Avaliação de Replicação de Dados Estruturados Mutáveis em Sistemas Peer-to-Peer

Alexandre Nodari , Alcides Calsavara , Luiz Lima

[1]Programa de Pós-Graduação em Informática Aplicada
Pontifícia Universidade Católica do Paraná
Rua Imaculada Conceição, 1155, Prado Velho
80215-901 Curitiba, PR

{nodari,alcides,laplima}@ppgia.pucpr.br

Abstract. *A method to evaluate optimistic protocols for mutable data replication in peer-to-peer systems is proposed. Its use is shown through the simulation of four distinct replica protocols with respect to replica update and reconciliation in case a replica reenters the system. The results of the simulation permit to observe the influence on the performance of each protocol of the number of replicas, the chance of each node to reply messages sent to it and the rate of write operations. The carried out experiments permit to conclude that it is appropriate to employ optmistic data replication in peer-to-peer systems if some conditions are satisfied.*

Resumo. *É proposta uma metodologia para avaliação de protocolos de replicação otimista de dados mutáveis em sistemas peer-to-peer e é mostrada a sua aplicação através de experimentos de simulação com quatro protocolos de replicação que diferem entre si na maneira como fazem atualização das réplicas e a reconciliação em caso de retorno de uma réplica ao sistema. Os resultados da simulaçao permitem observar a influência do número de réplicas, da chance de cada nó responder às mensagens recebidas e da taxa de operações de escrita no desempenho de cada protocolo. O experimento realizado permite concluir que o emprego de replicação otimista de dados é viável em sistemas peer-to-peer se algumas condições forem observadas.*

1. Introdução

Os mecanismos de replicação atualmente empregados em sistemas peer-to-peer atuam primordialmente sobre dados não estruturados e imutáveis, pois sua principal utilização é nas aplicações de compartilhamento de arquivos multimídia, como discutem [Cuenca-Acuna et al., 2003] e [Lin et al., 2004], a fim de proporcionar maior disponibilidade e mais rapidez no acesso aos dados. Nessas aplicações, a replicação é implícita no mecanismo de compartilhamento, usando eventualmente fragmentos dos arquivos compartilhados. Como tanto os arquivos quanto seus fragmentos são dados imutáveis, não é necessário utilizar um mecanismo de replicação que mantenha consistência entre as réplicas. Além de imutáveis, os dados não são estruturados, o que simplifica a sua fragmentação. Porém, a evolução desses sistemas e o crescimento de outros tipos de aplicações, como as de comunicação e colaboração, ou até a convergência da arquitetura peer-to-peer com a de computação em grade sugere também

a necessidade de replicação de dados estruturados e mutáveis, como discutido em [Androutsellis-Theotokis and Spinellis, 2004].

De fato, as oportunidades para o emprego de replicação de dados estruturados mutáveis são muitas, tanto para fins de melhoria no desempenho como para fins de tolerância a faltas. Algumas aplicações de compartilhamento de arquivos replicam metadados sobre esses arquivos para permitir consultas mesmo em nós desconectados e ainda permitir consultas estruturadas como, por exemplo, buscar músicas de um certo intérprete ou de um certo álbum. Um sistema de arquivos distribuído tira proveito da replicação de metadados para aperfeiçoar o acesso aos arquivos, como em [Bolosky et al., 2000]. Da mema forma, uma ferramenta de armazenamento distribuída pode aperfeiçoar o acesso aos dados, como descrito em [Kubiatowicz et al., 2000]. Aplicações de comunicação e colaboração poderiam evitar a necessidade de um servidor central se as funcionalidades de autenticação, inicialização e atualização de lista de contatos estivessem replicadas. Aplicações distribuídas poderiam replicar dados de controle de redes, como descrito em [Granville et al., 2005], ou replicar dados de execução da aplicação em grade de forma a se tornarem adaptáveis ao ambiente, redirecionando tarefas para nós com maior banda ou maior processamento disponível, conforme [Schuler et al., 2003].

Embora a teoria de replicação de dados seja bem consolidada, a implementação de protocolos de replicação em sistemas peer-to-peer exige um projeto adequado às suas características mais acentuadas, tais como a alta latência e a entrada e saída constantes de nós. O nível de consistência de dados mutáveis armazenados em uma DHT é verificado experimentalmente em [Picconi et al., 2007]. A DHT é hospedada em uma rede peer-to-peer, na qual os nós estão sujeitos a variados níveis de entrada e saída e a consistência é mantida usando-se o algoritmo baseado em quorum de operações de leitura e escrita. Observa-se que o algoritmo é muito sensível aos eventos de entrada e saída de nós, pois isso dificulta bastante a obtenção de quorum. Em [Antoniu et al., 2004], um protocolo hierárquico de consistência de réplicas é proposto e avaliado no contexto de computação em grade sobre redes peer-to-peer. O principal objetivo do protocolo é separar claramente a manutenção da réplicas do gerenciamento de tolerância a faltas. A avaliação feita para esse protocolo em particular permitiu verificar o seu desempenho quanto a latência de operações de leitura e escrita nos dados.

Neste artigo, propomos uma metodologia para avaliação de protocolos de replicação otimista de dados mutáveis em sistemas peer-to-peer e mostramos a sua aplicação através de experimentos de simulação com quatro protocolos de replicação. Todos esses protocolos são *single-master*, mas diferem na maneira como fazem a atualização das réplicas e a reconciliação em caso de retorno de uma réplica ao sistema. Cada protocolo é avaliado em simulações nas quais variam-se o número de réplicas de cada master, a chance de cada nó responder às mensagens recebidas e a taxa de operações de escrita.

