Um Estudo do Impacto de Desempenho de Dois Sistemas Genéricos de Comunicação em Grupo sobre o JGroups

Leandro Sales, Nabor C. Mendonça, Rafael Barbosa,
Jonathan D'Orleans, Fernando Trinta, Henrique Teófilo

Mestrado em Informática Aplicada, Universidade de Fortaleza
Av. Washington Soares, 1321, CEP 60811-905 Fortaleza – CE
leandro@enovar.com.br, nabor@unifor.br, bgrafael@gmail.com,
jonathan.dorleans@gmail.com, trinta@unifor.br, henriquetft@gmail.com

Resumo

Este artigo apresenta um estudo do impacto de desempenho de dois sistemas genéricos de comunicação em grupo, Hedera e jGCS, quando implementados sobre um mesmo sistema de comunicação em grupo, JGroups. O estudo comparou o desempenho dos dois sistemas genéricos, bem como do JGroups isoladamente, em um ambiente de rede local sob diferentes tamanhos de mensagens e diferentes protocolos de transporte. Os resultados obtidos mostram que há diferenças significativas no impacto causado pelos dois sistemas genéricos em relação ao desempenho do JGroups, e que essas diferenças estão fortemente relacionadas a variações no tamanho das mensagens e, em menor grau, no protocolo de transporte utilizado. Com base nesses resultados, o artigo oferece um conjunto de guias que podem auxiliar os desenvolvedores de aplicações distribuídas a avaliar se (e em que situações) vale a pena implementar comunicação em grupo utilizando esses dois sistemas genéricos.

1. Introdução

O desenvolvimento de aplicações distribuídas requer uma série de decisões de projeto quanto à arquitetura que será utilizada [8].
Algumas dessas decisões dizem respeito à forma como os participantes irão se comunicar, e envolvem questões como: qual protocolo de comunicação em grupo utilizar; qual nível de garantia esperado do protocolo; e qual semântica de entrega o protocolo escolhido deve oferecer [7].

Apesar da sua importância, o desenvolvimento de um protocolo de comunicação em grupo é uma tarefa complexa e de custo elevado. Visando suprir essa necessidade, vários sistemas de comunicação em grupo foram desenvolvidos nos últimos anos (por exemplo, JGroups [4], Spread [2], Appia [15], e NeEM [16]), de modo que o único desafio para o desenvolvedor seria reutilizá-los no código de suas aplicações. Embora essa diversidade de sistemas aumente o leque de opções à disposição dos desenvolvedores, ela também traz um novo problema: qual sistema de comunicação em grupo utilizar? A escolha de um sistema de comunicação em grupo é dificultada, principalmente, porque esses sistemas oferecem diferentes interfaces de programação, diferentes níveis de garantia, diferentes semânticas de entrega, entre outras diferenças, fazendo com que a implementação da aplicação fique fortemente acoplada às características do sistema escolhido [5]. Esse nível de acoplamento é indesejável, pois desencoraja os desenvolvedores a experimentarem novos protocolos de comunicação em suas aplicações, além de trazer grandes prejuízos no caso do sistema escolhido ser descontinuado.

Uma alternativa interessante à escolha de um sistema de comunicação em grupo específico é a utilização de sistemas genéricos, como Hedera [10], jGCS [5] e Shoal [18]. Esses sistemas são caracterizados por oferecerem uma interface de programação de alto nível e um mecanismo de *plug-in* que permite facilmente acoplá-los a diversos sistemas de comunicação em grupo existentes. O desenvolvimento e a utilização de sistemas genéricos de comunicação em grupo são motivados por, pelo menos, duas importantes razões. A primeira delas é o fato de que esses sistemas provêm uma semântica mínima necessária para garantir a portabilidade das

aplicações para diferentes implementações, onde o desenvolvedor pode escolher um sistema específico para um determinado cenário, substituindo um sistema por outro sem necessidade de refatorar o código da aplicação. Outra razão, sob o ponto de vista da Engenharia de Software, é o uso de uma arquitetura fracamente acoplada, evitando que uma má decisão de projeto (por exemplo, a escolha de um sistema que apresente um desempenho insatisfatório ou cujo desenvolvimento seja posteriormente descontinuado) possa vir a afetar a utilização ou manutenção da aplicação no futuro.

