Avaliação de Plataformas de Suporte à Composição de Protocolos para Sistemas de Replicação de Bases de Dados*

Alexandre Pinto
Universidade de Lisboa
apinto@di.fc.ul.pt

Novembro de 2003

*Trabalho suportado pela FCT através do projecto POSI/CHS/41285/2001, StrongRep.

Conteúdo

1 Introdução

Este relatório apresenta uma avaliação do desempenho de plataformas de suporte à composição de protocolos de modo a aferir a sua adequação ao suporte de sistemas de replicação de bases de dados. Este trabalho é feito no contexto do project FCT StrongRep (POSI/CHS/41285/2001).

Qualquer avaliação da concretização de um protocolo sobre uma plataforma pode ser feita sobre dois prismas: um qualitativo, em que se avaliam as vantagens e desvantagens conceptuais que influenciaram a forma como o protocolo foi concretizado e que auxiliaram, ou impediaram, que essa concretização fosse simples e elegante; e sobre um prisma quantitativo, em que se avalia se a plataforma com o protocolo apresenta um resultado em termos de desempenho vantajoso.

Neste relatório serão apresentadas primeiro as avaliações qualitativas das duas plataformas usadas, o Ensemble[5] e o Appia[10][9], para a concretização do *serviço de Grupos Ligeiros*[12][4][11]. Depois serão apresentados os resultados da avaliação quantitativa das duas concretizações. O relatório termina com uma análise de avaliações efectuadas por terceiros, que incidiam sobre a plataforma Appia.

2 Vantagens e desvantagens funcionais

O Ensemble revelou-se um sistema robusto e com um vasto conjunto de protocolos prontos a utilizar. A sua principal vantagem é, no entanto, a sua eficiência com resultados de desempenho muito bons, apesar de utilizar um reciclador automático de memória. Este revelou-se uma faca de dois gumes. Por um lado significou uma velocidade de desenvolvimento inicial relativamente alta, por outro, é necessário um enorme controlo e cuidado por parte do programador, para não deixar que o seu funcionamento afecte de forma muito negativa o desempenho obtido. Estes cuidados têm duas vertentes: usar só as primitivas iterativas da linguagem usada e utilizar correctamente as ferramentas de gestão dos tampões correspondentes às mensagens.

No entanto, a grande desvantagem do Ensemble é o facto de suportar unicamente composições de protocolos estritamente verticais, ou seja, em pilha sem possibilidade de partilha de camadas entre pilhas. Esta limitação é justificada pelas ferramentas de validação formal e optimização da execução, que só suportam essa composição. Embora estas ferramentas tenham bastante valor, as restrições ao nível da composição de protocolos não deixam de ser uma limitação que deveria ser abordada. Esta limitação obrigou a que a concretização efectuada do serviço de Grupos Ligeiros no Ensemble não fosse a melhor em termos do seu posicionamento no sistema. Isto por seu turno limitou as capacidades ao dispor do mesmo, sem acesso a vários eventos só disponíveis às camadas dentro da pilha.

O posicionamento provocou igualmente a perda da flexibilidade na composição dos vários *grupos ligeiros* projectados num *grupo pesado*. Na concretização efectuada, todos os *grupos ligeiros* têm que partilhar exactamente a mesma pilha, ou seja, com as mesmas propriedades que o *grupo pesado*. É impossível ter *grupos ligeiros* com diferentes propriedades projectados no mesmo *grupo pesado*, reduzindo as possibilidades de encontrar a projecção mais correcta entre *grupos ligeiros* e *grupos pesados*.

Embora na concretização efectuada esse problema não se tenha revelado, a experiência com o serviço de Grupos Ligeiros no Appia revelou que a capacidade de adicionar, de forma simples e dinâmica, novos eventos é muito útil. Só que no Ensemble os eventos disponíveis são fixos (cerca de 40) e adicionar um novo implica recompilar todo o sistema. Mesmo depois disso, podem existir problemas devido ao complexo processamento que estes sofrem por parte das diversas camadas.

Uma outra problema encontrada prende-se com a incapacidade de entregar uma nova vista mantendo a impossibilidade de enviar mensagens, ou seja, mantendo o grupo bloqueado. Esta limitação prende-se mais com a utilização normal da *sincronia na vista*, que assume que quando uma vista é entregue o grupo fica desbloqueado. No entanto, esta limitação do Ensemble não permitiu a utilização do protocolo de *esvaziamento por software*. Este permitia que fosse correctamente instalada uma nova vista do *grupo ligeiro* sem obrigar a uma mudança de vista do *grupo pesado*. Como isto não é possível, sempre que é necessário mudar uma vista de um *grupo ligeiro* é preciso efectuar uma mudança de vista do *grupo pesado*, para garantir a correcção da mudança, o que acarreta um enorme custo em termos de interferência entre os *grupos ligeiros* projectados, que como o *grupo pesado* mudou de vista também têm que mudar de vista. A mudança de vista de um *grupo ligeiro* obriga todos os outros a mudar de vista também.