O restante deste artigo está organizado como segue. A Seção 2 discute os principais conceitos de replicação de dados que influenciaram na metodologia proposta e no experimento realizado. A Seção 3 comenta sobre plataformas para a construção de sistemas peer-to-peer e justifica a escolha de uma plataforma específica para a realização do experimento de avaliação de protocolos de replicação. A Seção 4 apresenta a metodologia proposta e descreve as condições para a realização do experimento da metodologia. A Seção 5 descreve a aplicação utilizada no experimento. A Seção 6 apresenta os principais

resultados obtidos no experimento. A Seção 7 discute esses resultados. Finalmente, a Seção 8 apresenta as principais conclusões sobre o trabalho realizado e comenta sobre possíveis trabalhos futuros.

2. Replicação de Dados Mutáveis

Para protocolos de replicação em sistemas distribuídos que tratam de informações mutáveis, ou seja, aceitam operações de escrita, existem duas classificações. A primeira determina qual(is) nó(s) recebe(m) as operações de escrita e como e quando essas operações são transferidas para as réplicas [Lung et al., 2004], determinando as opções *passiva* e *ativa*. Outra forma de classificação de protocolos de replicação em sistemas distribuídos determina como lidar com operações de escrita e leitura concorrentes, determinando as opções *pessimista* e *otimista*. Na replicação otimista não há a necessidade de sincronização durante o acesso as réplicas: permite que a leitura e escrita de dados sejam feitas assumindo de forma "otimista" que conflitos raramente irão acontecer e, se acontecerem, podem ser resolvidos.

A replicação pessimista funciona bem em redes locais, onde a latência de rede é pequena e falhas não são comuns. Porém, a Internet, que é onde normalmente se configuram os sistemas peer-to-peer, ainda é lenta e pouco confiável. Além disso, redes formadas por computadores móveis, com conectividade intermitente, estão se tornando mais populares. Protocolos pessimistas também não escalam bem em ambientes distribuídos. Por outro lado, a replicação otimista de dados aumenta a sua disponibilidade, possibilitando melhora no desempenho das aplicações. Aumenta-se também a escalabilidade do número de nós participantes ao relaxar a sincronização necessária entre as réplicas. Um bom equilíbrio entre velocidade de propagação de operações de escrita, número de réplicas e consumo de banda é importante. Quanto mais rapidamente as alterações forem propagadas para as réplicas, maior é a convergência entre as réplicas e, ainda, menor é a probabilidade de um conflito ocorrer. Em sistemas peer-to-peer o fator preocupante fica com a taxa normalmente elevada de entrada e saída de nós da rede. Isso pode fazer com que muitas mensagens enviadas no sistema não cheguem ao seu destino.

Uma questão crítica no projeto de replicação otimista refere-se ao número de réplicas que podem aceitar operação de escrita. Os sistemas que só possuem uma réplica que aceita operações de escrita são denominados *single-master*, enquanto que sistemas onde mais de uma réplica aceita operações de escrita são denominados *multi-master*. Sistemas single-master são mais simples, porém têm sua disponibilidade para operações de escrita reduzida, principalmente em sistemas com operações de escrita freqüentes. A escalabilidade do sistema single-master está limitada ao desempenho da única instância que recebe operações de escrita. Sistemas multi-master oferecem maior disponibilidade, mas com um aumento significativo de complexidade. E apesar da escalabilidade ser maior que em um sistema single-master, ao aumentar o número de réplicas que aceitam operações de escrita aumenta-se também a taxa de conflitos entre operações, determinando que esta vantagem não seja normalmente muito grande.

Existem duas maneiras de transmitir alterações em dados para suas réplicas. A primeira é transferir todas as informações novamente, sobrepondo as informações existentes. Esse mecanismo é denominado transferência de estado. Outra forma é enviar somente a operação que causou a alteração, denominada transferência de operações. Para

definir quando transferir uma operação ou estado pode-se optar pela técnica *push*. Nesta técnica o nó que recebe uma operação inicia imediatamente a propagá-la para os outros nós. Isso é recomendado, pois ajuda a diminuir a divergência entre réplicas e evita buscas (*pull*) desnecessárias. Técnicas *pull* não são recomendadas para redes dinâmicas por aumentar o número de mensagens sobre um ambiente dinâmico e mais propenso a falhas. Mas em sistemas single-master é possível usar uma técnica híbrida na qual usualmente a réplica master propaga (push) as novas operações, mas também uma réplica que está se conectando a rede pode buscar (pull) na réplica master as últimas alterações.

3. Infra-estrutura para Sistemas Peer-to-Peer

Sistemas peer-to-peer têm sido bastante pesquisados devido ao sucesso de algumas aplicações comerciais. Com isso surgiram muitas opções de infra-estrutura para a construção desses sistemas. Entre essas, destacam-se os arcabouços JXTA e Pastry, pois têm recebido muita atenção nas pesquisas recentes, além de oferecem implementações públicas e gratuitas.

JXTA [1][JXTA, 2008] Uma especificação de arcabouço para sistemas peer-to-peer independente de linguagem de programação, plataforma para comunicação entre dispositivos, localização física e tecnologia de rede no qual se encontram instalados. Para tanto, mapeia os principais conceitos de computação distribuída numa série de entidades e elementos de comunicação, com distinção e hierarquia entre os peers de acordo com suas funções específicas. Basicamente, peers transmitem mensagens apenas através de *pipes*, isto é, canais virtuais que são, em geral, unidirecionais e não-confiáveis, e que se conectam a um ponto de entrada e outro de saída (*end points*). Mensagens são documentos XML que carregam em seu cabeçalho, além do identificador da fonte, a informação de roteamento necessária, tal como a seqüência de peers a ser percorrida.

