS*) tende a uma maior utilização dos servidores mais rápidos, atribuindo-lhe, as requisições mais impactantes (conforme o conceito de exigência anteriormente definido) no sistema. Não obstante exibindo a mesma característica, entretanto, o *MQ* perde por ter visões parciais de trechos das filas de requisições pendentes, ao passo que a *DSQ* tem condições de executar a tomada de decisão de escalonamento mediante uma visão global de todos os *jobs*. O *SQ*, por outro lado, utiliza os servidores de forma igual, favorecendo o sistema como um todo em questões de desempenho, porém não é orientada para o cumprimento das garantias contratuais.

5. Comportamento do Escalonador

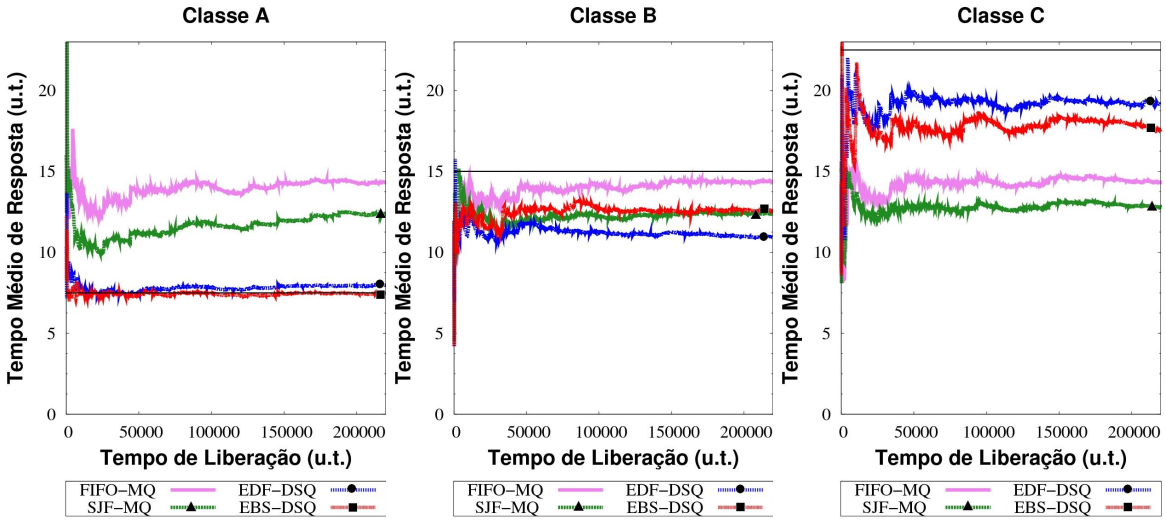
Para se ter uma ampla amostragem dos dados fixou-se como 100.000 o número de requisições submetidas em cada cenário a ser simulado, e com o intuito de alcançar confiabilidade estatística realizou-se os experimentos 15 vezes, simulando todos os cenários em cada uma das vezes. Cada simulação utilizou-se de uma semente diferente já fornecida pelo simulador *SMPL*.

Com referência a um valor médio de atendimento P de 15 unidades de tempo, empiricamente medido para um sistema sem diferenciação de serviço, o contrato da classe A é tal que as requisições a ela pertencentes devem ser atendidas em um tempo médio de resposta que deve ser inferior em $P\%$ do valor de referência; ao passo que o limite de tempo médio de resposta da classe C é $P\%$ superior ao valor de referência.

Nota-se, portanto, que este é um cenário de baixa exigência, em que a variação contratual entre as classes é baixa e a classe mais exigente (A) tem poucos usuários. Com tal carga leve, frente à potência computacional do conjunto de 4 servidores, excluindo-se a abordagem *MQ-FIFO*, todas as políticas de escalonamento restantes os quais foram simuladas atenderam os contratos para todas as classes. O resultado interessante a destacar nesse cenário é que é atendido o critério de confiabilidade (cumprimento do contrato de *QoS*).

Primeiramente, a abordagem *DSQ* foi comparada com as abordagens tradicionais, comumente chamadas de clássicas, que é a abordagem *MQ* ou também conhecida como *Switch Web* com um balanceador de carga. A Política de recurso *MQ* foi combinada às politicas de fila *FIFO e SJF*, enquanto a *DSQ* foi combinada a *EDF e EBS*.

