Uma Avaliação Empírica de Políticas de Invocação para Serviços Web Replicados

José Airton Fernandes da Silva, Nabor das Chagas Mendonça

Mestrado em Informática Aplicada – Universidade de Fortaleza
Caixa Postal 1258 – 60.811-905 – Fortaleza – CE

jairton.mia@unifor.br, nabor@unifor.br

Abstract. *This paper presents an invocation framework for geographically replicated web services. The framework includes several server selection and invocation policies which aim at reducing web service access time at the client side. The invocation policies implemented were evaluated empirically through a series of experiments involving a web service replicated in four servers distributed over three continents, and two clients with different connection characteristics. The results show that, in addition to the individual capacity of each server, the replica selection process is affected mainly by client differences in terms of network configuration and workload distribution throughout the day.*

Resumo. *Este artigo apresenta um framework para a invocação de serviços Web geograficamente replicados. O framework incorpora diversas políticas de seleção e invocação de servidores na perspectiva de tornar mais eficiente o acesso ao serviço no lado do cliente. As políticas implementadas foram avaliadas empiricamente através de experimentos envolvendo um serviço replicado em quatro servidores distribuídos em três continentes, e dois clientes com diferentes características de conexão. Os resultados obtidos mostram que, além da capacidade individual de cada servidor, a escolha da melhor réplica do serviço é afetada principalmente pelas diferenças de configuração de rede entre os clientes, bem como pela distribuição da carga de trabalho de cada um ao longo do dia.*

1. Introdução

Os serviços Web são uma padronização do novo paradigma de *Computação Orientada a Serviço* [12] para o contexto da Web, constituindo-se num poderoso mecanismo de interoperabilidade e integração entre aplicações distribuídas a partir de plataformas heterogêneas. Os serviços Web, mais que simples recursos da Web (página, imagem, *script,* etc), são aplicações fracamente acopladas que utilizam tecnologias padronizadas da Internet para se localizarem e se comunicarem entre si. As principais tecnologias utilizadas são: SOAP (*Simple Object Access Protocol*) [17], que é um protocolo de comunicação com estrutura de mensagens em XML; WSDL (*Web Services Definition Language*) [20], que é uma linguagem baseada também em XML para descrição de serviços; e UDDI (*Universal Discovery, Description and Integration*) [19], que é uma especificação para registro e localização de serviços.

Os serviços Web surgem num cenário onde a qualidade de serviço é um fator preponderante, dada a concorrência cada vez maior por recursos na Internet [4]. Nesse

contexto, a replicação de recursos pode oferecer uma alternativa interessante, tanto para aumentar a disponibilidade dos serviços, quanto para reduzir a carga nos seus respectivos servidores através do balanceamento no atendimento às requisições dos clientes [6].

As abordagens que tratam do acesso a recursos replicados na Internet estão divididas, de um lado, em mecanismos para a distribuição automática das requisições dos clientes, no lado do servidor, e, de outro lado, em mecanismos, incorporados nos próprios clientes ou em servidores próximos a eles, para explorar formas mais efetivas de acesso a serviços geograficamente replicados. Pelo lado do cliente, várias pesquisas (por exemplo, [1][7][5][16]) já foram conduzidas com foco na avaliação de políticas de seleção de servidores, no sentido de identificar aquela que possa oferecer o melhor desempenho. Essas pesquisas, porém, se limitaram a avaliar políticas de acesso a recursos na forma de documentos HTML e *downloads* de arquivos. Embora sejam conceitualmente aplicáveis ao domínio dos serviços Web, na prática essas políticas não permitem uma aplicação direta devido às características dos serviços Web, que estão mais próximos ao modelo de execução de objetos distribuídos na Internet do que da simples recuperação de documentos a partir do sistema de arquivos de um servidor Web.

Esse trabalho tem como objetivo contribuir para melhorar o acesso a serviços Web geograficamente replicados, explorando o modelo de descrição de serviços oferecido pela linguagem WSDL para tratar a localização das réplicas, e avaliando alternativas que tornem mais eficiente o processo de invocação das réplicas no lado do cliente. Para atender a esses objetivos, implementamos o *framework* RWS (*Replicated Web* Services) como uma extensão do *framework* AXIS [2], do Apache Group, que dá suporte à implementação de serviços Web em Java. Assim como o AXIS, na sua forma original, outros *frameworks* similares (por exemplo, .NET [9] e JWSDP [18]) também não exploram de forma adequada alternativas para melhorar o acesso a serviços replicados, deixando a cargo da aplicação a decisão sobre qual réplica do serviço invocar.

Em contraste aos *frameworks* existentes, o *framework* RWS implementa um processo transparente de invocação de serviços Web replicados, baseado em diferentes políticas de seleção de servidores. As políticas implementadas foram avaliadas empiricamente, através de experimentos envolvendo um serviço replicado em quatro servidores distribuídos em três continentes. Este serviço foi invocado utilizando as diferentes políticas de seleção de réplicas, a partir de dois clientes com diferentes características de conexão. Os resultados obtidos mostram que, além da capacidade individual de cada servidor, a escolha da melhor réplica do serviço é afetada principalmente pelas diferenças de configuração de rede entre os clientes, bem como pela distribuição da carga de trabalho de cada um ao longo do dia.

O restante do artigo está organizado da seguinte maneira. Na seção 2, apresentamos os trabalhos relacionados com a presente pesquisa. Na seção 3, detalhamos a implementação do *framework* RWS. Na seção 4, descrevemos a metodologia utilizada na condução dos experimentos. Na seção 5, comparamos o desempenho das diversas políticas através de uma análise quantitativa dos resultados. Por fim, na seção 6, concluímos o artigo com um resumo dos principais resultados e apontamos possíveis linhas de trabalhos futuros.