Outro aspecto é a incapacidade do interface da aplicação do Ensemble de forçar a instalação de uma nova vista com um conjunto de novos membros adicionados, o que origina que quando um *grupo ligeiro* muda de *grupo pesado*, potencialmente cada membro do *grupo ligeiro* terá que se juntar ao novo *grupo pesado* separadamente, o que provocará um número de mudanças de vista igual ao número de membros do *grupo ligeiro*. Isto representa um alto nível de interferência. Embora o Ensemble tente agrupar a junção destes novos membros numa única mudança de vista, isso não é imposto, pelo que na prática essas várias mudanças de vista do *grupo pesado* acontecem.

Já o Appia correspondeu genericamente ao esperado, excepto num grande factor, o desempenho. Este é algo deficiente, em parte devido à linguagem usada na sua concretização, o JAVA. Mas esta não justifica tudo e outras avaliações (apresentadas na Secção 4) revelaram que existe igualmente necessidade de maior estudo da concretização da plataforma e seus protocolos.

De resto, o Appia revelou alguns dos seus pontos fortes, como a fácil habituação e desenvolvimento de novos protocolos, já demonstrados na sequência de anteriores usos da ferramenta no ensino do desenvolvimento de protocolos para sistemas distribuídos.

O posicionamento do serviço concretizado no Appia é o ideal, funcionando como um encaminhador entre os "semi-canais" dos *grupos ligeiros* e dos *grupos pesados*. Desta forma o serviço tem acesso a todos os eventos das camadas de suporte à Comunicação em Grupo, permitindo o acesso a mecanismos inacessíveis à concretização no Ensemble, garantindo assim uma maior capacidade e transparência de operação. Com o posicionamento utilizado ganha-se igualmente uma grande flexibilidade na composição dos *grupos ligeiros*, permitindo que *grupos ligeiros* com propriedades diferentes mas filiação semelhantes possam ser projectados no mesmo *grupo pesado*. A diferença reside unicamente no "semi-canal" do *grupo ligeiro*. A flexibilidade extra advém da própria plataforma e não requer quaisquer mecanismos específicos por parte do serviço de

	k	m
latência	1	0
largura de banda	1	*grande*
débito	*grande*	0

Tabela 1: Relação entre parâmetros k e m no teste *ring* da aplicação Perf.

Grupos Ligeiros.

Um aspecto a considerar em futuros melhoramentos do Appia seria a capacidade de mudar a constituição do canal, em termos das sessões que o constituem, em tempo de execução. Esta capacidade permitiria que os "semi-canais" dos *grupos ligeiros* fossem substituídos por canais completos, em que sempre que o *grupo ligeiro* mudava de *grupo pesado* eram mudadas as sessões correspondentes a este. Com isto, deixava de ser necessário efectuar o encaminhamento no serviço de Grupos Ligeiros, poupando-se a mudança de estado que a mudança de canal implica. O serviço de Grupos Ligeiros assumiria um papel meramente monitorizador e de gestão dos diversos canais. No Ensemble já existe a capacidade de mudar as camadas em tempo de execução, mas devido ao posicionamento que o serviço de Grupos Ligeiros teve que tomar não foi possível tirar partido dessa capacidade.

3 Análise de desempenho

A análise de desempenho procurou demonstrar as vantagens de utilização do serviço de Grupos Ligeiros, em ambientes com vários grupos de filiação semelhante, sem com isso penalizar significativamente o desempenho noutros ambientes.

Nos testes efectuados utilizou-se a aplicação *Perf*. Esta aplicação existe tanto no Ensemble como no Appia, sendo que a concretização neste último foi baseada na do Ensemble. A aplicação oferece um conjunto de testes, dos quais o mais usado foi o "ring". Neste, numa vista com n membros, cada membro envia k mensagens seguidas, de tamanho m, ficando depois à espera de receber $(n-1)*k$ mensagens de todos os outros membros antes de voltar a enviar k mensagens. Este processo repete-se r vezes. Variando o número de mensagens enviadas de seguida e o tamanho das mesmas, obtêm-se aproximações das seguintes medidas:

latência por ronda que indica quanto tempo demora a completar uma ronda, ou seja, enviar uma mensagem para todos os membros e receber uma de cada um deles.

débito por membro que indica quantas mensagens cada membro conseguiu enviar por segundo.

largura de banda do grupo que indica quantos bytes foram trocados por segundo entre os membros de um grupo.