Pastry [Rowstron and Druschel, 2001] Um arcabouço para a construção de redes peer-to-peer sobrepostas à Internet, com mecanismos para localização de recursos e roteamento de mensagens. Os nós de uma rede Pastry são organizados em forma de anel, sendo que cada nó recebe um identificador único de 128 bits, isto é, a quantidade de nós na rede peer-to-peer pode variar entre 0 e 2^{128}. Há ainda mecanismos no arcabouço que permitem que a rede se recomponha automaticamente em caso de falhas, tornando-a relativamente robusta. O roteamento de mensagens baseia-se simplesmente numa função logarítmica que é aplicada a cada passo do roteamento.

Optamos pelo Pastry para a realização do experimento pelas seguintes razões:

1. Escalabilidade é um dos principais requisitos de sistemas peer-to-peer. O fato de JXTA utilizar o conceito de peers diferenciados para certas funções contribui negativamente para esse fim. Por outro lado, Pastry trabalha com redes completamente descentralizadas, que é uma uma característica que aumenta o potencial de escalabilidade.
2. Simplicidade é uma característica bastante desejável para o arcabouço, pois facilita o desenvolvimento de protótipos. Nesse ponto, a arquitetura e a interface de programação do Pastry são vantajosas em relação ao JXTA.

[1]Juxtapose

3. Facilidade de simulação é fundamental na realização de experimentos envolvendo computação distribuída. O arcabouço Pastry possui uma interface de programação específica para esse fim através da qual é possível simular uma rede peer-to-peer, sem perda de generalidade. Isto é, a plataforma de simulação para peer-to-peer permite avaliar aplicações com o mesmo grau de fidelidade que se teria em um sistema peer-to-peer real.

4. Metodologia e Ambiente de Avaliação

4.1. Nomenclatura

Os seguintes termos são definidos como parte da metodologia proposta para avaliação de protocolos de replicação otimista em sistemas peer-to-peer:

comando Uma mensagem contendo uma operação de escrita.

intensidade de escrita Chance de um nó enviar um comando ao invés de enviar uma mensagem comum (contendo uma consulta ou uma resposta). Por exemplo, se a intensidade de escrita é de 10%, cada nó envia aproximadamente 10 comandos e 90 mensagens comuns a cada 100 mensagens.

chance de vida Chance de um peer estar operante, ou seja, receber e, se necessário, responder as mensagens que lhe sejam enviadas. Por exemplo, se a chance de vida for 10%, cada nó recebe aproximadamente 10 em cada 100 mensagens enviadas para ele.

réplica efetiva Réplica que recebe pelo menos um comando executado no master. Isso permite ao nó que contém a réplica estar ciente que dados de um outro nó estão replicados nele. Assim é possível ao nó responder consultas sobre esses dados e executar reconciliações com o nó master desses dados.

4.2. Protocolos de Replicação

Somente os protocolos single-master de replicação foram avaliados. Futuramente, outras combinações de estratégia de propagação de atualização e de conciliação deverão ser experimentadas e avaliadas.

StatePush–NullSync A cada comando executado em um master, uma mensagem de atualização de estado é criada e enviada (em multicast) para o grupo de réplicas. Assim, na atualização das réplicas, há duas latências e o número de mensagens é da $O(n)$, onde n é o número de réplicas. Não ocorre reconciliação quando um nó volta a operar.

CommandPush–NullSync A cada execução de comando em um master, o próprio comando é enviado (em multicast) para execução em cada réplica. Assim, na atualização das réplicas, há duas latências e o número de mensagens é da $O(n)$, onde n é o número de réplicas. Não ocorre reconciliação quando um nó volta a operar.

CommandPush–StatePullSync A cada execução de um comando em um master, o próprio comando é enviado para execução em cada réplica. Assim, na atualização das réplicas, há duas latências e o número de mensagens é da $O(n)$, onde n é o número de réplicas. Ocorre reconciliação baseada em *estado* toda a vez que um nó contendo réplicas volta a operar: para cada réplica que o nó mantém, consulta nos respectivos masters os seus estados atuais. Assim, na reconciliação, há duas

latências e o número de mensagens é da $O(n)$, onde n é o número de masters consultados.

CommandPush–CommandPullSync A cada execução de um comando em um master, o próprio comando é enviado para execução em cada réplica. Assim, na atualização das réplicas, há duas latências e o número de mensagens é da $O(n)$, onde n é o número de réplicas. Ocorre reconciliação baseada em *comando* toda a vez que um nó contendo réplicas volta a operar: para cada réplica que o nó mantém, consulta nos respectivos masters os correspondentes comandos que deixou de executar. Assim, na reconciliação, há duas latências e o número de mensagens é da $O(n)$, onde n é o número de masters consultados.

4.3. Métricas de Avaliação

As seguintes medições são realizadas para a avaliação dos protocolos de replicação:

1. Número de mensagens trocadas entre os nós: permite verificar o *overhead* de comunicação provocado pelo protocolo.
2. Tamanho das mensagens em relação ao tamanho dos dados: permite verificar o consumo de banda provocado pelo protocolo.
3. Taxa de réplicas efetivas (medida através do percentual de réplicas efetivas em relação ao número de réplicas configurado para a simulação): permite verificar o nível de participação das réplicas.
4. Qualidade das réplicas efetivas (medida através da média dos percentuais de comandos aplicados corretamente em cada réplica em relação ao total de comandos aplicados no master): permite verificar a própria qualidade das réplicas, isto é, o nível de correção das réplicas.