No que diz respeito ao desempenho de sistemas genéricos de comunicação em grupo, há ainda uma série de questões em aberto ou que não foram tratadas de forma adequada na literatura. Por exemplo, qual o impacto que esses sistemas impõem sobre o desempenho dos sistemas comunicação em grupo que eles encapsulam? Até que ponto fatores como o tamanho das mensagens e o protocolo de transporte utilizado influenciam esse impacto? Claramente, ter acesso a esse tipo de informação seria de extrema valia para os desenvolvedores de aplicações distribuídas, que poderiam decidir com mais segurança sobre quando utilizar um sistema específico ou quando optar por um sistema genérico.

Para tentar responder às questões acima, este artigo apresenta um estudo do impacto causado por dois sistemas genéricos de comunicação em grupo atualmente existentes, Hedera e jGCS, quando estes estão acoplados ao JGroups, um sistema de comunicação em grupo amplamente conhecido e utilizado [12]. O estudo comparou o desempenho dos dois sistemas genéricos, bem como do JGroups isoladamente, em um ambiente de rede local sob diferentes tamanhos de mensagens e diferentes protocolos de transporte. Os resultados obtidos mostram que há diferenças significativas no impacto causado pelos dois sistemas em relação ao desempenho do JGroups. Além disso, observou-se que essas diferenças estão fortemente relacionadas a variações no tamanho das mensagens e, em menor grau, no protocolo de transporte utilizado. Esses resultados sugerem que o desempenho de alguns sistemas genéricos de comunicação em grupo não pode ser definido *a priori*, e que a decisão sobre se (e quando) vale a pena adotá-los depende da análise de fatores como o tamanho das mensagens transmitidas pela aplicação e o protocolo de transporte utilizado pelo sistema de comunicação em grupo subjacente.

O restante do artigo está organizado da seguinte maneira. A Seção 2 dá uma visão geral dos três sistemas de comunicação em grupo investigados. A Seção 3 descreve a metodologia utilizada no estudo, cujos resultados são apresentados na Seção 4. A Seção 5 oferece guias que podem auxiliar os desenvolvedores a decidir quando utilizar cada um dos dois sistemas genéricos estudados. A Seção 6 cobre trabalhos relacionados. Por fim, a Seção 7 conclui o artigo e sugere tópicos para trabalhos futuros.

2. Sistemas de Comunicação em Grupo Investigados

2.1. JGroups

JGroups é um sistema de comunicação em grupo de código aberto totalmente escrito em Java [4]. Ele oferece uma abstração alto nível, chamada *Channel*, que funciona como um *socket* de comunicação em grupo através do qual as aplicações podem enviar e receber mensagens para/de um grupo de processos. Essa abstração torna a pilha de protocolos do JGroups totalmente transparente para o desenvolvedor da aplicação, o que torna possível reusar o mesmo código da aplicação para diferentes cenários e configurações de rede, bastando, para isso, modificar a configuração da pilha de protocolos subjacente.

Uma importante característica do JGroups é que ele permite ao desenvolvedor definir sua própria pilha de protocolos, a partir de um conjunto de componentes que oferecem serviços como transporte, confiabilidade, ordenação, fragmentação, gerenciamento de membros, etc. Em particular, a pilha de protocolos do JGroups pode utilizar diferentes protocolos de transporte de mensagens em nível de rede, como UDP e TCP, além de outros que possam ser definidos utilizando um adaptador de protocolo próprio. Estas e inúmeras outras funcionalidades tornaram o JGroups um dos mais conhecidos e utilizados sistemas de comunicação em grupo existentes, tendo sido adotado como parte da solução de replicação de importantes projetos de código aberto, como Tomcat [19], JOnAS [14], JBoss [11] e C-JDBC [6]. Uma lista mais abrangente de projetos que utilizam o JGroups pode ser encontrada em [12].

A versão do JGroups utilizado no estudo foi a 2.6 GA, de 12 de novembro de 2007.

2.2. Hedera

O Hedera é um arcabouço de código aberto, escrito em Java, que oferece uma interface de programação uniforme para diferentes sistemas de comunicação em grupo [10]. Embora possa ser utilizado em diferentes cenários e configurações de rede, seu projeto teve como alvo implementar comunicação em grupo no

contexto de *clusters* de computadores, onde os grupos costumam apresentar um número moderado e mais estável de membros.