2. Trabalhos Relacionados

De forma geral, a idéia da replicação é manter cópias dos mesmos recursos em servidores diferentes [6]. Um cliente pode contatar qualquer um dos servidores disponíveis que tenha uma réplica do recurso requerido. Isso reduz a carga nos servidores e conseqüentemente a latência causada pela demora no atendimento das solicitações.

Duas abordagens têm sido utilizadas para tratar a replicação de recursos geograficamente distribuídos na Internet: uma abordagem que foca na redireção de requisições no lado do servidor e outra que foca no acesso aos recursos replicados pelo lado do cliente. As estratégias adotadas no lado do servidor objetivam maximizar a capacidade de atendimento de solicitações (*througput)* com pouca atenção ao tempo de latência experimentado pelos clientes. Por outro lado, com o objetivo de minimizar o tempo de resposta, um cliente pode utilizar mecanismos que o permitam conhecer quais são os melhores provedores de um serviço. Uma estratégia de tratar a escolha do servidor pelo cliente tem a vantagem de que pelo lado do cliente temos uma visão geral da rede.

Os trabalhos [13] e [15] propuseram mecanismos através dos quais o servidor informava a existência e a localização das réplicas à aplicação cliente. Em [13] foram incorporadas informações ao *header* do HTTP onde o servidor enviava a lista de réplicas ao cliente. A lista era mantida por um servidor *proxy* junto ao cliente. O serviço de seleção foi implementado no *proxy*. Em [15], no lado do cliente, foi implementada uma *applet* que era incorporada ao *browser*. A lista de réplicas era enviada a partir dos servidores que participavam da arquitetura proposta em [15] e mantida na *applet* do lado do cliente. A *applet* interceptava as requisições do *browser* e realizava o processo de seleção do servidor.

Os recursos e serviços na Internet são replicados, em alguns casos, geograficamente em vários pontos para melhorar o desempenho e disponibilidade. Neste modelo de replicação, e numa visão pelo lado do cliente algumas pesquisas recentes (por exemplo, [1][7][5][15]) têm sido direcionadas para a aplicação de políticas e mecanismos de seleção de servidores, com foco, principalmente, em propor alternativas para redirecionamento de requisições dos clientes ou, simplesmente, na avaliação das políticas propostas.

As políticas de seleção de servidores, até então estudadas, estão distribuídas no seguinte agrupamento:

- Estáticas – estão baseadas em recursos de infra-estrutura, tais como taxa de conexão da rede, hardware do servidor, número de *hops* etc.
- Estatísticas – estão fundamentadas em bases históricas coletadas de informações de requisição a servidores e submetidos a tratamentos estatísticos.
- Dinâmicas – Utilizam mecanismos que detectam as condições da rede e carga do servidor, no instante imediatamente anterior ao da requisição do recurso, aplicando o envio de sondas ao servidor na forma de *ping*, *download* de pequenos arquivos etc.

No presente trabalho, abordamos as políticas de seleção de servidores no contexto dos serviços Web replicados, na visão do lado do cliente. Estudos anteriores [7][14][15] focaram na avaliação de políticas, também pelo lado do cliente, no acesso a recursos quaisquer replicados na Web, não incluindo os serviços Web na forma caracterizada na introdução deste trabalho. Naqueles estudos, tinha-se a figura do *browser* que incorporava elementos adicionais de software.

Classificamos os estudos anteriores, quanto às políticas de seleção de servidores de recursos Web e mecanismo de identificação das réplicas, nas seguintes abordagens:

- O trabalho de Sayal [15] trata o mapeamento das réplicas existentes definindo mecanismo através do qual o servidor informa alterações aos clientes, em cada requisição. Inclui políticas de seleção de servidores
- Em [7] há o foco apenas no estudo das políticas de seleção, assumindo a existência e o conhecimento da localização das réplicas. Não define nenhum mecanismo de atualização dos endereços das réplicas.
- Nos trabalhos [1][10][14] o objetivo principal está apenas na avaliação de políticas de seleção de servidores, mesmo sem que estes contivessem réplicas dos mesmos recursos.

Este trabalho está inserido dentro da primeira entre as três abordagens citadas acima. O mapeamento das réplicas existentes toma como base as informações de localização obtidas a partir de um documento WSDL. O cliente configura os endereços numa fase inicial, quando as classes auxiliares são criadas e posteriormente estas informações são atualizadas de forma dinâmica.

As políticas propostas em trabalhos anteriores [5][7][15] são aplicáveis ao domínio dos serviços Web, porém, exigem tratamento especial em virtude da arquitetura que compõe o lado do cliente. Para isso propomos o *framework RWS*, descrito na seção seguinte, que define a incorporação, implementação e utilização das políticas a serem aplicadas na seleção dos servidores.

3. Um *Framework* para Invocação de Serviços Web Replicados

Nesta seção, apresentamos o *framework* RWS descrevendo aspectos ligados à localização das réplicas e estratégias de acesso implementadas.

3.1 Localização das Réplicas

O acesso a recursos replicados no lado cliente exige, entre outras informações, a identificação e a localização das réplicas. No contexto de serviços Web, informações sobre as réplicas que estão disponíveis para um determinado serviço, bem como informações sobre a localização de cada uma, podem ser obtidas diretamente a partir do documento WSDL definido para o serviço.