A Tabela 1, retirada de [6], apresenta a relação entre o número de mensagens, o tamanho das mesmas e a medida obtida.

Os testes foram efectuados em quatro PCs Pentium II a 350 MHz ligados por um *hub* Ethernet a 100Mb/s.

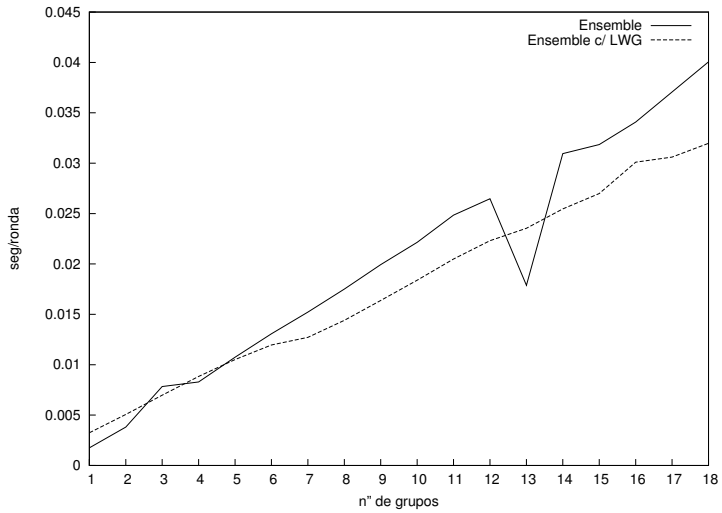

Figura 1: Latência por ronda relativamente ao número de grupos a funcionar em paralelo, no Ensemble

3.1 Grupos Ligeiros e Ensemble

A Figura 1 apresenta a latência por ronda com um número sucessivamente maior de grupos a funcionar concorrentemente. Embora o serviço de Grupos Ligeiros apresente piores resultados com poucos grupos, a partir de seis grupos os resultados apresentam uma melhoria razoável.

A Figura 2 mostra o débito por membro, por grupo. Aqui os resultados do uso dos Grupos Ligeiros não são expressivos, com resultados muito iguais aos obtidos sem o serviço de Grupos Ligeiros. Isto deve-se ao facto de os Grupos Ligeiros terem um custo fixo por mensagem. Mesmo assim os ganhos apresentados devem-se à diminuição dos recursos utilizados.

A Figura 3 mostra a largura de banda utilizada por cada grupo. Como seria de esperar quantos mais existem, menos largura de banda está disponível para cada um deles. Aqui os Grupos Ligeiros demonstram a sua utilidade, ao apresentarem resultados significativamente melhores com um número de grupos maior. Igualmente de realçar o facto de uma melhoria aparecer logo com apenas dois grupos, ao contrário dos testes anteriores em que isso só acontecia a partir dos cinco grupos a operar simultaneamente.

3.2 Grupos Ligeiros e Appia

A análise dos resultados obtidos com a concretização dos Grupos Ligeiros no Appia é semelhante à efectuada para o Ensemble.

A Figura 4 apresenta a latência por ronda com um número sucessivamente maior de grupos a funcionar concorrentemente. Tal como esperado, com poucos grupos o custo do serviço de Grupos Ligeiros implica que o seu uso induz uma pequena degradação do desempenho. No entanto a partir, de seis grupos começa a revelar uma melhoria significativa. De referir que com mais de quinze grupos

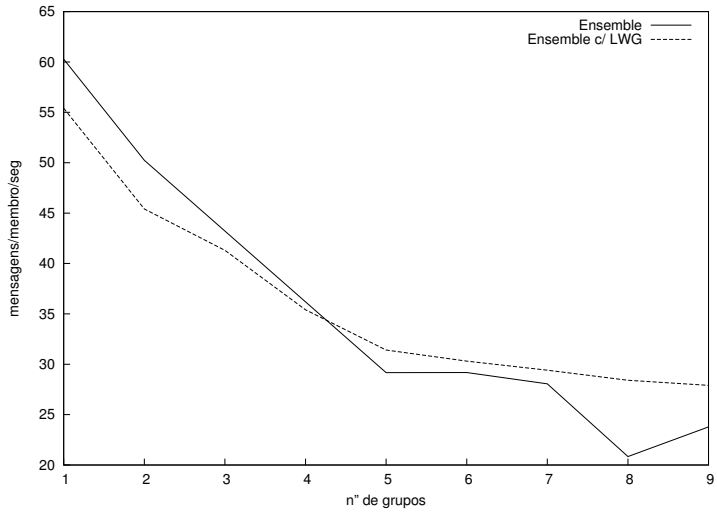

Figura 2: Débito por membro relativamente ao número de grupos a funcionar em paralelo, no Ensemble