O Hedera é a solução de comunicação em grupo utilizada pelo sistema Sequóia [17], um *middleware* para replicação transparente de servidores de banco de dados, desenvolvido como continuação do projeto C-JDBC.

A versão do Hedera utilizado no estudo foi a 1.5.5, de 13 de Dezembro de 2006. Essa versão inclui *plug-ins* para os seguintes sistemas: JGroups, Appia e Spread (esse último em versão beta).

2.3. jGCS

O jGCS é outro sistema escrito em Java que oferece uma interface genérica para comunicação em grupo [5]. Esta interface pode ser usada por aplicações que necessitem de diferentes primitivas de comunicação, de um simples serviços de difusão sobre IP Multicast a um completo serviço de difusão atômica confiável.

O jGCS implementa um novo conceito de serviço de comunicação em grupo, onde o padrão inversão de controle é utilizado para separar aspectos de configuração do uso do serviço. Dessa forma, o sistema melhora a modularidade do código, uma vez que as aplicações usam uma interface comum que pode ser implementada utilizando diferentes soluções de comunicação em grupo. A solução que será utilizada pela aplicação é definida em tempo de configuração.

O jGCS foi desenvolvido e é utilizado como solução de comunicação em grupo no contexto do projeto GORDA [9], que visa a propor soluções para a replicação transparente e em larga escala de sistemas de banco de dados.

A versão do jGCS utilizada no estudo foi a 0.6.1, de 29 de Outubro de 2007. Essa versão inclui *plug-ins* para os seguintes sistemas: JGroups, Appia, Spread, NeEM, além de uma implementação de referência sobre IP Multicast.

3. Metodologia de Avaliação

3.1. Objetivos e Pré-suposições

Conforme mencionado anteriormente, o foco do nosso trabalho está em avaliar o impacto de desempenho dos dois sistemas genéricos, Hedera e jGCS, quando utilizados como uma camada superior ao JGroups. A partir dos resultados da avaliação, esperamos obter informações suficientes para mostrar quando vale a pena utilizar essas camadas genéricas e em quais situações é mais interessante usar o JGroups isoladamente.

Inicialmente, nossa hipótese em relação ao desempenho dos dois sistemas genéricos era de que ambos afetariam negativamente o desempenho do JGroups, justamente devido à camada extra que cada um deles implementa. Nesse caso, as diferenças de desempenho observadas seriam determinadas pela forma com que cada camada é implementada. Essas diferenças poderiam existir por vários motivos, como chamadas desnecessárias de métodos, cópias desnecessárias de objetos, ou alguma má decisão de projeto que afetasse negativamente o tempo de execução da camada extra implementada pelo sistema genérico.

3.2. Ambiente de Teste

Para realizarmos os testes de desempenho, procuramos simular um cenário típico de utilização dos JGroups, onde pudéssemos obter resultados mais próximos àqueles obtidos em ambientes reais. Como o JGroups é muito utilizado no contexto de servidores de aplicação replicados em um *cluster* de computadores, utilizamos uma rede local dedicada, partindo do pressuposto de que servidores replicados precisam de redes dedicadas para um maior desempenho.

Os computadores utilizados nos experimentos tinham a seguinte configuração: Sistema Operacional Windows XP Professional (SP2); processador Intel Pentium 4 (3.00 GHz); e memória RAM (DDR2) de 2 GB. A rede local utilizada foi do tipo Fast Ethernet de 10/100 Mbps.

Todos os testes foram realizados utilizando a máquina virtual Java da SUN (versão 1.6.0_03), configurada com os mesmos parâmetros de execução utilizada pelos desenvolvedores do JGroups em seus mais recentes testes de desempenho [13].

3.3. Metodologia

Realizamos os testes de avaliação com um grupo de 4 computadores, comunicando-se de forma *n-n*. Testes realizados com grupos de 2 e 6 computadores apresentaram resultados semelhantes, e, por questão de espaço, não serão descritos neste artigo. Para grupos com 8 membros ou mais, a sobrecarga imposta pelo JGroups começa a se aproximar do nível de saturação da rede, comprometendo fortemente o seu desempenho. Uma análise minuciosa do desempenho e da escalabilidade do JGroups pode ser encontrada em [13].