4.4. Ambiente de Experimentação

O experimento foi realizado através da simulação de uma rede peer-to-peer de 100 nós, usando o arcabouço Pastry em um computador com processador Athlon 2600 e 1GB de memória e o desenvolvimento da aplicação alvo na linguagem Java, com as seguintes condições:

1. Cada nó da rede é *master* de seus próprios dados, enquanto outros nós armazenam réplicas desses dados. Assim, todos os nós são *master* e ainda podem armazenar réplicas de dados de outros nós. O número de réplicas esperadas é único em cada simulação, isto é, cada *master* tem os seus dados replicados o mesmo número de vezes, sendo que a alocação de réplicas é feita de maneira aleatória. Por exemplo, se o número de réplicas é configurado em quatro, os dados de cada master são replicados em outros quatro nós escolhidos aleatoriamente no conjunto completo de nós.
2. Em cada simulação são utilizados os 100 nós da rede, sendo que cada um envia 100 mensagens. Na prática, 100 é um número relativamente baixo para redes peer-to-peer reais, mas o equipamento disponível para o experimento impôs tal restrição, deixando experimentos com redes maiores para trabalhos futuros.
3. A cada mensagem enviada, um tipo de mensagem (comando ou mensagem comum) é escolhido de acordo com o parâmetro *intensidade de escrita*.
4. O recebimento das mensagens pelos nós depende da *chance de vida* especificada para a simulação.

5. As medições de taxa e qualidade de réplicas efetivas são feitas imediatamente após os ciclos de envio de mensagens a fim de evitar a convergência das réplicas nos protocolos de atualização com reconciliação, permitindo assim uma comparação mais justa entre os protocolos.
6. São feitas as seguintes variações nos parâmetros de simulação:
   (a) Número de réplicas esperadas: 1, 2, 4 e 8.
   (b) Chance de vida: 10%, 20%, 40% e 80%.
   (c) Intensidade de escrita: 10%, 20%, 40% e 80%.

5. Aplicação Alvo

Uma aplicação simples e, ao mesmo tempo, adequada para a verificação da metodologia de avaliação de protocolos de replicação proposta neste trabalho é o gerenciamento de listas de contatos pessoais. Atualmente a maioria dos sistemas existentes utiliza um servidor central que autentica os usuários e mantém suas listas de contatos. Esta lista de contatos é um exemplo de metadados que poderiam estar replicados, evitando a necessidade de um servidor central.

Como somente o próprio usuário pode alterar sua lista de contatos, o mecanismo de replicação pode ser *single-master*. O nó onde o usuário está conectado no momento é o *master*. Exemplos de comandos incluem inserção de um novo contato, remoção de um contato e edição de um contato. Se, por exemplo, o protocolo de replicação utilizado for o *CommandoPush-CommandPullSync*, ao receber um comando de escrita, o nó *master* propaga esse comando para todas as réplicas. As que estiverem operantes executam imediatamente o mesmo comando. Se alguma réplica não estiver operante, ao voltar a esse estado, obtém os comandos faltantes no *master*. Na ocorrência de comandos não comutativos, como por exemplo, uma inserção e uma deleção, a ordenação é feita no nó *master*. No caso de uma réplica receber comandos fora de ordem, ela deve ordená-los antes de executar ou obter novamente os comandos no *master*.

6. Resultados

Os resultados da simulação estão parcialmente representados pelos gráficos das Figuras 1 a 4. Os objetivos da exibição desses gráficos são o de auxiliar na discussão de certos fenômenos observados nos experimentos e o de mostrar o potencial de análise da metodologia de avaliação apresentada neste trabalho.

Os gráficos das Figura 1 a 3 referem-se ao protocolo de atualização de réplicas que denominamos *StatePush–NullSync*, definido na Seção 4.2. Em cada uma dessas Figuras, uma métrica de avaliação em particular (das quatro métricas definidas na Seção 4.3) é objeto dos correspondentes gráficos. A Figura 1 permite verificar o comportamento do número de mensagens em diversas situações. Observa-se que o número de mensagens cresce linearmente de acordo com o número de réplicas[2] também com a intensidade de escrita, mas há um certa independência quanto à chance de vida. A Figura 2 permite verificar a variação do tamanho médio das mensagens. Nota-se uma diferença significativa no tamanho médio das mensagens. Este aumenta com a intensidade de escrita, pois mais comandos são enviados e o tamanho dos comandos é usualmente maior que

[2]Para fins de legibilidade, foi feita interpolação nos gráficos em que o eixo horizontal representa o número de réplicas embora o seu domínio seja discreto.

o tamanho das mensagens comuns nesse protocolo; uma mensagem pode atingir até 40 vezes o tamanho dos dados. A Figura 3 permite verificar a qualidade das réplicas efetivas. Observa-se, principalmente, que a qualidade das réplicas efetivas aumenta com a chance de vida dos nós e com a intensidade de escrita da aplicação, pois assim aumentam-se as chances de uma réplica receber um comando enviado por seu master. O custo desse alto índice médio de qualidade é o tamanho elevado das mensagens, conforme já discutido. Essa mesma avaliação detalhada do protocolo *StatePush–NullSync* poderia ser feita para os três outros protocolos de replicação, mas aqui não dispomos do espaço necessário para isso.