Num documento WSDL, estruturado em XML, as informações necessárias para a localização e a invocação de um serviço estão descritas no elemento <service>. Esse elemento inclui um elemento <port> com a descrição do tipo de protocolo de ligação (HTTP, SMTP, etc) através do qual o serviço pode ser acessado. <port> por sua vez inclui o elemento <address> com a descrição do endereço do serviço na Internet.

Um mesmo Serviço Web pode ser invocado via diferentes protocolos de ligação. Por esse motivo, a especificação WSDL prevê que um elemento <service> pode conter

zero ou mais elementos do tipo <port>. No entanto, a especificação não impõe nenhuma restrição quanto ao valor do elemento <address> de cada elemento <port>. Portanto, através da inclusão de múltiplos elementos <port> também é possível definir diferentes endereços para um mesmo serviço. No caso de serviços Web replicados, esses valores poderiam representar os endereços das suasr espectivas réplicas. A Figura 1 mostra um exemplo de um segmento do documento WSDL com a descrição de dois endereços para o mesmo serviço.

```
<wsdl:service name="xxxxxxx">
  <wsdl:port binding="yyyyyyy" name="z1z1z1z1">
    <wsdlsoap:address location="http://location1" />
  </wsdl:port>
  <wsdl:port binding="yyyyyyy" name="z2z2z2z2">
    <wsdlsoap:address location="http://location2" />
  </wsdl:port>
</wsdl:service
```

Figura 1. Exemplo de um elemento <service> com dois endereços.

3.2 Escolha e Invocação da Melhor Réplica

Escolher a melhor réplica de um serviço replicado não é uma tarefa trivial. Variações na latência e na capacidade de transmissão da rede, e na capacidade de conexão e processamento dos clientes, entre outras, são fatores que podem ser decisivos para determinar a réplica a ser invocada pela aplicação cliente. Para explorar estas características, torna-se necessário dar um tratamento adequado à informação sobre a existência de réplicas contidas no documento WSDL do serviço. Uma vez obtida essa informação, a mesma poderá ser incorporada diretamente à aplicação, ou a camadas próximas a ela, como no *stub* gerado pela *middleware* SOAP utilizado ou até mesmo num serviço do tipo *proxy* inteligente.

Considerada a existência das réplicas, e com os seus respectivos endereços devidamente identificados, o que interessará à aplicação é que a requisição enviada ao serviço replicado seja atendida da melhor forma possível, não importando para qual réplica a solicitação seja encaminhada. Para tornar a escolha da réplica o mais transparente possível para a aplicação, o ideal é que as camadas auxiliares, sejam na forma de *stub*, ou na forma de um servidor *proxy* inteligente, incorporem mecanismos que busquem a invocação do modo mais eficiente. Nesse contexto, o *framework* RWS permite a geração automática das camadas que estarão ligadas à aplicação cliente, na forma de *stub*, com a incorporação da informação necessária para escolher e invocar, automaticamente, de acordo com diferentes critérios, as réplicas de um serviço Web replicado. O RWS também permite a atualização dinâmica desses endereços, e a inserção de novas políticas de invocação.

Cabe ressaltar que, no escopo deste trabalho, pressupõe-se que réplicas dos serviços são conhecidas e mantidas em um estado consistente entre si, cabendo ao provedor a responsabilidade pelo gerenciamento, manutenção e publicação das informações sobre as réplicas no documento WSDL que estará disponível nos serviços de registro. Portanto, não abordamos nenhuma proposta quanto à alocação e

gerenciamento das réplicas, mas apenas à avaliação de políticas de invocação para garantir um acesso mais eficiente às réplicas no lado do cliente.

3.3 Aspectos de Implementação

A motivação para a implementação do RWS foi estender o *framework* AXIS no processo da geração das classes auxiliares que vão estar ligadas diretamente à aplicação cliente, além de alterar o comportamento dessas classes na invocação do serviço. A opção pelo AXIS deve-se ao fato de que o mesmo implementa um processo de geração da classe *stub* mais próximo ao modelo aqui proposto, e também pelo fato do seu código fonte ser totalmente aberto.

O *framework* RWS foi implementado acrescentando dois novos componentes, o *Gerente de Seleção* e o *Gerente de Invocação*, à arquitetura da aplicação cliente originalmente utilizada pelo AXIS, conforme mostra a Figura 2. Com o tratamento dado ao documento WSDL, no que se refere à localização, é facultado à aplicação cliente especificar um endereço para acessar um serviço ou delegar ao objeto *stub* a escolha de um entre os vários disponíveis. Na segunda alternativa, o comportamento da classe *stub* é modificado com a incorporação de políticas de seleção de réplicas, onde a definição de para qual réplica será enviada a requisição do serviço poderá ser feita com base em mecanismos que busquem aquele de melhor desempenho.

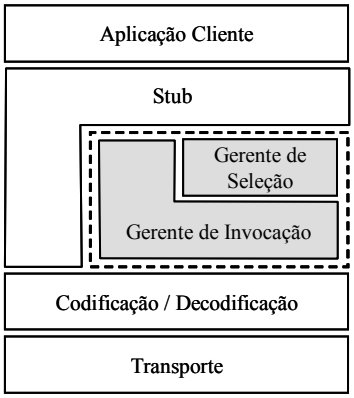

Figura 2. Arquitetura do *framework* RWS.

O conjunto formado pela aplicação, a classe *stub* e as demais classes compõem uma estrutura de cliente inteligente que se assemelha ao modelo proposto em [15][21]. As diferentes políticas de invocação são aplicadas pelo *stub* a partir dos parâmetros fornecidos pela aplicação cliente. Desse modo, o envio da requisição ao serviço passa de uma forma estática, antes definida apenas pela aplicação consumidora, para uma estrutura dinâmica que permite a seleção do melhor servidor, no momento da invocação, entre aqueles que hospedam réplicas do serviço.

