Avaliação Prática dos Detectores de Defeitos e sua Influência no Desempenho das Operações de Consenso

Luiz Angelo Steffenel, Ingrid Eleonora Schreiber Jansch-Pôrto

Avaliação Prática dos Detectores de Defeitos e sua Influência no Desempenho das Operações de Consenso

Luiz Angelo Barchet Estefanel
Ingrid Jansch-Pôrto (orientadora)
angelo@inf.ufrgs.br

Resumo

As operações de coordenação em ambientes puramente assíncronos sujeitos a falhas sofrem problemas devido à impossibilidade de distingüir processos falhos de processos muito lentos. Detectores de defeitos representam uma das formas mais eficientes e gerais de circundar tal problema. Entretanto, a diversidade de modelos de detectores sugeridos na bibliografia não foi acompanhada de um estudo comparativo entre as propostas, e até hoje a escolha de um ou outro modelo baseia-se em suposições sem a devida comprovação. O presente trabalho pretende fazer tal comparação, tomando por base um trabalho mais restrito identificado na literatura, mas procurando levantar o máximo possível de características que possam auxiliar a escolha do modelo de detector mais adequado a cada situação.

1. INTRODUÇÃO

Os sistemas distribuídos, devido às suas características inerentes, representam uma maneira de aumentar a confiabilidade e disponibilidade do sistema nos casos de falhas, pois permitem que a operação continue mesmo durante o colapso ou falha de alguns componentes. Entretanto, atingir essa tolerância a falhas não é fácil, pois envolve o uso de técnicas complexas para solucionar os problemas originados na própria distribuição como, por exemplo, a inconsistência das decisões.

Quanto mais genérico um sistema distribuído, mais ampla é a sua possibilidade de adaptação aos diversos ambientes de operação, e os sistemas distribuídos puramente assíncronos representam o modelo mais genérico utilizado. Há, no entanto, algumas restrições nesse modelo que aumentam a complexidade das operações nele realizadas, sobretudo na possibilidade de falhas. Por exemplo, é conhecida a incapacidade de garantir o fechamento de uma operação de consenso, uma das mais significativas operações de acordo entre processos, em ambientes assíncronos sujeitos a falhas. Isso se deve à impossibilidade de determinar se um processo encontra-se falho ou apenas mais lento que os demais. Essa restrição, conhecida como Impossibilidade FLP (em homenagem aos seus autores, Fischer, Lynch e Paterson), ocasiona que, devido à falta de conhecimento sobre o sistema, o algoritmo de consenso deve esperar a resposta dos demais processos, sob risco de tornar as decisões inconsistentes. No caso da falha de um integrante do consenso, o algoritmo irá esperar indefinidamente. Essa restrição não afeta somente o consenso; muitas outras operações freqüentemente utilizadas, como a votação e a difusão atômica, são tão ou mais complexas que o consenso, e ficam sujeitas à mesma restrição de operação.

Existem algumas técnicas que objetivam contornar tais restrições, como por exemplo o sincronismo parcial e o assincronismo temporizado [1]. Tais propostas, entretanto, servem apenas a situações específicas, não contemplando as diversas possibilidades do sistema. Chandra [2] propôs um modelo de assincronismo auxiliado por um detector de defeitos, que demonstrou ser mais abrangente e adaptável que os demais [1]. Tais detectores funcionam através da manutenção de uma lista contendo os processos suspeitos de falhas, baseada em mensagens trocadas entre os detectores. Embora os detectores de defeitos tampouco possam determinar com exatidão o estado dos processos do sistema, sua utilização aumenta a quantidade de conhecimento sobre os demais elementos, auxiliando na finalização das operações de consenso.