Finalmente, a Figura 4 exemplifica uma simples comparação entre os quatro protocolos definidos, com relação às quatro métricas experimentadas. Tal simplicidade na comparação foi possível através da fixação de dois parâmetros de simulação, a saber o número de réplicas e a intensidade de escrita. Entretanto, a variação no parâmetro de chance de vida dos nós é adequada para a análise, pois tende a ser o parâmetro mais significativo em sistemas peer-to-peer. Nesse cenário em particular, analisando-se o tamanho médio das mensagens, percebe-se que os protocolos mais complexos, isto é, que fazem reconciliação, acabam por compensar o aumento no número de mensagens, pois só utilizam mensagens maiores durante as reconciliações. Os protocolos que fazem reconciliação dobram o número de mensagens no pior caso, mas o protocolo *StatePush–NullSync* gera mensagens que são mais que quatro vezes maior que o tamanho dos dados do nó. A razão entre réplicas efetivas e esperadas aumenta de acordo com a chance de um nó estar operante, conforme esperado. Mas é interessante ressaltar que, a partir de 40% de chance de vida, essa razão já é quase 100%. A qualidade das réplicas efetivas nos protocolos com reconciliação é intermediária em relação às outras duas simulações: esses protocolos se comportam melhor que o protocolo de transferência de operações e pior que o protocolo de transferência de estados para chances de vida inferiores a 40%. Para chances de vida superiores a 40% os protocolos com com reconciliação se comportam de maneira semelhante ao protocolo de transferência por estados.

7. Discussão

Observamos que a replicação otimista se torna bastante interessante quando a chance de vida dos nós é superior a 40%, ou seja, que os nós participantes da aplicação tenham uma taxa de entrada e saída da rede que possibilite que pelo menos 40% das mensagens cheguem com sucesso ao seu destino. Apesar de essa condição ser facilmente obtida em aplicações departamentais, algumas aplicações distribuídas mundialmente não apresentam essa característica. A partir dessa faixa de chance de vida, o número de mensagens é cada vez menor devido ao menor número de reconciliações necessárias. A quantidade de réplicas efetivas é alto mesmo em sistemas com intensidade de escrita baixa. A qualidade das réplicas é boa, partindo de aproximadamente 85% de convergência com o master e tendendo a 100% com o aumento da chance de vida.

O protocolo baseado em transferência de estados é bastante simples e oferece bons resultados mesmo com chance de vida inferior a 40%. Porém, apresenta uma desvantagem, pois aumenta o tamanho médio das mensagens. Isso pode ser um problema caso o tamanho dos dados em cada nó seja grande, ou crescer rapidamente com a execução de comandos de escrita. Esses problemas ainda podem ser acentuados caso o sistema tenha uma alta intensidade de escrita.

Os experimentos confirmaram que a técnica de transferência *push* é interessante para sistemas com grande quantidade de operações de escrita, pois melhora sensivelmente a convergência das réplicas com o master.

Os protocolos que usam reconciliação (por estado ou por comandos) diminuem consideravelmente a média de tamanho das mensagens. Em contrapartida, aumentam o número de mensagens. Porém, principalmente em sistemas com alta intensidade de escrita, o consumo total de banda é menor nessas configurações. Os dois protocolos com reconciliação apresentam resultados bastante parecidos, exceto pela diferença a favor da reconciliação por comandos no tamanho médio das mensagens. Esse ganho, entretanto, vem com um aumento da complexidade, pois nesse protocolo é necessário que cada nó guarde um histórico dos últimos comandos executados.

8. Conclusão

O trabalho realizado permite concluir que o emprego de replicação otimista de dados é viável em sistemas peer-to-peer se algumas condições forem observadas. A metodologia de avaliação proposta e o correspondente experimento realizado geraram resultados que mostram que mesmo protocolos simples, como o *push* com transferência de estados, pode oferecer bons resultados e que, aumentando a complexidade do protocolo, é possível obter um mecanismo de replicação com uma relação consumo de banda versus convergência das réplicas em relação ao master bastante interessante, como nos protocolos com reconciliação. Para muitos sistemas não é necessário um projeto de replicação otimista muito complexo. Portanto, o emprego de replicação otimista de forma bem controlada permite aperfeiçoar sistemas peer-to-peer e até mesmo possibilita funcionalidades que seriam inviáveis sobre um mecanismo de replicação pessimista.

Futuramente, seria interessante avaliar as respostas dos protocolos estudados para aplicações de larga escala. Se aumentado o número de nós nas simulações é possível efetuar tal avaliação. Outra alteração interessante nos parâmetros das simulações seria utilizar intensidades de escrita pequenas, bastante comuns em aplicações reais, como por exemplo, valores menores que 10%. Pode-se ainda evoluir a simulação de entrada e saída dos nós da rede, tornando o comportamento da simulação mais próximo da realidade de algumas aplicações atuais, incluindo a necessidade de lidar com a saída permanente de alguns nós. Outros protocolos de replicação de dados estruturados podem ser interessantes para outros tipos de informações de sistemas peer-to-peer. Replicação multi-master, com opções de detecção e resolução de conflitos [Martins et al., 2006] são um tema extenso de estudo. Finalmente, latência poderia ser verificada como uma métrica de avaliação dos protocolos.

Referências

Androutsellis-Theotokis, S. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371.

Antoniu, G., Deverge, J.-F., and Monnet, S. (2004). Building fault-tolerant consistency protocols for an adaptive grid data-sharing service. Technical Report 5309, IRISA, Rennes, França.

Bolosky, W. J., Douceur, J. R., Ely, D., and Theimer, M. (2000). Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 34–43, New York, NY, USA. ACM.

Cuenca-Acuna, F. M., Martin, R. P., and Nguyen, T. D. (2003). Autonomous replication for high availability in unstructured p2p systems. In *SRDS*, pages 99–108. IEEE Computer Society.

Granville, L.Z.and da Rosa, D., Panisson, A., Melchiors, C., Almeida, M., and Tarouco, L. (2005). Managing computer networks using peer-to-peer technologies. *Communications Magazine, IEEE*, 43(10):62–68.

JXTA (2008). JXTA Project. URL: http://www.jxta.org.

Kubiatowicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., and Zhao, B. (2000). Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM.

Lin, W. K., Chiu, D. M., and Lee, Y. B. (2004). Erasure code replication revisited. In *P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, pages 90–97, Washington, DC, USA. IEEE Computer Society.

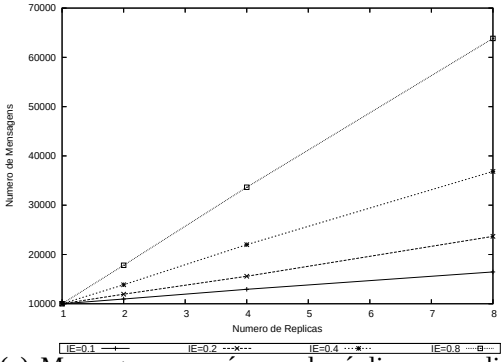
Lung, L., Bessani, A., and Fraga, J. (2004). Programação de sistemas distribuídos confiáveis. In *ERI-SC'04 – XII Escola Regional de Informática*, pages 1–40. SBC.

Martins, V., Akbarinia, R., Pacitti, E., and Valduriez, P. (2006). Reconciliation in the appa p2p system. In *ICPADS '06: Proceedings of the 12th International Conference on Parallel and Distributed Systems*, pages 401–410, Washington, DC, USA. IEEE Computer Society.

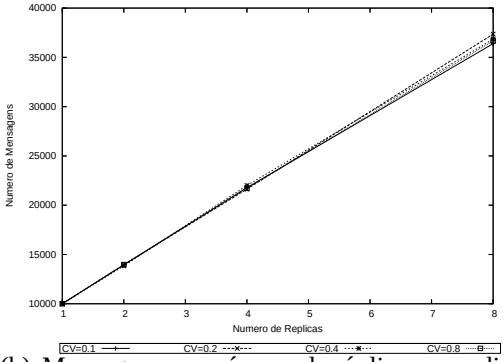
Picconi, F., Busca, J.-M., and Sens, P. (2007). An experimental evaluation of the Pastis peer-to-peer file system under churn. Technical Report 6114, INRIA, França.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350.

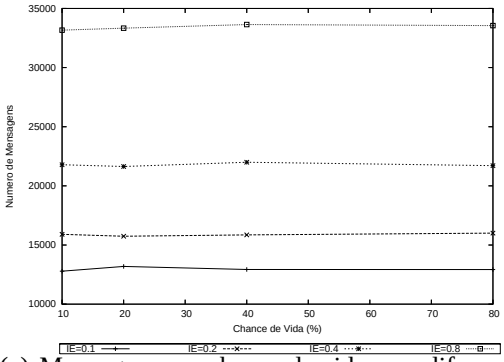
Schuler, C., Weber, R., Schuldt, H., and Schek, H. (2003). Peer-to-peer process execution with osiris. In *Proc. of First International Conference on Service-Oriented Computing ICSOC*.



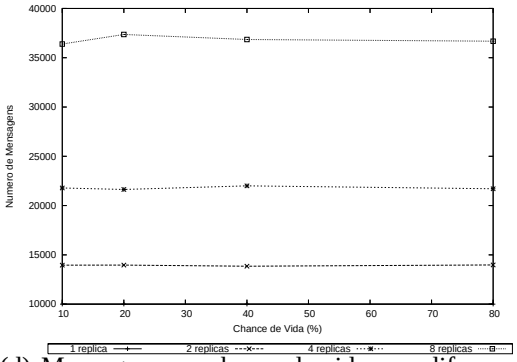
(a) Mensagens por número de réplicas para diferentes intensidades de escrita (IE) e chance de vida de 40%



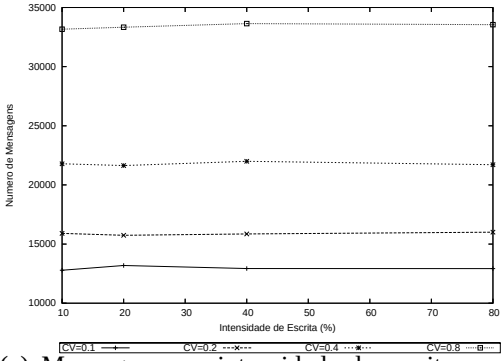
(b) Mensagens por número de réplicas para diferentes chances de vida (CV) e intensidade de escrita de 40%



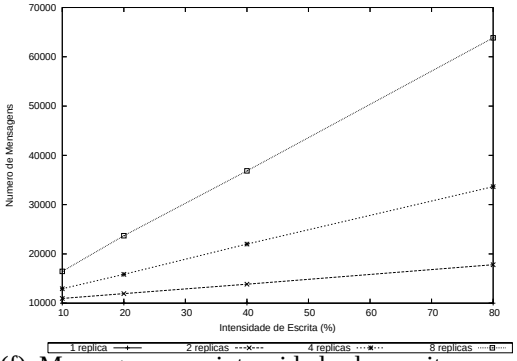
(c) Mensagens por chance de vida para diferentes intensidades de escrita (IE) e 4 réplicas



(d) Mensagens por chance de vida para diferentes números de réplicas e intensidade de escrita de 40%

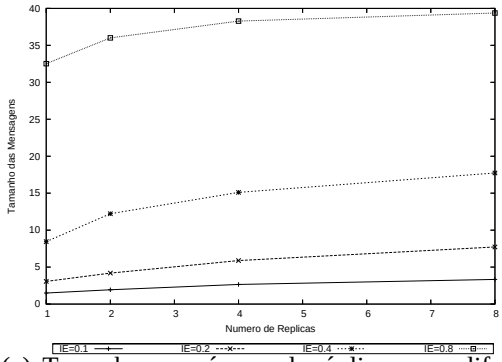


(e) Mensagens por intensidade de escrita para diferentes chances de vida (CV) e 4 réplicas