Durante os testes, cada um dos membros do grupo enviava 10.000 mensagens para os outros membros, perfazendo um total de 40.000 mensagens enviadas pela rede por teste. Essa quantidade de mensagens foi definida buscando estressar a rede e analisar o impacto que isso causaria no desempenho de cada sistema. Os tamanhos das mensagens enviadas em cada teste foram 1 B, 10 B, 100 B, 1 KB e 10 KB. Essa variação procurou cobrir um amplo espectro de tamanhos de mensagens, desde mensagens muito pequenas (da ordem de alguns poucos *bytes*) até mensagens relativamente grandes (da ordem de alguns milhares de *bytes*). Um faixa similar de tamanhos de mensagens foi utilizada em uma avaliação anterior do JGroups no contexto de servidores J2EE replicados [1].

Além disso, executamos os testes para duas diferentes pilhas de protocolos do JGroups:

- Pilha UDP – utiliza IP Multicast na camada de transporte e FC (*Flow Control*) como protocolo de controle de fluxo.
- Pilha TCP – utiliza múltiplas conexões TCP na camada de transporte, com o protocolo de controle de fluxo sendo implementado em nível de rede pelo próprio TCP.

Por fim, estabelecemos a *taxa de entrega* como o parâmetro a partir do qual iríamos comparar o desempenho dos três sistemas investigados. Nos experimentos, esse parâmetro foi calculado pela média do número de mensagens entregues por segundo em cada um dos quatro nós da rede.

Cada teste foi executado 10 vezes, com cada parâmetro de avaliação sendo calculado em função dos valores médios, mínimos e máximos observados.

4. Resultados

Nos gráficos mostrados a seguir, o eixo das abscissas representa o tamanho das mensagens em *bytes*. As barras nos gráficos representam os resultados observados nos testes; no caso de resultados obtidos a partir do cálculo da média entre um conjunto de valores, as barras representam os valores médios, com os valores máximos e mínimos representados por indicadores sobrepostos às suas respectivas barras. Pelos valores médios, mínimos e máximos obtidos, pode-se observar que os dados coletados foram relativamente estáveis.

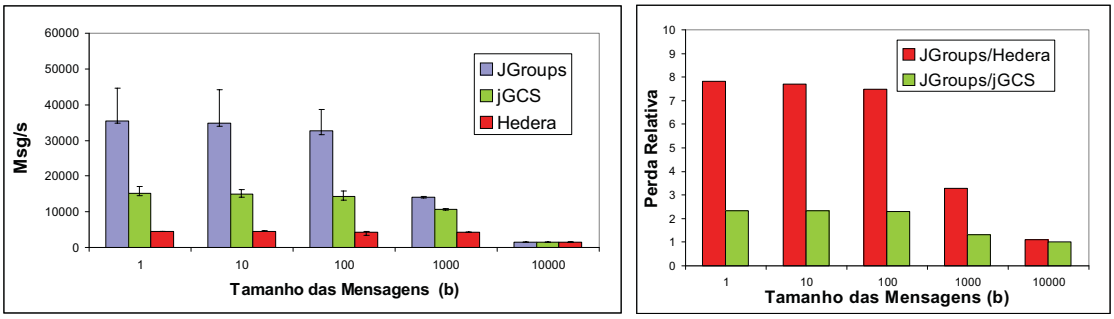
4.1. Resultados para a Pilha UDP

A Figura 1(a) mostra a taxa de entrega média obtida para os três sistemas, utilizando a configuração do JGroups com a pilha UDP. Como podemos observar, para mensagens de tamanho até 1 KB, tanto o jGCS como o Hedera causam um impacto significativo sobre o desempenho do JGroups, com o Hedera oferecendo o pior desempenho. Observa-se, também, que esse impacto vai diminuindo progressivamente para os dois sistemas à medida que o tamanho das mensagens aumenta. No caso de mensagens de tamanho 10 KB, observa-se que o impacto causado por ambos os sistemas é praticamente imperceptível. Atribuímos esse resultado ao fato de que, para mensagens maiores, o custo associado ao aumento do tráfego gerado na rede ultrapassa o custo decorrente da camada extra implementada por cada sistema genérico, com o primeiro ficando cada vez mais predominante no cálculo da taxa de entrega.

