Comparação de Desempenho de Algoritmos de Recuperação Síncrono e Assíncrono

Sérgio Luis Cechin Ingrid Jansch-Pôrto
{cechin, ingrid}@inf.ufrgs.br
Curso de Pós-Graduação em Ciência da Computação
Instituto de Informática - UFRGS
Caixa Postal 15064 - CEP 91501-970
Porto Alegre - RS - Brasil

Resumo

A recuperação de processos por retorno pode ser implementada seguindo paradigmas síncrono ou assíncrono. Pretende-se, neste artigo, apresentar alguns resultados teóricos da comparação de desempenho entre dois algoritmos das categorias citadas, tomando-se por base os algoritmos de Koo e Toueg (síncrono) e o de Juang e Venkatesan (assíncrono). O objetivo da comparação é demonstrar que as vantagens e desvantagens relativas dependerão das características das aplicações.

Abstract

In distributed systems, backward recovery has two main implementation paradigms: synchronous and asynchronous. In this paper, we intend to compare two representative algorithms on these groups and to present some theoretical results. Koo & Toueg synchronous algorithm and Juang & Venkatesan asynchronous one are considered. Our goal is to demonstrate that the advantages and disadvantages between them are related to the characteristics of the applications.

1 Introdução

A recuperação de processos em sistemas distribuídos apresenta algumas dificuldades devidas à características do próprio ambiente: o sistema é formado por um conjunto de processos separados fisicamente sem o compartilhamento de memória ou de relógio (*clock*). Estes processos trocam informação exclusivamente através de mensagens [JAL94]. Na literatura, existem diversos trabalhos dedicados à apresentação de algoritmos destinados a realização de recuperação de processos neste contexto de sistemas. O presente trabalho objetiva comparar, através de modelos teóricos desenvolvidos pelo próprio autor [CEC98], dois algoritmos representativos dos enfoques síncrono e assíncrono, de forma a comprovar vantagens e desvantagens conceitualmente comentadas na literatura como parte da apresentação destes algoritmos.

Os algoritmos são aplicados a sistemas cujo comportamento baseia-se no que segue: o canal de comunicação é ideal com *buffers* infinitos, livres de erros e que entregam as mensagens na mesma ordem de envio [JAL94]; as falhas são temporárias e os processos atuam em *fail-stop*; e é admitida a ocorrência de mensagens perdidas e mensagens órfãs.

Para fins de análise, o tempo total de cada mensagem, tm - desde o envio por um processo até o recebimento pelo processo destino - foi dividido em três etapas: o "empacotamento", t_{me}; tempo para percorrer o canal de comunicação, t_{mt}; e o "desempacotamento", t_{md}.

2 Características do algoritmo síncrono

O algoritmo de Koo e Toueg [KOO87] baseia-se na determinação de pontos de recupe-

ração (PR) por coordenação entre os processos. Durante o processo de estabelecimento dos PRs, apenas as mensagens de controle do procedimento podem transitar pelo canal. Desta forma, cada conjunto de PRs locais (de cada processo) forma uma **linha consistente de recuperação**, possibilitando o descarte dos PRs anteriormente estabelecidos.

Mesmo tendo desaparecido a falha (temporária, por hipótese), um dos processos detecta a existência de erros resultantes e inicia o processo de recuperação. Este processo resume-se em atualizar os dados e o processamento a partir do último (e único) PR registrado.

O algoritmo utiliza um protocolo de duas fases (*two phase commit protocol*) para estabelecer um PR e retornar a um PR. Desta forma, uma tentativa de estabelecimento de um PR pode vir a falhar, sendo adiada; no caso do retorno, o algoritmo não adia o procedimento, ficando bloqueado até conseguir o retorno de todos os processos.

Deve-se notar que o mecanismo de recuperação envolve o descarte de uma porção de processamento, mais precisamente aquela compreendida entre o estabelecimento do último PR e a detecção da ocorrência da falha. Este fato implica queda de desempenho.

A escolha da periodicidade de estabelecimentos dos PRs inadequada aos parâmetros de falhas do sistema tem repercussões sobre o desempenho. Se o período entre o estabelecimento dos PRs for muito grande, em média muito processamento será desfeito a cada falha, levando a um baixo desempenho na ocorrência freqüente de falhas; caso o período seja muito pequeno, o sistema gastará muito tempo coordenando a tomada dos PRs, reduzindo significativamente o desempenho também, mesmo em operação normal (sem falhas).