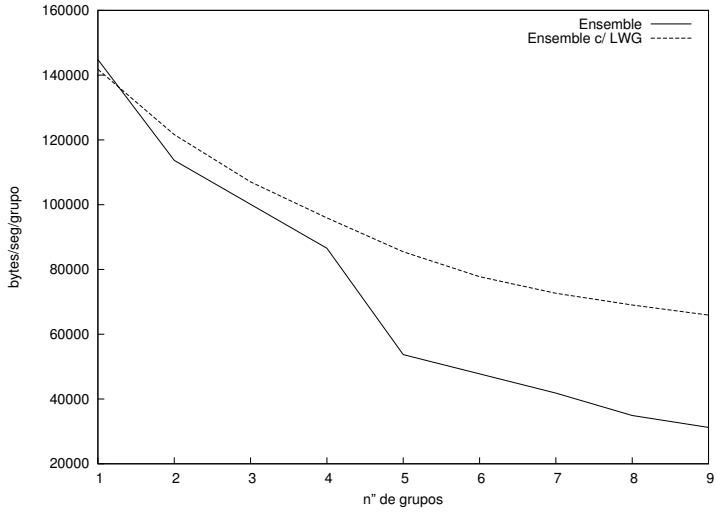

Figura 3: Largura de banda usada por cada grupo a funcionar em paralelo, no Ensemble

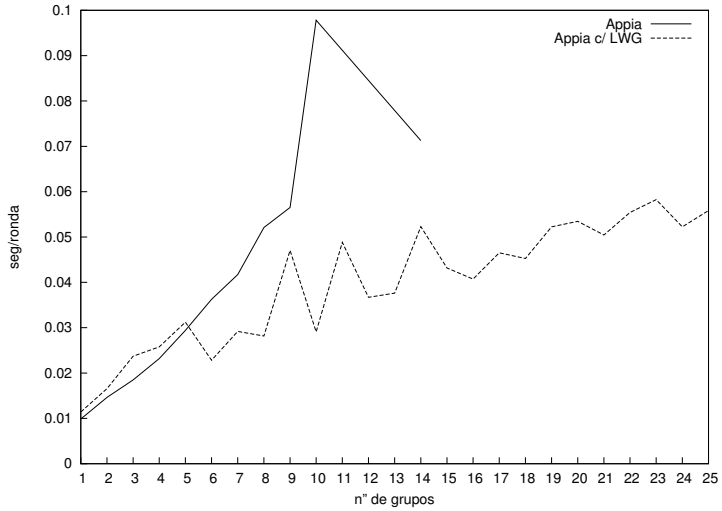

Figura 4: Latência por ronda relativamente ao número de grupos a funcionar em paralelo, no Appia

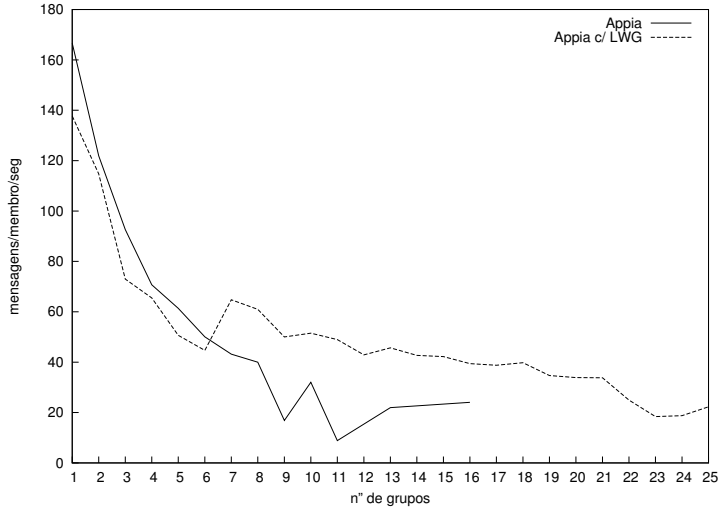

Figura 5: Débito por membro relativamente ao número de grupos a funcionar em paralelo, no Appia