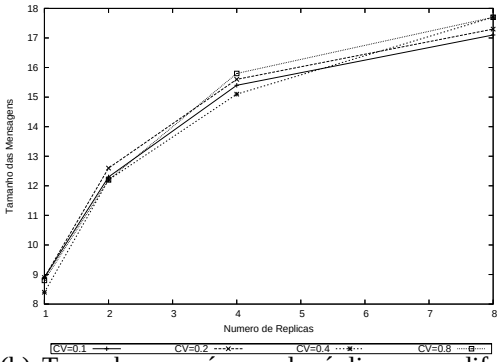


(f) Mensagens por intensidade de escrita para diferentes números de réplicas e chance de vida de 40%

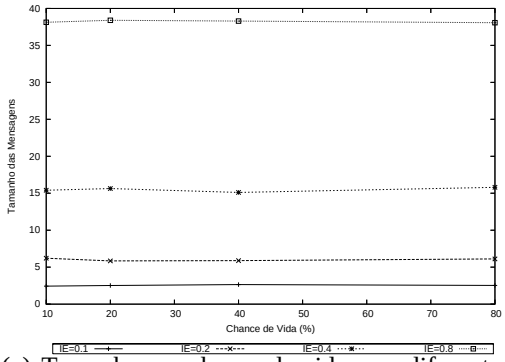
Figura 1. Número de mensagens no protocolo StatePush–NullSync



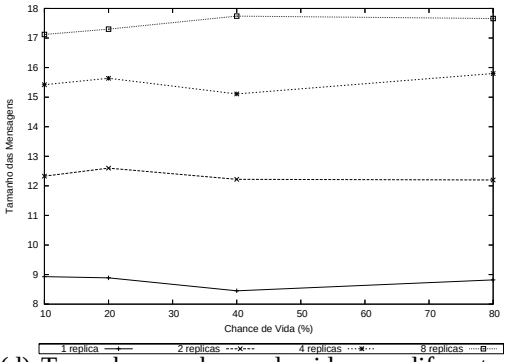
(a) Tamanho por número de réplicas para diferentes intensidades de escrita (IE) e chance de vida de 40%



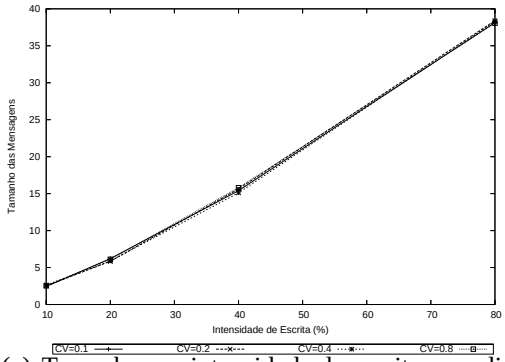
(b) Tamanho por número de réplicas para diferentes chances de vida (CV) e intensidade de escrita de 40%



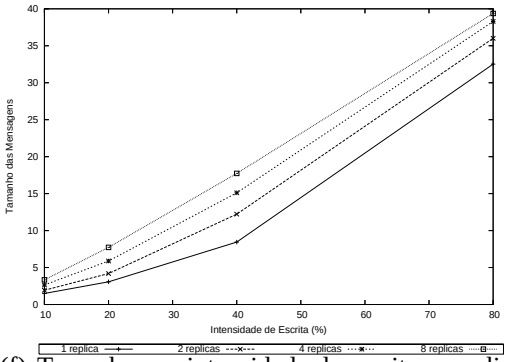
(c) Tamanho por chance de vida para diferentes intensidades de escrita (IE) e 4 réplicas



(d) Tamanho por chance de vida para diferentes números de réplicas e intensidade de escrita de 40%

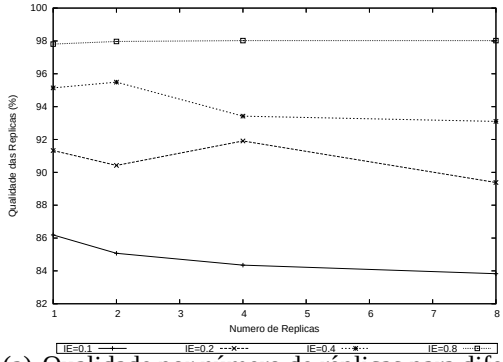


(e) Tamanho por intensidade de escrita para diferentes chances de vida (CV) e 4 réplicas

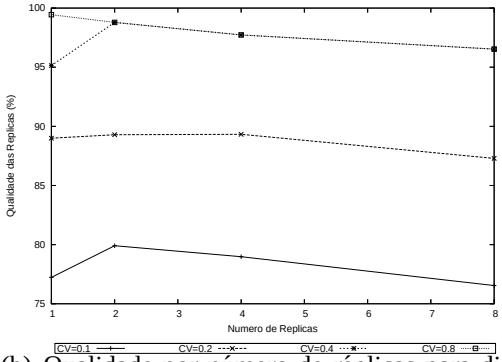


(f) Tamanho por intensidade de escrita para diferentes números de réplicas e chance de vida de 40%

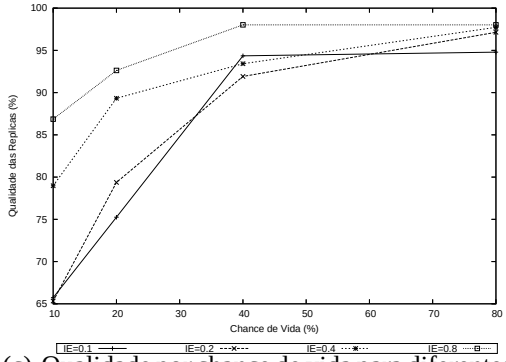
Figura 2. Tamanho das mensagens no protocolo StatePush–NullSync