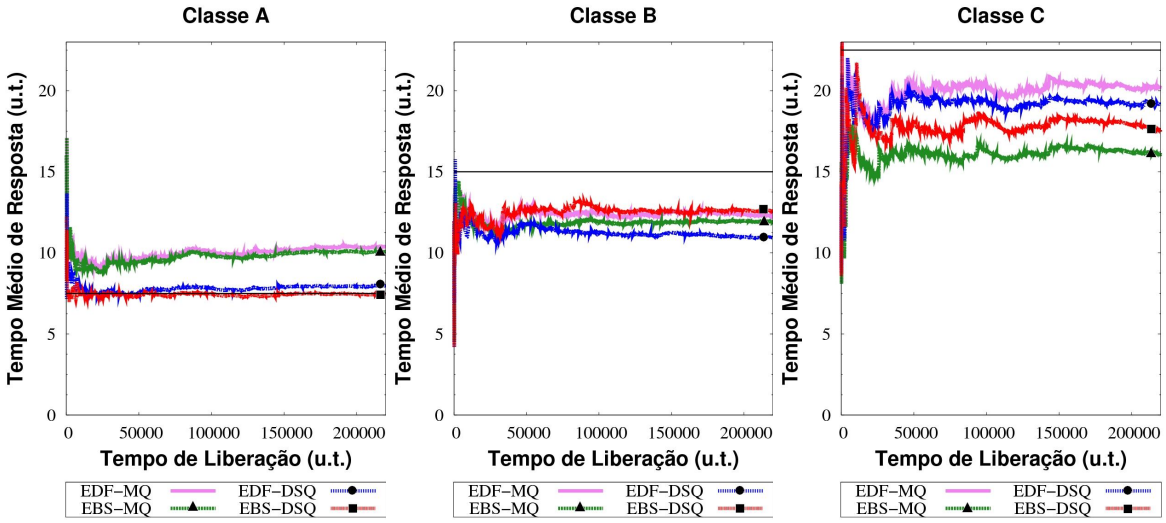
Nesse caso configura-se uma condição de exigência mais elevada para o Cenário 1. Em todos os gráficos da Figura 4, para as classes A, B e C, nota-se que as combinações (FIFO e SJF) para a abordagem *MQ* não foram capazes de cumprir os contratos da classe A. Entretanto, mantiveram as garantias nos cumprimentos dos contratos das classes B e C.



(a) CONTRATO A (b) CONTRATO B (c) CONTRATO C

Figura 4. Comparativo do Cenário 1 (MQ-(FIFO,SJF) X DSQ-(EDF,EBS)) com Variação Contratual de 50%

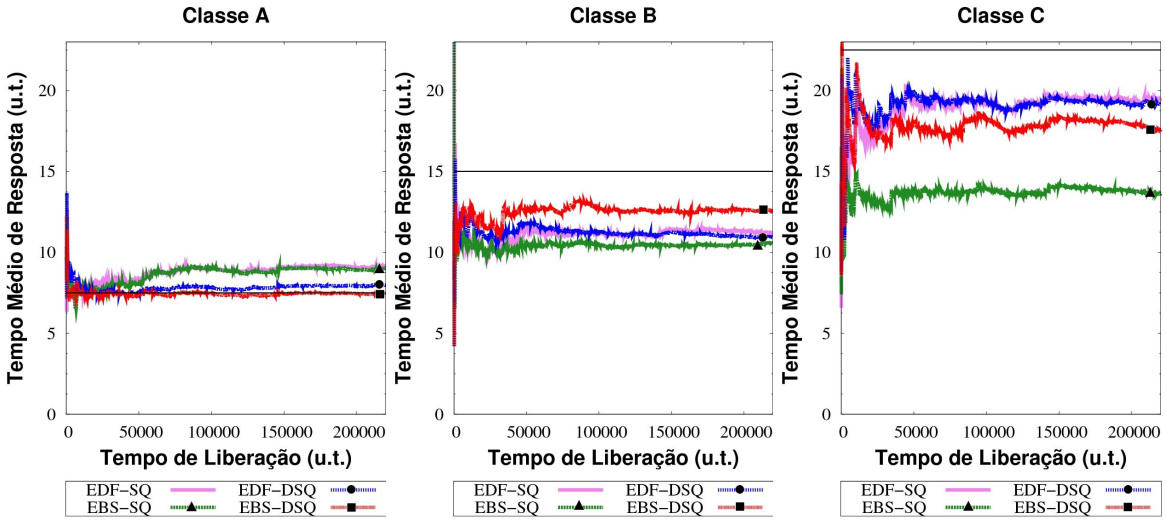
Fato igual acontece com a abordagem *EDF-MQ e EBS-MQ*. Os gráficos da Figura 5 mostram que a abordagem *MQ* é limitada quando a taxa de variação contratual torna-se alta, fazendo com que o cenário em questão apresente um caráter de maior exigência para com as requisições da classe A.