Na seção seguinte apresentamos a metodologia utilizada na condução da avaliação empírica das políticas de invocação implementadas.

4. Avaliação Empírica

Nosso experimento incluiu dois clientes que realizaram chamadas a um mesmo serviço Web geograficamente replicado. Os dados referentes ao resultado e ao tempo de resposta de cada chamada foram coletados a partir de uma aplicação cliente, escrita em Java, e estruturada sobre o *framework RWS,* que incorpora as políticas de seleção utilizadas na avaliação. A coleta dos dados foi realizada durante oitos dias consecutivos, 24 horas por dia, no período de 22 a 29 de julho de 2003.

4.1 Métrica de Desempenho

Adotamos o tempo de resposta obtido pela aplicação cliente como métrica para a medida de desempenho na seleção dos servidores. Esta medida tem como objetivo refletir todo o tempo experimentado pelo cliente em qualquer uma das políticas de invocações, visto que esse é o tempo de interesse na utilização do serviço. Estudos semelhantes adotaram métricas diferentes em suas avaliações, tais como HTTP *request RTT* [16] e *Round Trip latency* [11].

4.2 Clientes e Servidores

A aplicação cliente foi executada, de forma dedicada, em duas estações de trabalho localizadas em Fortaleza (Brasil), as quais estavam conectadas à Internet através de *backbones* diferentes. Uma estação estava instalada no laboratório da Universidade de Fortaleza (Unifor), utilizando o *backbone* da Rede Nacional de Pesquisa – RNP, com conexão de 2Mbps. A outra estava nas dependências do Banco do Nordeste (BNB), utilizando o *backbone* da EMBRATEL, com conexão de 4Mbps. As aplicações clientes das duas estações efetuaram chamadas uma mesma operação dentre as disponibilizadas pelo serviço *UDDI Business Registry* [19], que atualmente é provido por servidores de quatro empresas localizadas em três continentes. São elas: Microsoft e IBM (América do Norte), SAP (Europa) e NTT Com (Ásia). A Figura 3 ilustra a localização geográfica dos quatro servidores e os dois clientes no Brasil.

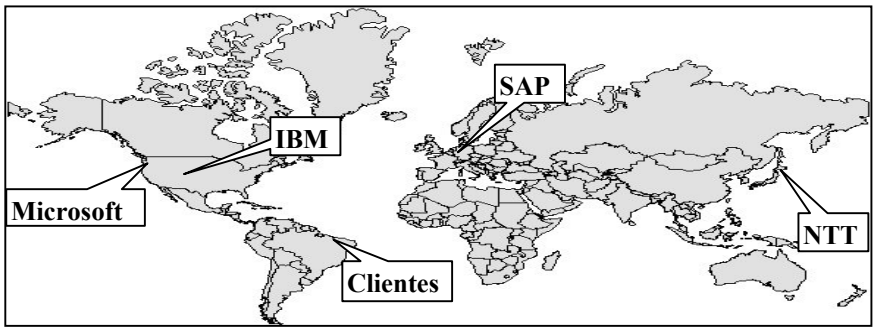

Figura 3. Localização das réplicas do serviço *UDDI Business Registry*.

4.3 Políticas de Seleção Aplicadas

Para a condução da avaliação experimental, as políticas de seleção de servidor, foram incorporadas no *framework RWS* de forma estática, porém a estrutura de implementação utilizada permite a incorporação das políticas no processo automático de geração das classes auxiliares.

As políticas de seleção implementadas e avaliadas estão distribuídas em randômica, dinâmicas e estatísticas, conforme descrição a seguir:

- Randômica - Invocação direta em endereço específico – A escolha é feita de forma aleatória, ou seja, a chamada da operação é enviada a um dos servidores que disponibiliza o serviço.
- Dinâmica Paralela - Invocação dos sites em paralelo – O serviço é buscado de forma concorrente nas URLs onde esteja disponível. A aplicação efetua a chamada simultânea aos servidores com uso de *threads*. A primeira resposta obtida é repassada à aplicação. As demais *threads* são interrompidas e as respostas parciais ignoradas. Esta política é uma modificação do que foi proposto por [14]. Naquele trabalho a requisição para *download* de um mesmo arquivo era feita dividindo-se a requisição para vários servidores. Considerando a chamada de uma operação de um serviço *Web* como uma ação atômica isso não é possível.
- Dinâmica HTTPing - Invocação por sonda – Esta política de invocação faz uso de uma classe adicional que envia uma sonda ao servidor. Uma chamada HTTP HEAD é enviada a cada um dos sites de forma concorrente por *threads*. A chamada da operação é encaminhada para aquele que responde primeiro ao HTTP HEAD. Esta política é uma modificação do que foi proposto por [5],onde foi utilizada a sonda *tcping*.
- Estatística Melhor Última – Nesta política, a seleção do servidor leva em conta a última invocação anterior efetuada a cada uma das réplicas. A informação é obtida a partir de base histórica (log) que armazena dados das invocações anteriores ao serviço. O servidor selecionado é aquele que apresenta o melhor tempo de resposta, entre as últimas invocações, para uma mesma operação de mesma parametrização.
- Estatística Melhor Mediana - Nesta política, a seleção é feita com base no histórico das 60 últimas invocações efetuadas. A partir desses dados, obtemos a melhor relação mediana de todas as invocações tomando como base a divisão dos bytes recebidos pelo tempo em milissegundos. Aquele com a melhor relação mediana é selecionado.