Chandra definiu os detectores de defeitos através de duas propriedades: *completeness* (abrangência, completeza) e *accuracy* (precisão). A propriedade *completeness* refere-se à capacidade do detector identificar

todos os processos que estão falhos, enquanto a propriedade *accuracy* determina a precisão desta suspeita, a fim de evitar a inclusão de processos corretos nas listas de suspeitos. Ao respeitar essas duas propriedades, um detector de defeitos garante que os algoritmos que o utilizem não perderão a consistência das decisões, nem ficarão indefinidamente bloqueados (preservando as propriedades *safety* e *liveness*, respectivamente).

A definição dos detectores, no entanto, não os vincula a nenhum mecanismo específico de implementação, pois a essência de seu funcionamento baseia-se no cumprimento das propriedades de *completeness* e *accuracy*. Dessa forma, foram propostos diversos algoritmos detectores de defeitos que diferem sob diversos aspectos, tanto nas características de construção e comunicação quanto nos modelos de falhas considerados. Esta diversidade, conforme constatado em trabalho anterior [3], não foi devidamente acompanhada por uma avaliação de desempenho ou confiabilidade, de forma que os desenvolvedores normalmente escolhem entre os diversos algoritmos baseando-se apenas em observações sobre o possível funcionamento dos detectores. Por exemplo, Felber avalia os detectores de defeitos segundo estimativas de desempenho e número de mensagens, e aconselha o uso de um ou outro método de acordo com as exigências dos projetos. Essas considerações limitaram-se à abordagem conceitual da matéria, não tendo sido apresentada qualquer verificação prática do impacto dos modelos sobre os sistemas. De fato, como mostra Guerraoui quando avalia algoritmos de consenso [1], nem sempre o algoritmo que parece mais ágil corresponde às expectativas.

2. PONTO DE PARTIDA

Para a escolha de quais os aspectos dos detectores de defeitos a serem comparados, foi feita uma verificação na bibliografia, procurando algum trabalho que já tivesse realizado uma avaliação dos detectores. A grande maioria dos trabalhos sobre detectores de defeitos concentra-se na sugestão de modelos ou na adaptação destes para diversos ambientes de falhas, tendo sido encontrado apenas um trabalho que realmente faz uma comparação. Este trabalho, publicado por Sergent *et al*. [4], descreve a influência de diversos tipos de detectores de defeitos no desempenho de uma operação de consenso, amparando suas conclusões em resultados de simulações. Além das restrições associadas à técnica escolhida, tal trabalho limitou-se às observações de desempenho no algoritmo de consenso. Muitas das questões que envolvem a comparação entre os detectores como, por exemplo, a influência da quantidade de informações trocadas entre os detectores e a influência dos detectores em outros sistemas foram deixadas sem resposta.

Não obstante as limitações da técnica escolhida e do escopo de estudo, o trabalho de Sergent demonstrou ser interessante como ponto de partida para uma comparação mais abrangente entre os detectores de defeitos. Assim, o presente trabalho pretende tomar como base os resultados obtidos nas simulações de Sergent, comparando-os com resultados obtidos através da avaliação das implementações dos detectores. Além disso, serão avaliados outros aspectos do funcionamento dos detectores de defeitos, a partir de questionamentos sugeridos no Trabalho Individual [3] ou observados durante a elaboração das experiências.

3. DETECTORES DE DEFEITOS

O conjunto dos modelos de detectores de defeitos utilizados foi definido a partir dos detectores avaliados por Sergent [4], com a adição de outros modelos presentes na literatura considerados interessantes.

Os detectores considerados por Sergent são os seguintes: detectores *Push*, detectores *Pull*, *ad hoc "no message"* e *ad hoc "heart-beat"*. Os dois primeiros modelos podem ser implementados como módulos independentes, e representam os detectores freqüentemente citados na literatura; os dois últimos são implementações otimizadas e que precisam estar inseridas no algoritmo de consenso. Além dos detectores utilizados por Sergent, este trabalho irá avaliar os detectores *Push-adaptativo* e *Pull-adaptativo* — variações dos modelos *Push* e *Pull* que não utilizam *timeouts* fixos para a elaboração da lista de suspeitos [2], e o detector *Heartbeat* [5], que apresenta características interessantes para a análise, como o mecanismo de julgamento das suspeitas, a utilização de mensagens com mais informações e o conceito de vizinhos para a disseminação das mensagens. O

detector *Heartbeat* já havia sido implementado no Trabalho Individual, mas a estrutura da sua construção (Figura 1) provou ser eficiente também para a implementação dos outros detectores "modulares", tornando a construção de detectores com características diferentes mais fácil e padronizada.