Para enfatizar as diferenças nos níveis de impacto causados pelo jGCS e pelo Hedera, a Figura 1(b) mostra a perda relativa imposta por cada um desses dois sistemas em relação ao desempenho do JGroups. Nessa figura, o eixo das ordenadas representa a razão entre o desempenho (taxa de entrega) medido para o JGroups, quando utilizado isoladamente, e o desempenho correspondente observado para cada sistema genérico.

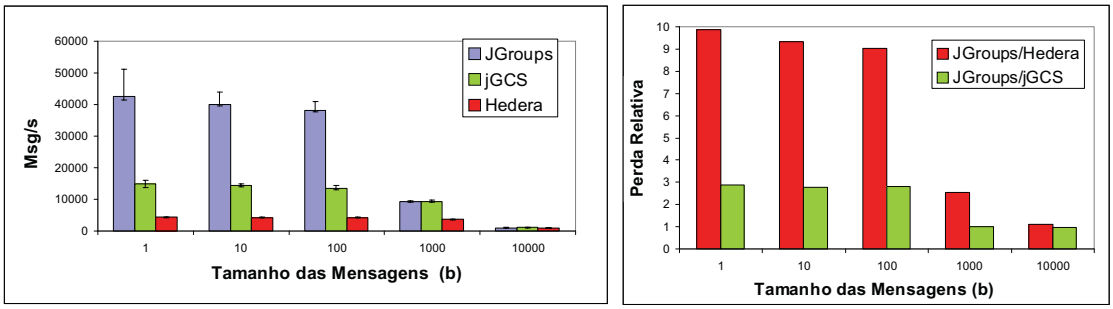
A partir do gráfico mostrado na Figura 1(b), fica claro que há uma grande diferença no nível de impacto causado por cada sistema genérico sobre o desempenho do JGroups, e que essa diferença é mais significativa para mensagens menores. Mais especificamente, para mensagens de até 100 B, o desempenho obtido com o Hedera é cerca de 7 a 8 vezes inferior ao desempenho obtido com JGroups sozinho. No caso do jGCS, a perda fica em torno de 2,3 vezes para os mesmos tamanhos de mensagem. Isso mostra um nível de degradação significativamente alto do Hedera em comparação ao jGCS.

Essa grande diferença de desempenho observada entre o jGCS e o Hedera atraiu nossa atenção, nos levando a investigar se o nível de impacto imposto pelo Hedera poderia ser de fato atribuído exclusivamente ao processamento adicional realizado na sua camada extra de código. Mais adiante, na Seção 4.4, veremos que essa diferença ocorre devido ao fato de que o Hedera adiciona uma quantidade substancial de dados de controle ao conteúdo das mensagens transmitidas pela aplicação, proporcionando, assim, um maior nível de congestionamento da rede.



(a) (b)

Fig. 1. Taxa de entrega (a) e perda relativa (b) para a pilha UDP.



(a) (b)

Fig. 2. Taxa de entrega (a) e perda relativa (b) para a pilha TCP.

4.2. Resultados para a Pilha TCP

A Figura 2(a) mostra os resultados referentes à taxa de entrega para os três sistemas, quando configurados com a pilha TCP do JGroups. Como podemos observar, os resultados se mantêm parecidos com os apresentados para a configuração que utiliza a pilha UDP (Figura 1(a)), com os melhores desempenhos sendo oferecidos pelo JGroups sozinho, jGCS e Hedera, nessa ordem. Note, também, que as diferenças entre os sistemas são mais acentuadas para mensagens menores (até 100 B), ficando praticamente imperceptíveis com mensagens de tamanho 10 KB.

Da mesma forma que fizemos para os resultados da pilha UDP, a Figura 2(b) ilustra a perda relativa do Hedera e do jGCS em relação ao desempenho do JGroups, quando configurado com a pilha TCP. Mais uma vez, fica evidente a superioridade do jGCS sobre o Hedera. A diferença a ser destacada, dessa vez, é que a perda apresentada pelos dois sistemas com a pilha TCP é nitidamente superior às perdas obtidas com os mesmo sistemas quando configurados para utilizarem a pilha UDP (Figura 1(b)).

No caso do Hedera, a perda varia entre 9 e 10 vezes para mensagens de até 100 B, enquanto que o nível de perda do jGCS, para a mesma faixa de tamanhos de mensagem, fica em torno de 2,9 vezes.