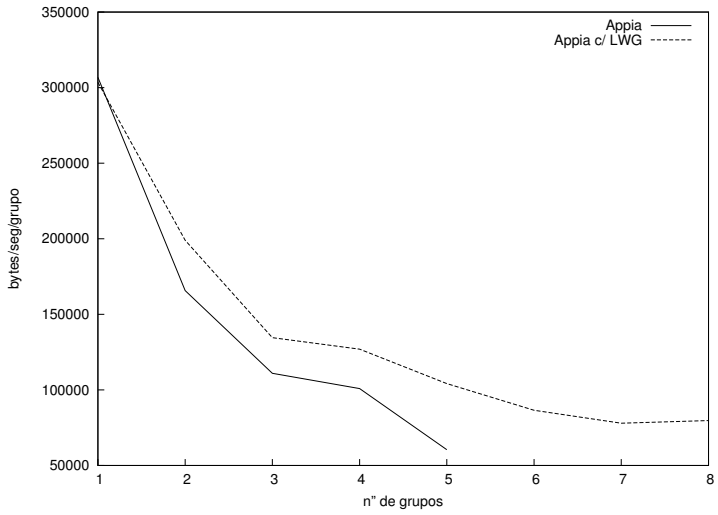

Figura 6: Largura de banda usada por cada grupo a funcionar em paralelo, no Appia

não existem resultados para o Appia sem Grupos Ligeiros porque os testes excederam o limite de tempo estipulado para cada teste.

A Figura 5 mostra o débito por membro, por grupo. Os resultados são semelhantes, com o serviço de Grupos Ligeiros a demonstrar novamente as suas vantagens com mais de seis grupos e o Appia sem Grupos Ligeiros a não conseguir terminar o teste no tempo limite, com mais de dezasseis grupos.

A Figura 6 mostra a largura de banda utilizada por cada grupo. O importante deste gráfico é demonstrar como o serviço de Grupos Ligeiros ao reduzir o consumo dos recursos de suporte à Comunicação em Grupo permite disponibilizar mais largura de banda para cada grupo. Por essa razão, a melhoria do uso do serviço de Grupos Ligeiros neste cenário é imediata.

4 Outras Análises/Avaliações

4.1 "Appia vs Cactus"

Uma outra avaliação das plataformas de composição de protocolos é oferecida por [13], em que são comparados os sistemas Appia e Cactus. A escolha destes sistemas deve-se ao facto de serem, segundo os autores, dois sistemas representativos das duas grande famílias de plataformas de composição, por um lado o Horus[14], Ensemble, Appia e JavaGroups [8] e por outro x-Kernel[7] e Cactus[3].

Como forma de comparação, foi concretizado um serviço idêntico de *difusão atómica tolerante a falhas* nas duas plataformas. O serviço foi concretizado a partir de quatro módulos:

Canal Fiável, que concretiza a comunicação fiável entre dois processos e que é baseada no TCP.

Detector de Falhas, baseado na troca de mensagens de "ping".

Consensos, que utiliza os dois módulos anteriores para resolver questões de consenso.

Difusão Atómica, que utiliza os outros módulos para oferecer difusão atómica.

Com o intuito de simplificar a concretização nas duas plataformas e como forma de garantir que em ambas era executado exactamente o mesmo algoritmo, estes módulos foram concretizados cada um numa classe JAVA, independente da plataforma. Dentro de cada plataforma basta invocar a funcionalidade destas classes. Já a forma como o serviço é colocado dentro da plataforma varia entre as duas concretizações, na medida em que as plataformas também oferecem modelos diferentes. No Cactus o serviço foi concretizado como um único protocolo composto, em que cada um dos módulos referidos é encapsulado dentro de um micro-protocolo. Já no Appia cada módulo foi encapsulado dentro de uma camada, sendo que o serviço corresponde assim a uma pilha com quatro camadas.

Para evitar trocas de contexto entre actividades e para que a comparação seja justa, as concretizações assentam no uso de uma única actividade. No Appia, por definição, esta era a única possibilidade, mas no Cactus teve-se que usar um "semáforo" para controlar o acesso ao protocolo composto e garantir que apenas uma actividade acedia ao mesmo.

O trabalho identifica três características comuns às duas plataformas:

- A estrutura interna dos protocolos, quer sejam sessões do Appia ou micro-protocolos do Cactus, é igual, sendo essencialmente um conjunto de funções para tratamento de eventos e um estado interno, que reagem à ocorrência dos eventos registados. Foi esta semelhança que permitiu que ambas as concretizações usassem a mesma funcionalidade.