4.4 Sessões, Ciclos e Políticas

As invocações foram distribuídas em sessões que, por sua vez, foram divididas em ciclos. Cada sessão compreendeu a realização de três ciclos. Dentro de cada ciclo foram aplicadas, seqüencialmente, as 5 políticas descritas no item anterior, na seleção do servidor que receberia uma chamada ao serviço. O ciclo um, dois e três tinham como tamanho de resposta esperada 3 Kb, 30 Kb e 300 Kb respectivamente.

Nos ciclos, as invocações aos servidores foram distribuídas de forma que pudéssemos observar o efeito do tempo de resposta de acordo com tamanho da resposta recebida. Em cada ciclo, foram estipulados os tempos máximos para a resposta de uma invocação ao servidor.

Caso a resposta não fosse recebida no tempo esperado, a chamada à operação seria tratada com a exceção de tempo expirado(*time-out*). Na etapa de calibração do experimento, as chamadas às operações dos serviços foram efetuadas sem qualquer controle de tempo por igual período ao da coleta efetiva dos dados. A partir desse

período, os tempos máximos e médios foram identificados e, a partir destes, foram definidos os tempos máximos que a aplicação cliente poderia aguardar pela resposta, dentro do experimento. O tempo máximo de espera refere-se ao tempo de uma chamada completa da operação do serviço, ou seja, envolvendo a requisição e apresentação da resposta na aplicação cliente. Por se tratar de uma chamada de operação de um Serviço Web, que formata a mensagem de requisição e a resposta na estrutura do protocolo XML/SOAP, o recebimento dos pacotes na resposta, adicionado ao processo de serialização/deserialização executado pala camada *parser* XML da aplicação cliente, levou a tempos de resposta consideráveis em determinadas invocações. Esse efeito do processamento exigido pelo *parser* foi objeto de estudo por [8].

A definição de um tamanho e tempo máximo para resposta de uma chamada de operação ao serviço foram as alternativas utilizadas para que pudéssemos manter algumas condições constantes durante a execução de cada ciclo.

Cada sessão do experimento envolveu quinze chamadas à operação do serviço, cinco por ciclo. A duração máxima de uma sessão no cliente Unifor foi de cerca de 31 minutos enquanto no cliente BNB a duração máxima foi em torno de 15 minutos. Esses valores traduzem a diferença entre as larguras de banda disponíveis nos dois clientes para acesso à Internet.

5. Análise de Resultados

Em cada política, a distribuição do tempo de resposta ao longo do dia apresentou alta variabilidade. Esta foi a principal característica observada em todos os ciclos/políticas aplicados. Neste aspecto, o presente trabalho reflete o comportamento observado por outras pesquisas [5][14]. A seguir é descrita a metodologia utilizada para a análise dos dados coletados ao longo do experimento.

5.1 Metodologia de Análise

Os dados coletados foram analisados considerando a mediana como medida de tendência central. A mediana foi utilizada tanto no cálculo dos dados estatísticos, usados pela política Melhor Mediana, quanto na própria avaliação dos tempos de respostas observados para cada uma das políticas implementadas e avaliadas. A decisão pela mediana foi decorrente dos picos nos tempos de resposta observados durante todo o experimento.

Uma análise dos dados foi realizada com a aplicação do método estatístico Z-*Score* para identificação dos pontos de alta variabilidade (*outliers*), dentro de cada ciclo. Os valores de *z-scores* utilizados ficaram no intervalo de +3 a –3 para o tratamento dos dados. Na avaliação, os tempos de resposta com escores acima e abaixo do intervalo estabelecido estariam de alguma forma fora do conjunto típico das observações colhidas na amostra de todos os ciclos/políticas ao longo do período experimental. É importante ressaltar que apenas tempos com escores acima do intervalo estipulado foram evidenciados. Concluímos, assim, que a média da amostra, pela sua própria natureza de ser sensível aos extremos, não refletia o comportamento típico da maioria dos tempos de resposta obtidos. A mediana evidenciou-se como a medida mais adequada para avaliação, uma vez que foi pouco afetada pelos altos tempos de resposta considerados atípicos, a partir do conjunto de dados analisados.

Nas subseções seguintes, destacamos os efeitos dos períodos do dia para a análise dos resultados e o desempenho observado em cada uma das políticas avaliadas.

5.2 Efeitos dos Períodos do Dia

No presente trabalho, fizemos, inicialmente, uma análise no comportamento de cada cliente, no sentido de identificar em qual ponto ocorria um incremento nos tempos de resposta e no momento em que esses tempos diminuíam. Com isso, para a análise, dividiu-se o dia de 24 h em períodos que denominamos de Pa (período de maior uso) e Pb (período de menor uso). A divisão da análise entre períodos já vem sendo utilizada por outros trabalhos que tratam de seleção de servidores. O que acrescentamos é que pode haver variações significativas não apenas nas diferenças entre os tempos obtidos em cada período, mas também nos horários de início e fim desses períodos em cada cliente.

A Figura 4 ilustra a identificação desses dois períodos no dois clientes. No cliente Unifor, Figura 4b mostra que o comportamento das políticas sofre um incremento no tempo de resposta por volta das 8 h e vai até próximo das 22 h. No BNB, o intervalo, que incluí os maiores tempos de resposta, em destaque no gráfico da Figura 4a, compreende o período que vai das 8 h às 19 h. A política randômica foi omitida para uma melhor clareza dos gráficos.