4.3. Tamanho das Mensagens vs. Quantidade de Dados Transmitida

Como mencionado anteriormente, a significativa diferença de desempenho observada entre o jGCS e o Hedera nos levou a questionar a nossa hipótese inicial, de que as diferenças de impacto entre esses dois sistemas se dariam apenas por diferentes estratégias de implementação utilizadas em suas respectivas camadas.

Outra hipótese levantada foi a de que o Hedera, de algum modo, poderia estar gerando um tráfego de rede substancialmente maior do que o jGCS, para os mesmos tamanhos de mensagem. Para investigar essa hipótese, nós medimos o tamanho real de cada mensagem transmitida através da camada de transporte por cada sistema, e então comparamos esse tamanho com o tamanho das mensagens que eram enviadas na camada de aplicação. A Tabela 1 apresenta, para cada sistema, os tamanhos das mensagens enviadas na camada de aplicação e a quantidade de dados que é de fato transmitida para cada mensagem pela camada de transporte.

Tamanho da Mensagem Enviada (b)	Quantidade de Dados Transmitida (b)			
	JGroups	jGCS	Hedera	
	UDP/TCP	UDP/TCP	UDP	TCP
1	10	10	1095	1101
10	19	19	1104	1110
100	109	109	1194	1200
1000	1009	1009	2094	2100
10000	10009	10009	11094	11100

Tabela 1. Comparativo entre o tamanho das mensagens enviadas pela aplicação e a quantidade de dados transmitida pela camada de transporte

Analisando os dados da Tabela 1, percebemos que, enquanto o JGroups e o jGCS adicionam apenas 9 *bytes* ao conteúdo das mensagens enviadas pela aplicação, independentemente do protocolo de transporte utilizado, o Hedera adiciona surpreendentes 1094 e 1100 *bytes* por mensagem para as pilhas UDP e TCP, respectivamente. A partir de uma inspeção de seu código fonte, foi constado que o Hedera sempre embute nas mensagens, entre outras informações de controle, a lista de membros do grupo para os quais a mensagem se destina. No caso dos experimentos realizados, quando a pilha TCP é utilizada, 912 desses 1100 *bytes* correspondem ao cabeçalho fixo da mensagem e 188 *bytes* correspondem à quantidade necessária para representar os identificadores dos quatro membros do grupo (47 *bytes* por identificador). Quando é utilizada a pilha UDP, o tamanho do cabeçalho enviado é um pouco menor, 906 *bytes*, enquanto a quantidade de *bytes* enviados referente aos identificadores dos membros do grupo continua a mesma. Claramente, essas informações sobre os membros do grupo são enviadas de forma redundante pelo Hedera, quando este está acoplado ao JGroups, uma vez que a lista de membros já faz parte do conjunto de atributos do objeto *Channel* associado a cada grupo do JGroups.

Sabendo que o Hedera adiciona mais de 1000 *bytes* à mensagem originalmente enviada pela aplicação, voltamos a analisar os seus resultados referentes a taxa de entrega. A partir dessa nova análise, ficou evidente que esses dados extras estavam sendo responsáveis pela alta degradação de desempenho propiciada pelo Hedera. Isto ocorre porque, mesmo que aplicação utilize o Hedera para enviar mensagens de alguns poucos *bytes*, ele sempre transmite mensagens a partir de um tamanho cujo impacto no tráfego gerado na rede já começa a exercer uma forte influência sobre o seu desempenho. Já no caso do jGCS, acreditamos que seu nível de perda (sempre inferior a 3 vezes o desempenho do JGroups) esteja, pelo menos em princípio, de acordo com a hipótese inicialmente levantada, uma vez que o tamanho das mensagens que ele transporta é idêntico ao tamanho das mensagens que são transportadas pelo JGroups, quando utilizado isoladamente.

5. Discussão

Com base nos resultados apresentados na seção anterior, esta seção discute se (e sobre quais circunstâncias) valeria a pena utilizar sistemas genéricos, como Hedera e jGCS, para implementar comunicação em grupo em aplicações distribuídas. A idéia é auxiliar os desenvolvedores de aplicações a decidir sobre quando usar um sistema de comunicação em grupo isoladamente, ou quando esconder seu uso sob a interface de programação de alto nível oferecida por algum sistema genérico.