- A informação que é enviada pela rede é encapsulado por uma *mensagem*, onde são colocados e retirados cabeçalhos. Embora a estrutura das mesmas varie o conceito é semelhante.

- Ambos os sistema não oferecem nenhum mecanismo de controlo de fluxo, que regule o ritmo pelo qual os eventos são colocados no sistema.

São igualmente identificadas quatro áreas em que os dois sistemas divergem:

- O Appia usa um modelo de composição hierárquico, em que as sessões podem ser partilhadas por vários canais. A comunicação entre as sessões de um canal tem que seguir a ordem pela qual estas foram organizadas no mesmo, não podendo comunicar directamente com as sessões que não lhe estão adjacentes, mesmo que os eventos possam saltar algumas devido à optimização nos seus caminhos. Já o Cactus usa um modelo cooperativo, em que qualquer micro-protocolo pode comunicar directamente com qualquer outro micro-protocolo, dentro do protocolo composto, que deseje.

- No Appia a interacção entre os protocolos só pode ser feita através de eventos, enquanto no Cactus além dos eventos também se pode utilizar informação partilhada, que no entanto não foi utilizada na concretização

efectuada. Igualmente diferente é a ordem pela qual os eventos são recebidos pelos protocolos, que no Appia é FIFO e no Cactus não existe, ou seja, no Cactus não existe uma ordem definida pela qual os micro-protocolos recebem os eventos.

- O modelo de concorrência é diferente, com o Appia a suportar apenas uma actividade enquanto o Cactus permite múltiplas actividades, embora a correcta sincronização entre elas tenha que ser feita pelos programadores dos micro-protocolos, sendo que a plataforma não ofereça qualquer ferramenta que auxilie na tarefa.

- O interface com o ambiente é diferente. No Cactus quer a aplicação como a rede têm que adoptar o interface de um protocolo composto, que não é mais que o interface de um protocolo do x-Kernel. Isto limita a aplicação, pois só oferece um mecanismo de envio de mensagens, que não permite fazer distinções entre dois tipos de mensagens de forma elegante. Por exemplo, considerando uma aplicação que envia mensagens que precisam de ser entregues seguindo uma ordem total, ou mensagens que podem ser entregues por qualquer ordem, o Cactus não oferece mecanismos para distinguir entre as duas sem ser colocando um atributo de tipo na mensagem, que teria que ser conhecido pelos micro-protocolos. No Appia a interacção entre a aplicação e/ou a rede e o ambiente é feita através dos chamados eventos assíncronos. Isto oferece maior flexibilidade, pois o evento que é colocado pode ser de um tipo que exprima as características do mesmo, mas o mecanismo só funciona no sentido da aplicação para o canal, enquanto que a comunicação do canal para a aplicação tem que se feita de uma forma "ad-hoc".

Foram efectuados testes de desempenho das duas concretizações. O primeiro teste efectuado foi de débito relativamente ao tamanho das mensagens enviadas. Foram usados os resultados de usar o TCP directamente como valor de referência. Os resultados indicaram que quer o Appia como o Cactus impõem um custo bastante significativo, sobretudo quando usadas mensagens pequenas. Outro resultado obtido foi que à medida que o tamanho das mensagens aumenta o Cactus melhora mais que o Appia, chegando-se a resultados em que Cactus apresenta um valor quatro vezes melhor. O segundo teste foi de medição do tempo de ida-e-volta de uma mensagem. Foi novamente usado a utilização directa do TCP como referência. Os resultados são análogos aos do primeiro teste, com ambas as plataformas a induzirem um custo significativo, que se nota especialmente nas mensagens de pequena dimensão, e o Cactus a obter resultado melhores que o Appia.

Para compreender os resultados foi usada uma ferramenta de monitorização ("profiling") que indicou que a maior parte do tempo de execução é gasto no tratamento das mensagens, cerca de 84% no Appia e 66% no Cactus, e portanto o acondicionamento da informação na mensagem é o principal condicionador do desempenho de ambas as plataformas. Ambas as plataformas usam os mecanismos de codificação de objectos ("serializing") do Java que se revelam algo ineficientes. A Tabela 2 apresenta os valores obtidos.