A observação, a partir dos dois clientes, mostra que o comportamento do tempo de resposta está relacionado, em parte, com a carga da rede no lado do cliente, pois, num mesmo período, por volta das 19hs, observamos uma redução no tempo de resposta das melhores políticas no cliente BNB, enquanto que no cliente Unifor esta redução só ocorre após as 22hs. Isso traduz que o desempenho de rede e da carga no servidor do serviço, sob o ponto de vista do cliente BNB, está melhor após as 19 h, porém essa melhora só é percebida pelo cliente Unifor quando a rede do próprio lado do cliente reduz sua carga.

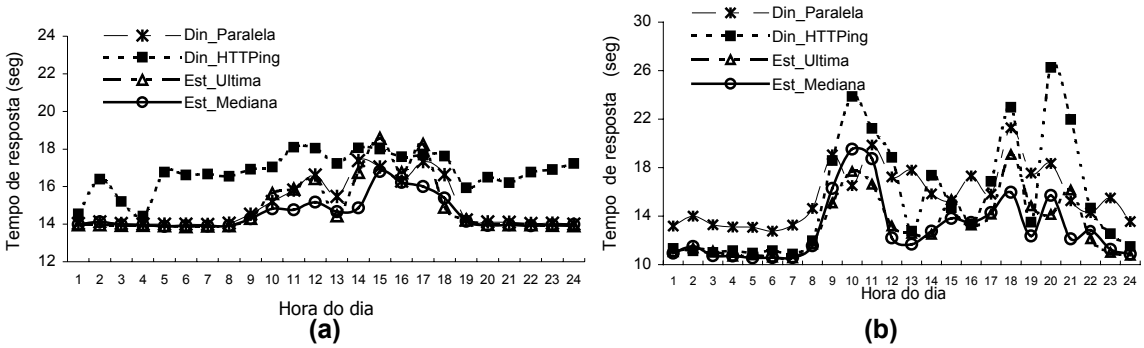

Figura 4. Desempenho das políticas ao longo do dia: cliente BNB (a) e cliente Unifor (b).

5.3 Desempenho das Políticas

A Figura 4 apresenta, na medida de tempo de resposta mediano por hora, o comportamento das políticas em cada cliente. Esse comportamento se assemelha ao que foi observado nos demais ciclos. Para cada política, o tempo de resposta exibe, em determinados momentos do dia, alta variabilidade, para a invocação de uma mesma operação com iguais parâmetros. Isso está consistente com outras pesquisas [5][7].

Devido a essa variabilidade, a utilização da mediana evidenciou-se mais adequada, uma vez que o tempo de resposta mantém-se dentro de certa regularidade e os picos, apesar de serem elevados em dados momentos, não afetaram substancialmente a medida utilizada para a avaliação.

Os gráficos da Figura 4 mostram o efeito da variabilidade do tempo de resposta ao longo das horas do dia. Para a identificação da política com melhor desempenho, elaboramos uma análise global entre elas. As políticas foram agrupadas por ciclos e por períodos de forma que facilitasse a observação do desempenho geral entre elas. As Figuras 5 e 6 ilustram, através de gráficos de colunas, os resultados agrupados por tamanho de resposta (ciclos) e pelos períodos categorizados por Pa e Pb.

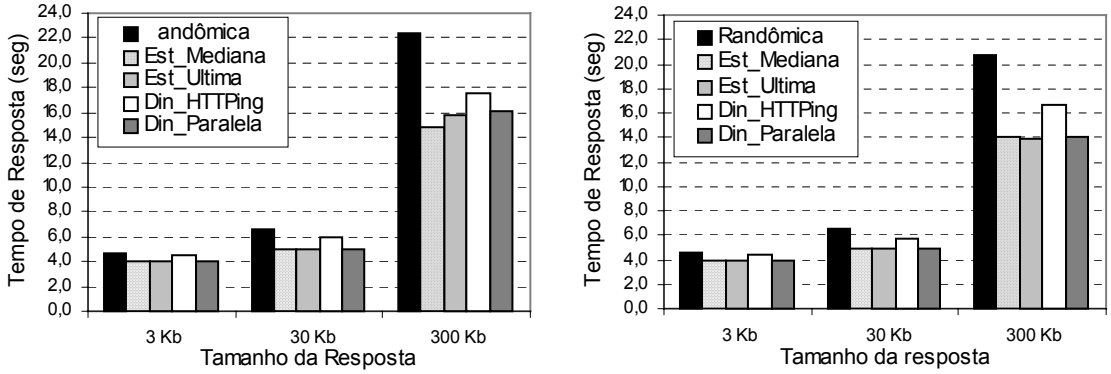

Figura 5. Resumo geral das políticas por ciclos, períodos Pa (esquerda) e Pb (direita) - Cliente BNB