(a) CONTRATO A (b) CONTRATO B (c) CONTRATO C

Figura 5. Comparativo do Cenário 1 (MQ-(EDF,EBS) X DSQ-(EDF,EBS)) com Variação Contratual de 50%

Apesar da abordagem *SQ* possuir tempos médios de resposta menores, segundo os gráficos da Figura 6 não foi capaz de cumprir os contratos da classe A. Isso mostra que apesar do desempenho empregado por essa abordagem ser eficiente, isso não é suficiente para que todas as classes recebam um tratamento de forma diferenciada, criando assim a diferenciação de serviço.



(a) CONTRATO A (b) CONTRATO B (c) CONTRATO C

Figura 6. Comparativo do Cenário 1 (SQ-(EDF,EBS) X DSQ-(EDF,EBS)) com Variação Contratual de 50%

Contudo, é a política *DSQ*, principalmente a combinação *EBS-DSQ*, que apresenta o fenômeno singular (entre os casos estudados, Figuras 4, 5 e 6) de inverter-se nessa relação de desempenho, passando a ser a que oferece menores tempos médios de resposta para a classe A mais exigente, à custa de uma ligeira diminuição de desempenho

para a classe B, embora ainda plenamente cumprindo o contrato daquela classe. Essa propriedade de buscar a garantia de contratos diferenciadamente, balanceando a exigência imposta ao escalonador no atendimento de cada classe, demonstra que a *DSQ*, dentre os três algoritmos ensaiados, é a que opera de modo mais coerente com as características da política *EBS*, sendo a única que realiza o cumprimento dos contratos da classe A.

A validação estatística confirma que os menores intervalos para o tempo médio de resposta do Cenário 1 é realmente da abordagem *EBS-DSQ*, segundo uma estipulação de variação contratual de 50%. A Tabela 2 mostra o intervalo de confiança dos dados da classe A que foram traçados para as principais abordagens desenvolvidas em todos os Cenários.

Tabela 2. Intervalo de Confiança de Ω

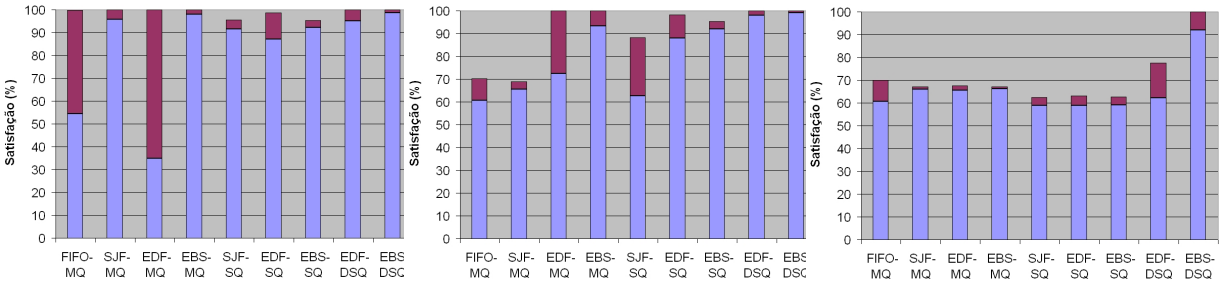
Classe A - Cenário 1 com 50%								
	FIFO-MQ	SJF-MQ	EDF-MQ	EBS-MQ	EDF-SQ	EBS-SQ	EDF-DSQ	EBS-DSQ
Inferior	13,19	11,78	9,87	9,84	8,89	8,81	7,67	7,30
Superior	15,33	13,28	10,62	10,55	9,25	9,08	8,08	7,63
Classe A - Cenário 2 com 40%								
	FIFO-MQ	SJF-MQ	EDF-MQ	EBS-MQ	EDF-SQ	EBS-SQ	EDF-DSQ	EBS-DSQ
Inferior	13,18	11,77	10,17	10,05	9,09	9,03	8,07	8,42
Superior	15,34	13,24	10,95	10,68	9,53	9,27	8,85	8,78
Classe A - Cenário 3 com 50%								
	FIFO-MQ	SJF-MQ	EDF-MQ	EBS-MQ	EDF-SQ	EBS-SQ	EDF-DSQ	EBS-DSQ
Inferior	13,11	11,76	9,14	9,74	8,68	8,59	7,30	6,89
Superior	15,32	13,23	11,51	10,50	9,26	9,02	8,00	7,41
Classe A - Cenário 4 com 30%								
	FIFO-MQ	SJF-MQ	EDF-MQ	EBS-MQ	EDF-SQ	EBS-SQ	EDF-DSQ	EBS-DSQ
Inferior	13,19	11,83	10,65	10,32	9,98	9,49	9,98	10,21
Superior	15,32	13,16	12,35	11,04	11,34	9,81	11,39	10,57
Classe A - Cenário 5 com 40%								
	FIFO-MQ	SJF-MQ	EDF-MQ	EBS-MQ	EDF-SQ	EBS-SQ	EDF-DSQ	EBS-DSQ
Inferior	13,19	11,83	10,49	10,22	9,43	9,19	8,74	8,76
Superior	15,32	13,16	11,22	10,80	9,68	9,34	9,12	8,95