Por deficiência na documentação que acompanha o Appia, os autores deste trabalho não usaram alguns dos mecanismos para codificação existentes na ExtendedMessage para resolver exactamente esse problema, o que pode justificar,

Operação	Appia	Cactus
Sincronização de Actividades	7%	24%
Plataforma	4%	8%
Avaliação	5%	2%
Processamento de Mensagens	84%	66%

Tabela 2: Resultados da utilização de uma ferramenta de monitorização no Appia e Cactus.

pelo menos em parte, os resultados obtidos. No entanto, torna-se essencial um estudo mais aprofundado dos mecanismos de acondicionamento usados e suas implicações.

Finalmente são apresentadas guias para o desenvolvimento de futuras plataformas:

- Como muitos eventos destinam-se unicamente de um protocolo para outro, deve haver um mecanismo para que sejam entregues directamente, não usando uma estrutura tão rígida na comunicação entre os protocolos como a hierárquica usada no Appia.

- Qualquer que seja o modelo de concorrência utilizado, deve oferecer as seguintes garantias: sincronização de actividades por omissão, para os programadores dos protocolos não terem que se preocupar com esse aspecto; os eventos devem ser recebidos respeitando uma ordenação FIFO; e transparência do modelo de concorrência, de forma a poder mudá-lo para o que melhor se adaptar as necessidades sem ter de modificar os protocolos.

- O interface entre a aplicação e a plataforma deve seguir aquele usado no Appia, onde a aplicação, ou a rede, podem inserir eventos na plataforma. Já o interface da plataforma para a aplicação deve consistir num conjunto de interfaces comuns alternativos, visto que a especificação de um único interface que se adeqúe a todas as aplicações seria impossível.

Estes três pontos merecem alguns comentários. Em relação ao primeiro à que referir que, ao contrário do que é afirmado, uma análise dos eventos trocados entre camadas, durante o funcionamento de um sistema de Comunicação em Grupo, revelou que poucos eventos se destinam unicamente a duas camadas. Além disso, a capacidade de adicionar novos eventos aliada à optimização que é efectuada sobre o caminho percorrido por estes, permitia que se duas camadas quisessem comunicar directamente entre elas o pudessem fazer usando um tipo de evento criado para esse efeito. Por estas razões não é óbvio que essa rigidez na comunicação entre camadas seja uma desvantagem. Já o segundo ponto vem de encontro ao preconizado aquando da concepção do Appia, que levou ao uso de uma única actividade. O problema inerente ao último ponto também já tinha sido detectado durante o desenvolvimento da plataforma Appia. Numa tentativa de resolver parte do problema foi concretizada uma interface para o sistema de Comunicação em Grupo, a CommAPI.

4.2 "Análise de desempenho de plataformas, em Java, de suporte à comunicação em grupo"

Outra avaliação de plataformas de composição de protocolos aparece em [2]. Esta centra-se mais nos aspectos de desempenho e justifica-se na crescente procura por soluções que permitam o desenvolvimento de sistemas fiáveis baseados no JAVA. Uma forma de conseguir estes sistemas fiáveis é através da replicação por software. Para garantir a consistência das réplicas usa-se muitas vezes primitivas de Comunicação em Grupo. Assim uma forma de atingir a replicação é usando sistemas de Comunicação em Grupo. A avaliação não destinge a plataforma de composição dos protocolos em si, avaliando o conjunto como um todo.

Os sistemas avaliados são o *Spread* [1], Appia e *JavaGroups* [8], enquanto que o Ensemble também é avaliado como referência. O Spread usa uma arquitectura cliente-servidor em que o cliente comunica com um servidor que se encontra na mesma máquina, que participa na comunicação do grupo. O servidor tem um número fixo de protocolos e não permite a configuração ou adição das propriedades pretendidas. O cliente é concretizado em JAVA mas o servidor é concretizado em C. Já o JavaGroups é um sistema totalmente desenvolvido em JAVA e pode ser usado como interface com outro sistema, como o Ensemble, ou utilizar uma plataforma e protocolos próprios. A plataforma para composição de protocolos do JavaGroups oferece uma composição em pilha.

Para efectuar a avaliação foi concretizado em cada uma das plataformas um sistema de replicação activa em três níveis. Nestes, os clientes e as réplicas não participam na sincronização. Esse trabalho é executado por um nível intermédio, responsável por garantir que todas as réplicas executam exactamente as mesmas operações pela mesma ordem, de forma a que os seus estados se mantenham sincronizados. Para garantir isto, este nível tem que resolver duas questões principais, atomicidade das operações e ordenação global das mesmas. Os sistemas de Comunicação em Grupo já oferecem protocolos que garantem estas propriedades. Na realidade até oferecem vários protocolos que utilizando algoritmos diferentes oferecem as mesmas propriedades. Um exemplo é a ordenação global das mensagens, que é alcançada por um protocolo de ordem total, do qual existem muitas alternativas a escolher consoante as necessidades. Na avaliação, quer o JavaGroups como o Apppia foram usados com duas versões de protocolos de ordem total, uma baseada num sequenciador e outra distribuída (no JavaGroups baseada num "token" e no Appia baseada em duas fases).