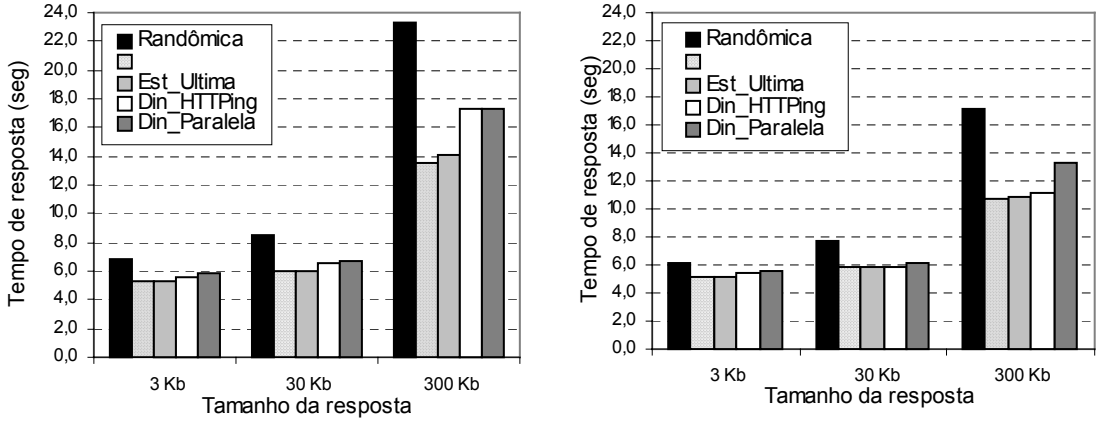

Figura 6. Resumo geral das políticas por ciclos, períodos Pa (esquerda) e Pb (direita) - Cliente Unifor

No cliente BNB, observamos que a política de invocação paralela acompanha as políticas estatísticas em quase todos os ciclos e períodos, com alguma variação no período Pa, que é o período de maior utilização da rede nesse cliente. A política de sonda (HTTPing), por sua vez, apresenta maior tempo de resposta que a paralela e as estatísticas. Por fim, a variação nos tempos de resposta de uma mesma política nos períodos Pa e Pb são relativamente pequenas. Já no cliente Unifor, observamos que a política de invocação paralela, ao contrário do cliente BNB, é bem superior do que as políticas estatísticas, superando até a política HTTPing em todos os ciclos e períodos. A variação do tempo de resposta entre os períodos Pa e Pb é mais acentuada do que no

cliente BNB. Isto demonstra que no período de menor uso da rede desse cliente, os tempos de resposta obtidos são menores.

De forma global, observamos que há pouca variação entre os tempos de reposta das chamadas com tamanho 3Kb e 30Kb, com diferenças da ordem de milissegundos. A razão está associada ao fato de que, nesses casos, o tempo de resposta tende a ser dominado pela latência do protocolo TCP/HTTP.

A política randômica é a que apresenta o maior tempo de resposta independente do cliente, do ciclo e do período. Mostrando-se, portanto, como uma política não adequada para a seleção, pois a escolha randômica do servidor, como era esperado, faz a seleção de servidores que têm tempos de resposta baixos, mas em outros períodos pode fazer escolha de servidor com alta latência na resposta da operação, seja por carga no próprio servidor, na rede ou até mesmo por causa da combinação de servidor Web e camada de tratamento dos serviços Web. O comportamento desta política corresponde ao resultado obtido por [5], ao avaliar o acesso a imagens e documentos HTML.

A política baseada em sonda, HTTPing, está com tempos de resposta medianos acima das políticas estatísticas. Porém, quando relacionada com a paralela apresenta comportamento diferenciado de um cliente para outro. A diferença, entre a HTTPing e as estatísticas, está mais acentuada no ciclo três, no período Pa de ambos clientes. Em [5] a política baseada em sonda (*probe*) mostrou-se como a melhor política, tendo desempenho melhor em todas as condições avaliadas. O presente estudo mostra que a política baseada em sonda não se mostrou adequada, pelo desempenho observado. O fato interessante nesta política foi de que o servidor que geralmente tinha o melhor tempo de resposta para a operação invocada, em alguns momentos, não respondia a sonda em menor tempo. Contrariamente, um servidor que respondia mais rapidamente à sonda, era lento na resposta à chamada da operação, levando a chamada, às vezes, a expirar o tempo máximo de reposta.

As políticas estatísticas apresentaram quase sempre o mesmo desempenho. Como ambas políticas estavam baseadas em respostas anteriores, levando em conta os melhores tempos de resposta, faziam a escolha dos servidores com o melhor desempenho. Na maioria das vezes, essas políticas faziam a seleção entre os dois servidores que historicamente tinham melhor desempenho, ou seja, Microsoft e IBM.

A política paralela teve melhor desempenho que a política por sonda no cliente BNB. O desempenho obtido foi bem similar em todos os ciclos. Já no cliente Unifor, a política paralela foi a terceira melhor política, tendo um resultado melhor apenas do que a randômica.

6. Conclusão

Este trabalho apresentou um *framework* para a invocação de serviços Web replicados. O *framework* incorpora diversas políticas para seleção e invocação de réplicas, de modo a tornar o acesso ao serviço replicado mais eficiente no lado do cliente. As políticas implementadas, incluindo políticas estatísticas, dinâmicas, paralela e randômica, foram avaliadas empiricamente. Os experimentos envolveram a utilização do *framework RWS*, através das suas diversas políticas, para invocar um serviço geograficamente replicado em três continentes, a partir de dois clientes com diferentes configurações de rede e distribuição de carga de trabalho ao longo do dia.

As políticas de seleção que utilizam bases estatísticas históricas se apresentaram como a melhor alternativa quando não se leva em consideração características específicas do cliente, a partir de onde a seleção é realizada. Essa conclusão se contrapõe ao resultado obtido por [5] onde a política baseada em sonda era a melhor em todas as circunstâncias. Observamos nas chamadas de operações dos serviços que há uma regularidade na invocação de servidor para servidor, embora tenhamos picos em alguns momentos. Em virtude do número reduzido de servidores não podemos generalizar os resultados obtidos com as políticas estatísticas.