6. Avaliação da Satisfação do Cliente

A análise dos resultados por meio do tempo médio de resposta é importante pois fornece detalhes do comportamento do escalonador. No entanto, uma visão mais abrangente, com maior validade estatística também se faz necessária. Por ela pode-se constatar se a eficiência e desempenho vistos nos gráficos anteriores são ou não casos isolados.

A satisfação de um dado usuário mostra o percentual de vezes que a média real oferecida ficou dentro da faixa de tolerância especificada por cada contrato de serviço. O número dos tempos de respostas oferecidos abaixo da média contratada quantifica a variação da qualidade de um dado atendimento, pois, em média, a satisfação pode se manter alta, mas os valores individuais dos tempos de respostas oferecidos podem variar muito. Esse percentual reflete a estabilidade da qualidade do escalonamento oferecido.

Os gráficos da Figura 7 apresentam um comparativo estatístico da satisfação contratual, obtidos através da média de todas as classes e pelas diferentes disciplinas de escalonamento do cenário 1. A parte mais escura da barra dos gráficos representa o intervalo de confiança da Satisfação dos Usuários no sistema.



(a) Variação Contratual 10% (b) Variação Contratual 30% (c) Variação Contratual 50%

Figura 7. Intervalo de Confiança para a Satisfação dos Usuários do Cenário 1

Pela Figura 7 pode-se verificar como varia a qualidade individual de atendimento das requisições para o Cenário 1. No escalonamento *EBS-DSQ*, mesmo sob grandes variações contratuais das exigências, a qualidade do escalonamento se mantém em nível bom para os três gráficos de variações contratuais. Para variações contratuais de até 50% a qualidade individual dos tempos de resposta reais é melhor que a contratada em mais de 90% dos casos, o que mostra que, mesmo sob graus mais altos de diferenciação, o algoritmo *EBS-DSQ* consegue manter ótima qualidade de serviço, quando o sistema não está saturado por requisições prioritárias. Comparadas às demais políticas, a *SJF-MQ* consegue oferecer pouca oscilação de qualidade para classes prioritárias, mas com pouca diferenciação de serviço, apenas para contratos até 10% mais exigentes. A política *EDF-DSQ*, para o Cenário 1 com alta variação contratual, apresentou qualidade de tempo de resposta inferior à *EBS-DSQ* e alta oscilação no intervalo de confiança.

7. Política Adaptativa

É possível utilizar as vantagens de cada abordagem em relação às características corrente do ambiente. A especificação da arquitetura adaptativa motiva o uso de todas as abordagens em conjunto, realizando a troca entre elas no momento em que uma se mostra superior. Essa troca é dada de acordo com as características das variações contratuais conhecidas a priori. O gráfico da Figura 8 representa os melhores estados de cada modelo já apresentado para cada Cenário em um dado instante da variação contratual. Essa análise é realizada exclusivamente através da métrica de tempo médio de resposta da Classe A, que determina a maior exigência do sistema.