Inicialmente, deve-se considerar se o fraco acoplamento da arquitetura da aplicação em relação à utilização de um sistema de comunicação em grupo específico é um requisito primordial de projeto. Isto porque sistemas genéricos sempre impõem níveis extras de indireção entre a aplicação e o sistema de comunicação em grupo escolhido, fazendo com que perdas de desempenho sejam inevitáveis.

Outros fatores importantes a serem considerados, como demonstrados nos experimentos descritos na seção anterior, são o tamanho das mensagens enviadas ou recebidas pela aplicação, e o protocolo de rede utilizado para o transporte das mensagens. Tomando-se como base os cenários investigados, se o desempenho for o requisito mais importante da aplicação, para mensagens de até 100 B o desenvolvedor deverá optar por utilizar diretamente o JGroups, tendo o TCP como protocolo de transporte. Caso o fraco acoplamento da aplicação esteja entre seus requisitos primordiais, a escolha deverá ser pelo jGCS configurado com pilha de protocolos UDP do

JGroups, que se mostrou bem mais eficiente do que o Hedera para essa faixa de tamanhos de mensagens.

Para mensagens entre 1 KB e 10 KB, a sugestão continua sendo o uso do JGroups sozinho, caso o objetivo maior seja priorizar o desempenho da aplicação, mas com uma diferença: o protocolo de transporte a ser utilizado deverá ser o UDP. Caso seja vontade do desenvolvedor utilizar uma camada genérica, a escolha ainda dever ser pelo jGCS configurado com a pilha UDP do JGroups.

Para mensagens maiores que 10 KB, os experimentos mostram que o custo de transmissão da rede se torna predominante em relação a qualquer outro custo imposto pelo sistema de comunicação em grupo utilizado, sugerindo que, nesses casos, a escolha por um determinado sistema seja feita de acordo com as necessidades da aplicação, com base nas características e serviços oferecidos por cada sistema. Para essa faixa de tamanhos de mensagens, o protocolo de transporte indicado continua sendo o UDP, que apresentou um impacto menor que o TCP para mensagens maiores.

É importante ressaltar que as considerações acima estão restritas aos cenários investigados nos experimentos. Obviamente mais experimentos serão necessários, cobrindo um maior número de cenários de teste, para que esses resultados possam ser generalizados.

6. Trabalhos Relacionados

Por ser um sistema de comunicação em grupo amplamente utilizado, o JGroups já teve seu desempenho avaliado em diversos trabalhos anteriores. Em [1], os autores compararam o desempenho de diferentes pilhas de protocolo do JGroups em um ambiente de teste que simulava o contexto de clusters de servidores de aplicação J2EE. Outros experimentos de avaliação do desempenho do JGroups, com características similares aos experimentos realizados por Abdellatif *et al.*, são descritos em [13]. Em contraste a esses trabalhos, nosso estudo não teve como objetivo principal comparar o desempenho de diferentes configurações do JGroups, mas sim o impacto que diferentes sistemas genéricos causam no desempenho do JGroups quando estão acoplados a ele.

Outra avaliação envolvendo diferentes sistemas de comunicação em grupo escritos em Java foi realizada por Baldoni *et al.* [3]. No entanto, o foco dos autores não estava em mostrar o impacto causado por diferentes sistemas genéricos sobre um sistema de comunicação em grupo específico, mas apenas em comparar o desempenho de cada sistema (nenhum deles genérico) em diferentes cenários e configurações e rede.

7. Conclusão

Este trabalho apresentou os resultados de um estudo do impacto de desempenho de dois sistemas genéricos de comunicação em grupo, Hedera e JGCS, com ambos os sistemas configurados para utilizarem o serviços de um sistema de comunicação em grupo específico, JGroups, em um ambiente de rede local. Em linhas gerais, os resultados mostram que a diferença entre o impacto causado por cada sistema genérico no desempenho do JGroups é significativa, com o Hedera chegando a ser até 11 vezes mais lento que o JGroups no pior caso, e que essa diferença pode ser afetada pelo tamanho das mensagens enviadas pela aplicação e, em menor escala, pelo protocolo de transporte utilizado na configuração do JGroups.