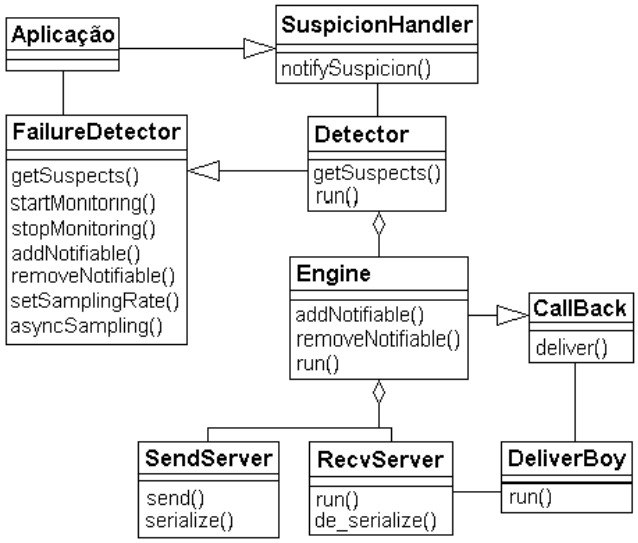

Figura 1. Estrutura de um dos detectores de defeitos implementados

4. O ALGORITMO DE CONSENSO

Para a implementação do módulo de consenso, foi escolhido o algoritmo de consenso com rotação de coordenadores apresentado por Chandra [2], uma vez que este algoritmo foi especificado para ambientes com as mesmas características dos detectores de defeitos (funciona com detectores $\langle\rangle\mathcal{S}$ e $\langle\rangle\mathcal{W}$) e foi o algoritmo utilizado no trabalho de Sergent.

Basicamente, o algoritmo de consenso realiza os seguintes passos:

- na primeira fase, os processos enviam suas propostas iniciais ao coordenador;
- na segunda fase, o coordenador espera que a maioria dos processos tenham sugerido um valor, e toma uma decisão entre estes valores sugeridos;
- na terceira fase, os processos esperam a decisão do coordenador. Eles adotam este valor e enviam uma resposta afirmativa (ACK). No entanto, se for detectado que o coordenador falhou antes que sua decisão chegasse ao processo, este envia uma resposta NACK, partindo para a próxima rodada do consenso;
- na quarta fase, o coordenador espera até receber mensagens da maioria dos processos. Se esta maioria for de mensagens ACK, então o coordenador faz um *broadcast* confiável, validando a decisão.

Atualmente, o consenso já foi implementado e encontra-se em fase de testes, e problemas encontrados na transcrição do algoritmo de Chandra para uma implementação real mereceram destaque junto à elaboração do trabalho. Um destes problemas refere-se ao avanço dos passos do algoritmo, uma vez que este considera um nível de sincronização entre os processos incoerente com a própria definição dos sistemas distribuídos assíncronos. Este problema ocorre quando participam do consenso mais processos do que o quorum mínimo para a decisão: assim que atingir este ponto de tomada de decisão, o coordenador imediatamente envia sua proposta, mas os processos que ainda estão executando o passo anterior não compreendem estas novas mensagens, e acabam não participando do consenso, contrariando a própria exigência do algoritmo de que todos os processos corretos recebam e acatem a decisão do consenso. O segundo problema encontrado refere-se aos pontos críticos onde a operação do consenso pode ser abortada. O algoritmo define apenas um ponto crítico por falha do coordenador logo após ter enviado sua decisão, e neste caso há o tratamento necessário para que uma nova rodada de consenso seja iniciado. Entretanto, se o coordenador falhar imediatamente antes de enviar o *broadcast* confiável, todos os demais processos irão esperar por tempo indeterminado a entrega da mensagem do coordenador.