Observa-se que em Cenários que possuem baixa variação contratual a abordagem *EBS-SQ* é indicada; quando a variação é intermediária o uso da abordagem *EDF-DSQ* apresenta melhores resultados. Por outro lado, é melhor usar a abordagem *EBS-DSQ* para variações contratuais altas. Por outro lado, quando a métrica envolvida é a Satisfação do Cliente, determinada por meio da média das três classes, Figura 9, observa-se que a métrica preponderante é a *EBS-DSQ*.

O custo envolvido entre as trocas das abordagens é baixo. Mudar *EBS-SQ* para *EDF-DSQ* significa apenas ativar o cálculo do (τ) utilizando assim a fila virtual na

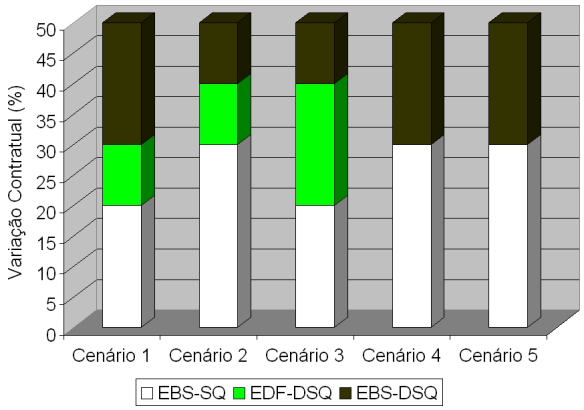


Figura 8. Abordagem Adaptativa com Referência o Tempo Médio de Resposta

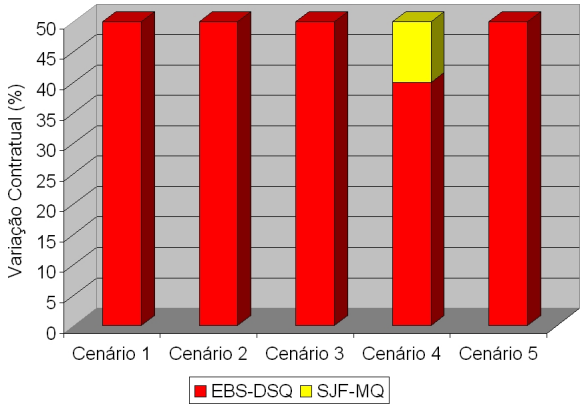


Figura 9. Abordagem Adaptativa com Referência a Satisfação do Cliente

política de recurso e desativar o uso da informação de carga de trabalho (T_{p_j}) no cálculo das prioridades da política de fila. A mudança do modelo *EDF-DSQ* para *EBS-DSQ* é feita por meio da inclusão da informação de carga das requisições na política de fila.

Essas trocas realizadas na política adaptativa podem ocorrer em alguns momentos do ambiente, com o objetivo de melhorar os tempos médios de resposta e possivelmente a Satisfação dos Clientes.

8. Conclusões Gerais

Nesse projeto foi introduzida uma nova técnica que utiliza uma arquitetura ortogonal para dinamicamente escalonar requisições aos servidores compostos no *array*. A abordagem proposta controla automaticamente o tempo de escalonamento para garantir que o serviço seja entregue segundo o contrato estipulado da *QoS*. Os algoritmos e modelos foram simulados considerando diferentes parâmetros e configurações do ambiente.

O objetivo deste projeto foi de investigar um modelo conjugado de escalonamento — uma política de fila e uma política de recurso — adequada às características dos sistemas interativos *on-line*, com foco em uma solução otimizada para *array* de servidores Web heterogêneos, proporcionando assim a gerência dos tempos de respostas contratados, e ao mesmo tempo maximizando a utilização de recursos.

O modelo clássico *Web Switch*, *MQ*, com balanceamento de carga foi utilizado como referência: nessa arquitetura, as requisições são direcionadas para diferentes processadores, cada qual com sua fila individual. Nestas, foi a disciplina de fila *EBS* que obteve melhores resultados, já demonstrada superior às alternativas clássicas para o caso de restrições de tempo-real estocásticas.