As políticas estatísticas ficaram bem próximas em termos de desempenho. Estimar qual delas poderá ter melhor desempenho envolve mais medições e repetição dos experimentos aplicando-se outros tratamentos nos tempos de resposta obtidos e, armazenados em bases históricas. O desempenho das políticas estatísticas depende da existência de dados recentes sobre cada servidor. Nos experimentos, este problema foi contornado invocando separadamente os servidores no início de cada ciclo. Uma solução mais realista envolveria o monitoramento das réplicas e reavaliação periódica das bases históricas de forma que elas possam refletir as alterações por *upgrades* tanto em nível de servidor quanto na rede.

A política paralela mostrou-se como bom desempenho onde a largura de banda era suficiente para absorver a concorrência no recebimento dos pacotes na resposta à chamada da operação do serviço. Com largura de banda menor o desempenho é afetado na medida em que aumenta o tamanho de dados na resposta do serviço invocado, isto pode ser percebido no cliente Unifor, a partir da Figura 6. De qualquer modo, o experimento envolveu apenas quatro servidores. Experimentos adicionais são necessários para determinarmos em que ponto, a relação da largura de banda e a concorrência no recebimento dos pacotes podem tornar inviável a utilização dessa política.

Cabe ressaltar que na avaliação de estudos anteriores, o tempo de acesso levava em conta o tempo de rede, da carga no servidor e na recuperação de documentos, imagens e arquivos, a partir do sistema de arquivos. Neste experimento, além dos itens mencionados, tem-se adicionalmente a camada de software adicional para processar o Serviço Web que além do tempo de processamento da operação requisitada tem o fator do tempo de processamento da serialização/deserialização do *midlleware* SOAP que empacota uma chamada de operação**.**

Referências

[1] Amini, L., Shaikh, A., Schulzrinne, H., "Modeling Redirection in Geographically Diverse Server Sets", Proceedings of twelth International Conference on World Wide Web, Budapeste, Hungria, Maio, 2003.

[2] AXIS, The Apache SOAP Project, Version 1.0, Junho, http://ws.apache.org/axis/, 2002.

[3] Chapell, D. and Jewell, T., "Java Web Services", Editora O'Reilly, 1a. Edição, Março, 2002.

[4] Conti, M., Kumar, M., Das, S. K., Shirazi, B. A., "Quality of Service Issues in Internet Web Services", IEEE Transactions on Computers, vol. 51, no. 6, Junho, 2002.

[5] Dikes, S., Robbins, A. K., Jeffery, L.C., "An Empirical Evaluation of Client-side Server Selection Algorithms", IEEE INFOCOM 2000, The Conference on Computer Communications, Tel Aviv, Israel, Março, 2000.

[6] Gettys, J., Berners-Lee, T., Nielsen, H. F., "Replication and Caching Position Statement", http://www.w3.org/Propagation/Acitivity.html, 1996.

[7] Hanna, M.K., Natajaran, N., Levine, N.B., "Evaluation of a Novel Two-Step Server Selection Metric", Proceedings of IEEE International Conference on Network Protocols, Califórnia, Estados Unidos, Novembro, 2001.

[8] Litoui, M., "Migration to Web Services – Latency and Scalability", Proceedings of the Fourth International Workshop on Web Site Evolution (WSE'02), Montreal, Canadá, Outubro, 2002.

[9] MICROSOFT, *Visual Studio .NET e Web Service*s, versão 7.0, disponível em: http://www.microsoft.com/net/ e http://msdn.microsoft.com/webservices/, 2002.

[10] Myers, A., "Performance Characteristics of Mirror Servers on The Internet", Proceedings of the Conference on Computer Communications (IEEE Infocom), Nova York, Estados Unidos, Março, 1999.

[11] Obracza, K., Silva, F., "Network Latency Metrics for Server Proximity", Proceedings of the IEEE Globecom, Califórnia, Estados Unidos, Dezembro, 2000.

[12] Papazoglou, M. P., Georgakopoulos, D., "Service-Oriented Computing: Introduction", Communication of the ACM, Volume 46, Outubro, 2003.

[13] Rastogi, K. K., "Replication of Documents in the World Wide Web", Indian Institute of Technology, Kanpur, Março, Dissertação de Mestrado, 1999.

[14] Rodriguez, P., Biersack, E.W., "Dynamic Parallel Access to Replicated Content in the Internet", IEEE/ACM Transactions on Networking, vol. 10, nº 4, Agosto, 2002.

[15] Sayal, M. O., "A Scalable and Adaptative Web Client-Server Architecture", Nortwestern University, Illinois, Estados Unidos, Junho, Tese de Doutorado, 2000.

[16] Sayal, M. O., "Selection Algorithms for Replicated Web Servers", Workshop on Internet Server Performance, SIGMETRICS, Estados Unidos, Junho, 1998.

[17] SOAP "Simple Object Access Protocol", World Wide Web Consortium (W3C), Version 1.1, Note 08 May, http://www.w3.org/TR/SOAP/, 2000.

[18] SUN, *Java Web Services Developer Pack,* versão 1.0, SUN Microsystems, disponivel em: http://java.sun.com/webservices/downloads/webservicespack.html, 2003.

[19] UDDI "Universal Description, Discovery e Integration of Web Services", UDDI.org's OASIS Technical Committees, Version 2.0, July, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv2, 2002.

[20] WSDL "Web Services Definition Language", World Wide Web Consortium (W3C), Version 1.1, Note 15 March, http://www.w3.org/TR/wsdl, 2001.

[21] Yoshikawa, C., Chun, B., Eastham, P., Vahdat, A., Anderson, T., Culler, D., "Using Smart Clients to Build Scalable Services", Proceedings of the USENIX Annual Technical Conference, Califórnia, Estados Unidos, Janeiro, 1997.