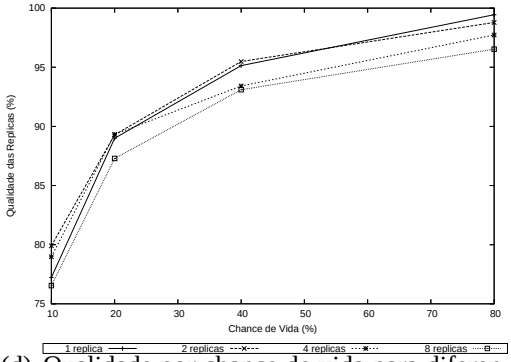
(a) Qualidade por número de réplicas para diferentes intensidades de escrita (IE) e chance de vida de 40%



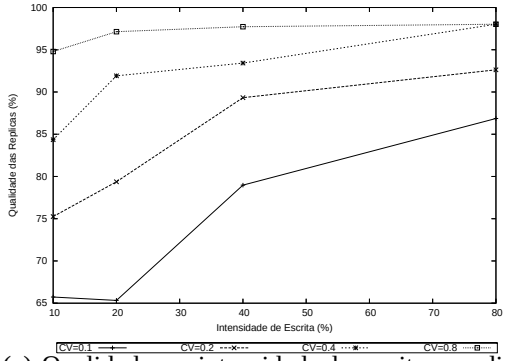
(b) Qualidade por número de réplicas para diferentes chances de vida (CV) e intensidade de escrita de 40%



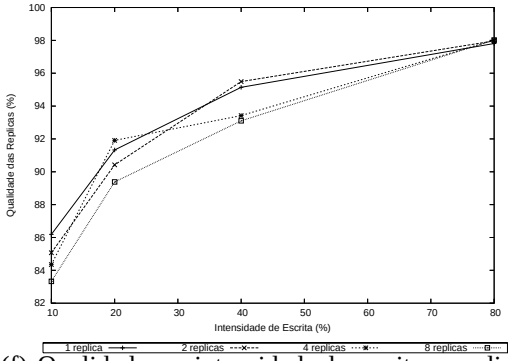
(c) Qualidade por chance de vida para diferentes intensidades de escrita (IE) e 4 réplicas



(d) Qualidade por chance de vida para diferentes números de réplicas e intensidade de escrita de 40%

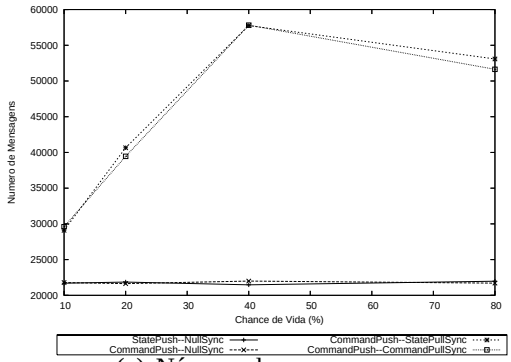


(e) Qualidade por intensidade de escrita para diferentes chances de vida (CV) e 4 réplicas

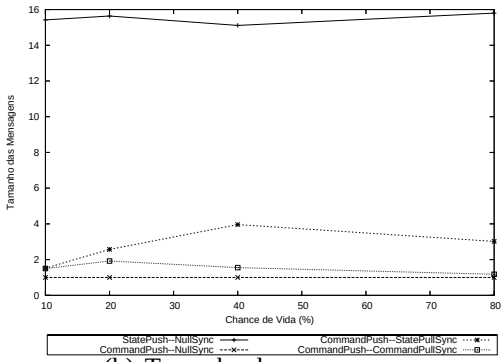


(f) Qualidade por intensidade de escrita para diferentes números de réplicas e chance de vida de 40%

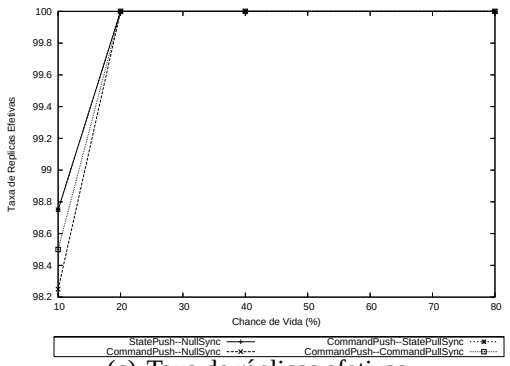
Figura 3. Qualidade das réplicas efetivas no protocolo StatePush–NullSync



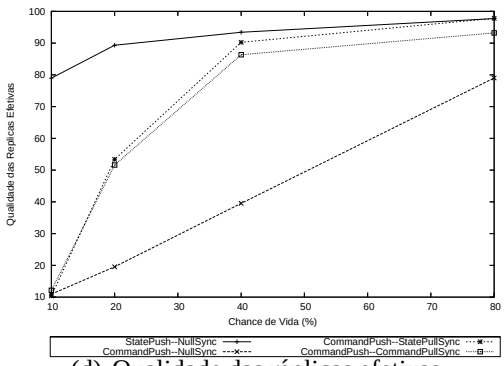
(a) Número de mensagens



(b) Tamanho das mensagens



(c) Taxa de réplicas efetivas



(d) Qualidade das réplicas efetivas

Figura 4. Comparação dos quatro protocolos por chance de vida para quatro réplicas e intensidade de escrita de 40%