Um primeiro modelo alternativo baseado em uma fila única foi inicialmente comparado com a *Web Switch* clássica. Nesse esquema, denominado *SQ*, os melhores resultados para a disciplina da fila única é ordenada segundo o fator de exigência das requisições presentes. A requisição mais à frente da lista (prioritária segundo o critério *EBS*) é atribuída ao mais rápido dos servidores livres a qualquer momento.

Em uma segunda alternativa, a política *DSQ* implementa um híbrido entre o *SQ* com fila única, e o *Web Switch* com balanceamento. Esta terceira modalidade possui também a visão global da fila de requisições pendentes, mas em lugar de atribuir a requisição mais prioritária ao primeiro servidor livre, a *DSQ* faz, segundo o mesmo princípio da *EBS*, uma previsão do impacto causado ao sistema, buscando atribuir ao processador livre um job prioritário (não necessariamente o primeiro), que tenha chance de ser completado mais rapidamente naquele recurso.

Os resultados experimentais mostraram que as melhores combinações foram: *SQ-EBS, DSQ-EDF e DSQ-EBS*. As avaliações de desempenho destes modelos demonstraram que eles provêem uma taxa de atendimento robusta e garantias de serviços absolutos com relação às instantâneas mudanças nas cargas de trabalho da Web.

Apesar dos tempos médios de resposta serem significativamente maiores para a abordagem *MQ*, é importante ressaltar que a *MQ* possui a vantagem da tolerância a faltas, pois ela distribui a carga entre os servidores presentes no sistema não necessitando de um centralizador para gerenciamento do sistema, como também o *overhead* nulo para identificação de um servidor livre, o qual é necessário para as abordagens *SQ e DSQ*.

No momento em que o cenário corrente apresenta uma alta variação contratual, é utilizado DSQ-EBS, devido a propriedade de balanceamento inerente ao escalonador. A *DSQ* é o algoritmo que melhor adaptou-se às características da política de fila EBS.

Finalmente, foi implementado um mecanismo de controle de admissão para todas as abordagens, que emprega uma média exponencialmente ponderada da utilização do *array* de servidores para orientar as decisões de descarte. Este mecanismo consegue manter a carga de trabalho sempre abaixo do limiar especificado, evitando aceitar no sistema requisições que não poderão ser atendidas. A variação do peso da média ponderada permite, ainda, ajustar a sensibilidade do sistema a mudanças no perfil da carga de trabalho.

Os experimentos realizados permitem concluir que a utilização de um Controle de Admissão é fundamental para o fornecimento de serviços diferenciados na Web com uma melhor qualidade para abordagem de sistemas de Tempo-real que lidam com o restrições temporais, tornando-se a diferenciação de serviços ainda mais eficiente, possibilitando um serviço com característica estável e, principalmente, mais justa. Assim, evita-se penalizar ou favorecer demasiadamente uma das classes de clientes.

Devido ao muitos parâmetros que são envolvidos, e certamente não se abordaram todos eles neste trabalho, mas tentou-se relacionar os mais importantes e que refletissem alguma medida de desempenho, indicando o quão bom ou ruim seria o atendimento para uma dada classe de serviço. Aliás, a grande quantidade de variáveis envolvidas quando se trabalha especificamente com servidores Web é uma dificuldade que surgiu ao longo do desenvolvimento deste estudo e que continua sendo muito explorada em trabalhos da área de análise de desempenho em servidores Web ao longo dos anos.

9. Agradecimentos

Os autores agradecem o apoio financeiro da CAPES, CNPq e FAPESP e as contribuições de Lucas Casagrande.

Referências

Abdelzaher, T., Stankovic, J., Lu, C., Zhang, R., and Lu, Y. (2003). Feedback performance control in software services. IEEE Control Systems.

Almeida, J., Dabu, M., Manikutty, A., and Cao, P. (1998). Providing differentiated levels of service in web content hosting. Technical Report CS-TR-1998-1364, University of Wisconsin-Madison.

Bampis, E. and Kononov, A. (2001). On the approximability of scheduling multiprocessor tasks with time-dependent processor and time requirements. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 200. IEEE Computer Society.

Barbato, A. K., Traldi, O., Santana, R. H. C., Santana, M. J., and Teixeira, M. M. (2006). Algoritmo de escalonamento para servidores web baseado em sessão. *XII Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia)*. Natal, RN.