Duas importantes linhas de pesquisa estão sendo consideradas como tópicos para trabalhos futuros. A primeira consiste em repetir a mesma série de experimentos para outros sistemas de comunicação em grupo também bastante utilizados, como Appia [15] e Spread [2]. A idéia é poder verificar até que ponto os resultados observados para o JGroups podem ser generalizados para esses outros sistemas. A segunda linha de pesquisa visa a realizar novos experimentos com o JGroups, jGCS e Hedera, mas agora no contexto da arquitetura de replicação de um dos servidores de aplicação que utilizam o JGroups como solução de comunicação em grupo, como JOnAS [14] ou JBoss [11]. A idéia aqui é poder verificar se as mesmas diferenças podem ser observadas nesse novo contexto, onde o tamanho das mensagens enviadas pelos membros do grupo vai depender do tamanho do estado de cada sessão ou objeto replicado gerenciado pelo servidor de aplicação.

8. Referências

[1] Abdellatif, T., Cecchet, E. e Lachaize, R. (2004) "Evaluation of a Group Communication Middleware for Clustered J2EE Application Servers", In Proc. of the 6th International Symposium on Distributed Objects and Applications (DOA'04), Lecture Notes in Computer Science Vol. 3291, Springer.

[2] Amir, Y., Danilov, C. e Stanton, J. (2000) "A Low Latency, Loss Tolerant Architecture and Protocol for Wide Area Group Communication", In Proc. of the IEEE International Conference on Dependable Systems and Networks (ICDSN'00), IEEE CS Press.

[3] Baldoni, R., Cimmino, S., Marchetti, C. e Termini, A. (2002) "Performance Analysis of Java Group Toolkits: a Case Study", In Proc. of the International Workshop on Scientific Engineering for Distributed Java Applications (FIDJI'02), Lecture Notes in Computer Science Vol. 2604, Springer, pp. 81-90.

[4] Ban, B. (1998) "Design and Implementation of a Reliable Ggroup Communication Sistema for Java", Cornell University, Disponível em http://www.jgroups.org/javagroupsnew/docs/papers/Coots.ps.gz.

[5] Carvalho, N., Pereira, J. e Rodrigues, L. (2006), "Towards a Generic Group Communication Service", In Proc. of the 8th International Symposium on Distributed Objects and Applications (DOA'06), Lecture Notes in Computer Science Vol. 4276, Springer.

[6] C-JDBC (2007), "C-JDBC: Clustered JDBC", Disponível em http://c-jdbc.objectweb.org.

[7] Chockler, G. V., Keidar, I. e Vitenberg, R. (2001), "Group Communication Specifications: a Comprehensive Study", ACM Computing Surveys, 33(4):427–469.

[8] Coulouris, G., Dollimore, J. e Kindberg, T. (2005), Distributed Systems: Concepts and Design, Addison Wesley, Fourth Edition.

[9] GORDA (2007), "GORDA – Open Replication of Databases", Disponível em http://gorda.di.uminho.pt/.

[10] Hedera (2007), "Hedera Project", Disponível em http://hedera.continuent.org.

[11] JBoss (2007) "JBoss Application Server", Disponível em http://labs.jboss.com/jbossas/.

[12] JGroups (2007a) "JGroups Success Stories", Disponível em http://www.jgroups.org/javagroupsnew/docs/success.html.

[13] JGroups (2007b) "JGroups Performance", Disponível em http://www.jgroups.org/javagroupsnew/perfnew/Report.html.

[14] JOnAS (2007) "Java Open Application Server", Disponível em http://jonas.objectweb.org.

[15] Miranda, H., Pinto, A. e Rodrigues, L. (2001) "Appia: a Flexible Protocol Kernel Supporting Multiple Coordinated Channels", In Proc. of the 21st International Conference on Distributed Computing Systems (ICDCS'01), IEEE CS Press, pp. 707-710.

[16] Pereira, J., Rodrigues, L., Monteiro, M. J., Oliveira, R. e Kermarrec, A.-M. (2003) "NeEM: Network-friendly Epidemic Multicast", In Proc. of the 22nd IEEE Symposium on Reliable Distributed Systems (SRDS'03), IEEE CS Press, pp. 15-24.

[17] Sequóia (2007), "Sequóia Project", Disponível em http://sequoia.continuent.org/HomePage.

[18] Shoal (2007), "Shoal – A Dynamic Java Clustering Framework", Disponível em https://shoal.dev.java.net/.

[19] Tomcat (2007), "Apache Tomcat", Disponível em http://tomcat.apache.org/.