Foram efectuados testes que mediram a latência na execução de uma operação no cliente e na obtenção de uma ordenação no serviço de Comunicação em Grupo. Foram executados três conjuntos de testes para avaliar a capacidade de escalar do sistema em termos do número clientes, de réplicas, e de elementos no sistema de Comunicação em Grupo. Foram sempre obtidos resultados para o Ensemble, Spread, Appia com ordenação baseada num sequenciador, Appia com ordenação distribuída, JavaGroups com ordenação baseada num sequenciador e JavaGroups com ordenação distribuída. Os resultados, como esperado, indicam que o Ensemble apresenta o melhor desempenho, mas as plataformas baseadas em JAVA conseguem, em algumas condições, resultados muito perto dos do Spread, que é concretizado parcialmente em C. Já nas plataformas em JAVA, o algoritmo de ordenação total distribuída do Appia revelou-se melhor

que o correspondente do JavaGroups, mas ambos apresentam resultados inferiores aos baseados num sequenciador. Com estes o Appia apresenta melhores resultados que o JavaGroups excepto quando se aumenta o número de clientes.

Em termos gerais o Appia apresenta melhores resultados que o JavaGroups, pelo menos em grupos com até oito membros, já que não foram testados grupos maiores.

5 Sumário

Neste relatório foram apresentadas as vantagens e desvantagens que cada uma das plataformas. Depois, foi apresentada uma avaliação dos resultados dos testes de desempenho efectuados sobre as duas plataformas. Finalmente, foram apresentados duas outras avaliações que também analisavam o comportamento da plataforma Appia.

Referências

[1] Y. Amir and J. Stanton. The spread wide area group communication system. Technical Report 98-4, CNDS, 1998.

[2] Roberto Baldoni, Stefano Cimmino, Carlo Marchetti, and Alessandro Termini. Performance analysis of java group toolkits: A case study. In *Scientific Engineering for Distributed Java Applications, International Workshop, FIDJI 2002. Luxembourg-Kirchberg, Luxembourg*, volume 2604 of *Lecture Notes in Computer Science*, pages 49–60. Springer, 2003.

[3] Nina T. Bhatti, Matti A. Hiltunen, Richard D. Schlichting, and Wanda Chiu. Coyote: A system for constructing fine-grain configurable communication services. *ACM Transactions on Computer Systems*, 16(4):321–366, 1998.

[4] K. Guo and L. Rodrigues. Dynamic light-weight groups. In *Proceedings of the 17th International Conference on Distributed Computing Systems*, pages 33–42, Balitmore, Maryland, USA, 1997. IEEE.

[5] M. Hayden. *The Ensemble System*. PhD thesis, Cornell University, Computer Science Department, 1998.

[6] Mark Hayden and Ohad Rodeh. *Ensemble Reference Manual*. Cornell University, Hebrew University, 2000.

[7] N. Hutchinson and L. Peterson. The x-Kernel: An architecture for implementing network protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, 1991.

[8] Javagroups web site, 2004. http://www.javagroups.com/.

[9] H. Miranda, A. Pinto, and L. Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 707–710, Phoenix, Arizona, 2001. IEEE.

[10] Hugo Miranda. Plataforma de suporte ao desenvolvimento e composição de malhas de protocolos. Master's thesis, Departamento de Informática - Universidade de Lisboa, 2001.

[11] A. Pinto, H. Miranda, and L. Rodrigues. Light-weight groups: an implementation in ensemble. In *Fourth European Research Seminar on Advances in Distributed Systems (ERSADS'01)*, Bertinoro (Forli),Italy, 2001.

[12] L. Rodrigues and K. Guo. Partitionable Light-Weight Groups. In *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems (ICDCS'20)*, pages 38–45, Taipe, Taiwan, 2000. IEEE.

[13] Sergio Mena, Xavier Cuvellier, Christophe Grégoire, and André Schiper. Appia *vs.* Cactus: Comparing protocol composition frameworks. In *22nd Symposium on Reliable Distributed Systems. Florence, Italy*, 2003.

[14] R. van Renesse, Ken Birman, and S. Maffeis. Horus: A flexible group communications system. *Communications of the ACM*, 39(4):76–83, 1996.