Casagrande, L. S., Monaco, F. J., de Mello, R. F., Bertagna, R., and Filho, J. A. A. (2007). Exigency-based real-time scheduling policy to provide absolute QoS for web services. In *SBAC-PAD '07: Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing*, http://doi.ieeecomputersociety.org/10.1109/SBAC-PAD.2007.22. IEEE Computer Society.

Eggert, L. and Heidemann, J. (1999). Application-level differentiated services for web servers. *World Wide Web*, 2(3):133–142.

Franklin, G. F. (1993). *Feedback Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Henriksson, D., Lu, Y., and Abdelzaher, T. (2004). Improved prediction for web server delay control. In *ECRTS '04: Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS'04)*, pages 61–68. IEEE Computer Society.

Kang, K.-D., Son, S. H., and Stankovic, J. A. (2003). Differentiated real-time data services for e-commerce applications. *Electronic Commerce Research*, 3(1-2):113–142.

Lee, W. Y., Hong, S. J., and Kim, J. (2003). On-line scheduling of scalable real-time tasks on multiprocessor systems. *J. Parallel Distrib. Comput.*, 63(12):1315–1324.

Liebeherr, J., Burchard, A., Oh, Y., and Son, S. H. (1995). New strategies for assigning real-time tasks to multiprocessor systems. *IEEE Trans. Comput.*, 44(12):1429–1442.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of ACM*, 20(1):46–61.

Lu, C., Abdelzaher, T. F., Stankovic, J. A., and Son, S. H. (2001). A feedback control approach for guaranteeing relative delays in web servers. In *RTAS '01: Proceedings of the Seventh Real-Time Technology and Applications Symposium (RTAS '01)*, page 51. IEEE Computer Society.

MacDougall, M. H. (1989). *Simulating Computer Systems*. Computer Systems. MIT Press, Cambridge, Massachusetts London, England, 2 edition.

Mok, A. K. (1983). Fundamental design problems of distributed systems for the hard real-time environment. MIT, Cambridge, Mass.

Peixoto, M. L. M., Tott, R. F., and Monaco, F. J. (2007). Política de escalonamento de tempo-real para garantia de QoS absoluta em cluster de servidores web heterogêneos. In *WebMedia '07: XIII Simpósio Brasileiro de Multimídia e da Web*, Gramado, RS, Brazil. ACM Digital Library.

Ramamritham, K., Stankovic, J. A., and Shiah, P. F. (1990). Efficient scheduling algorithms for real-time multiprocessor systems. *IEEE Trans. Parallel Distrib. Syst.*, 1(2):184–194.

Rashid, M. M., Alfa, A. S., Hossain, E., and Maheswaran, M. (2005). An analytical approach to providing controllable differentiated quality of service in web servers. *IEEE Trans. Parallel Distrib. Syst.*, 16(11):1022–1033.

Stankovic, J. A., Lu, C., and Son, S. H. (1998). The case for feedback control real-time scheduling. Technical report, Charlottesville, VA, USA.

Stankovic, J. A., Spuri, M., Natale, M. D., and Buttazzo, G. C. (1995). Implications of classical scheduling results for real-time systems. *Computer*, 28(6):16–25.

Teixeira, M. A. M., Santana, M. J., and Santana, R. H. C. (2005). Servidor web com diferenciação de serviços: Fornecendo QoS para os serviços da internet. In XXIII Simpósio Brasileiro de Redes de Computadores (SBRC), Fortaleza, CE.

Traldi, O. A., Barbato, A. K., and Santana, R. H. C. (2006). Service differentiating algorithms for QoS-enabled web servers. In *WebMedia '06: Proceedings of the 12th Brazilian symposium on Multimedia and the web*, pages 263–272. ACM Press.

Vasiliou, N. and Lutfiyya, H. (2000). Providing a differentiated quality of service in a world wide web server. *SIGMETRICS Perform. Eval. Rev.*, 28(2):22–28.

Ye, N., Gel, E. S., Li, X., Farley, T., and Lai, Y.-C. (2005). Web server QoS models: applying scheduling rules from production planning. *Comput. Oper. Res.*, 32(5):1147–1164.

Zhang, R., Lu, C., Abdelzaher, T. F., and Stankovic, J. A. (2002). Controlware: A middleware architecture for feedback control of software performance. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 301. IEEE Computer Society.