Para o primeiro problema, a solução é relativamente fácil e não altera a funcionalidade do sistema, pois trata-se apenas da permissão para que o algoritmo "pule" alguns passos de acordo com as mensagens recebidas do coordenador e dos demais processos. Já o segundo problema é mais complexo, pois entra no mérito da precisão da detecção e do efeito de um cancelamento na consistência entre os processos. Optou-se por não alterar o algoritmo neste último ponto, restringindo-se ao mesmo conjunto de testes suportados pelo trabalho de Sergent, e deixando a discussão das possíveis soluções para o texto da dissertação.

5. ADEQUAÇÃO DOS PARÂMETROS

Os resultados obtidos por Sergent em suas simulações derivaram diretamente dos parâmetros considerados como, por exemplo, a velocidade de transmissão na rede que interliga as máquinas. Para evitar que a comparação entre os resultados da simulação e da experiência prática seja distorcida devido às características diferentes dos ambiente de testes, foi realizada a avaliação dos parâmetros inerentes ao ambiente de testes real, para que possam também ser comparados aos parâmetros da simulação.

Ao avaliar o ambiente prático, verificou-se que as taxas de transmissão de mensagens e sua influência no desempenho dos sistemas de suporte à aplicação – o esgotamento dos *buffers* do sistema operacional é um exemplo – representam a grande barreira que limita o desempenho dos detectores de defeitos. Estes fatores não estão presentes na experiência simulada – por exemplo, Sergent considerou na simulação que não há custo de processamento, o que não condiz com a prática. Para permitir a melhor avaliação possível dos detectores, foi feito o levantamento das características de comunicação no ambiente real e determinados os pontos de melhor desempenho, que deverão ser usados nas tomadas de desempenho.

6. CONCLUSÕES

Os resultados obtidos até o momento demonstram que não só os objetivos principais do trabalho serão atingidos, no caso, a comparação de desempenho dos detectores, como também a descrição de diversos problemas encontrados. Estes casos não são discutidos nas publicações da área, embora representem um material necessário ao desenvolvimento e implementação de sistemas distribuídos tolerantes a falhas. A discussão sobre algumas destas questões já levou à elaboração de artigos aprovados em evento da área, e tudo leva a crer que mesmo após o término do trabalho de dissertação restarão vários pontos interessantes para serem investigados.

AGRADECIMENTOS

Agradeço ao CNPq pelo auxílio financeiro fornecido, e à minha orientadora Profa. Ingrid, ao colega Raul Ceretta Nunes e aos demais integrantes do Grupo de Tolerância a Falhas da UFRGS pela ampla e valiosa discussão dos assuntos do trabalho.

REFERÊNCIAS

[1] R. Guerraoui, A. Schiper. Consensus: The Big Misunderstanding. In: IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS'97). Proceedings. Out. 1997.

[2] T. D. Chandra, S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. Journal of the ACM. Vol. 43, No. 2, Mar. 1996. pp 225-267.

[3] L.A.B.Estefanel. Detectores de Defeitos Não Confiáveis.PPGC/UFRGS, Jan. 2000. Trab. Indiv. nº 880

[4] N. Sergent, X. Défago, A. Schiper. Failure Detectors: implementation issues and impact on consensus performance. Lausanne: EPFL, Technical Report, Mai 1999.

[5] M.K.Aguilera, W.Chen, S.Toueg. Heartbeat: a timeout-free failure detector for quiescent reliable communication. In: 11th Intl. Workshop on Distributed Algorithms. Proceedings. Set. 1997.