3 Características do algoritmo assíncrono

No caso do algoritmo de Juang e Venkatesan [JUA91], não há coordenação entre os processos para a tomada de um PR. Cada processo, após o recebimento de uma mensagem, salva um PR em memória volátil (que pode ser perdido em caso de falha) e, de forma periódica, transfere estes dados para a memória estável (protegida contra falhas). O estabelecimentos destes PRs locais evita que a comunicação normal precise ser suspensa. Além disso, não existem as trocas de mensagens necessárias a coordenação.

Entretanto, quando o sistema detecta um erro, antes de iniciar o processo de retorno propriamente dito, deve ser encontrada uma linha consistente de recuperação entre os vários PRs locais armazenados nos processos: se o processo tiver falhado, deverão ser utilizados os dados da memória estável; em caso contrário, podem ser usados os dados da memória volátil.

O algoritmo assíncrono não está livre do "efeito dominó", mas é limitado ao último armazenamento em memória estável do processo que falhou. Entretanto, como as ações de salvamento não são difundidas aos demais processos, devem ser mantidos em memória os PRs obtidos desde que a aplicação foi iniciada.

Na literatura, é sustentado que, em operação normal, a queda de desempenho é relativamente menor no uso do algoritmo assíncrono se comparado ao síncrono; a situação inverte-se quando ocorrem falhas.

4 Suposições e princípios gerais

Algumas restrições foram necessárias para simplificar a análise de desempenho; outras foram necessárias para compatibilizar a operação dos dois algoritmos:

- não foi considerado o mecanismo recursivo proposto por Koo e Toueg no estabelecimento dos PRs e no retorno a um PR;
- foi considerado que as falhas ocorrem de forma periódica, sendo o período médio designado por TBF (*Time Between Faults*);
- todos os processos ativos serão considerados no procedimento de tolerância a falhas,

mesmo que não tenham trocado mensagens com o grupo de processos que falhou;

- não ocorrem falhas durante a execução dos algoritmos (tomada de PRs e retorno).

O *desempenho* calculado fornecerá o percentual do tempo utilizado para o processamento da aplicação sobre o tempo total gasto, o qual corresponde à soma dos tempos gastos pela aplicação e nas atividades de tolerância a falhas. Esta equação do desempenho é:

$$Desempenho = \frac{(TempoTotal - TempoTotalToleranciaFalhas)}{TempoTotal}$$

Como, por hipótese, as falhas ocorrem de forma periódica, o *desempenho* calculado a partir dos tempos totais será, em média, o mesmo obtido utilizando-se os tempos de um período entre falhas. Ou seja, o *desempenho* ou a esperança do *desempenho relativo* (DR) será:

$$E(DR) = E\left(\frac{(TBF - TempoToleranciaFalhas)}{TBF}\right) = 1 - \frac{E(TempoToleranciaFalhas)}{TBF}$$

O tempo gasto com as atividades de tolerância a falhas foi dividido em duas etapas: o estabelecimento dos PRs e o retorno a um PR, as quais dependem do algoritmo considerado.

5 Desempenho relativo do algoritmo síncrono

Considerando que T_{CP} é a periodicidade da tomada dos pontos de recuperação, T_{PR} é o tempo gasto para a tomada de um PR e T_{REC} o tempo gasto no processo de retorno, a equação que descreve o desempenho relativo do algoritmo síncrono é a seguinte:

$$E(DR) = \left(1 - \frac{E(T_{PR})}{T_{CP}}\right)\left(1 - \frac{E(T_{REC})}{TBF}\right)$$

Esta equação é formada por duas parcelas que contribuem para a queda no desempenho: uma devida ao tempo gasto no estabelecimento dos PRs e a outra devida ao processo de retorno a um PR. O tempo gasto no estabelecimento dos PRs é calculado pela equação:

$$E(T_{PR}) = T_{FIXO} + \left[P_{PR} \times E(T_{VAR}^{OK}) + (1 - P_{PR}) \times E(T_{VAR}^{NOK})\right],$$

onde T_{FIXO} é o tempo gasto no estabelecimento de qualquer PR, P_{PR} é a probabilidade de obter-se um PR, T_{VAR}^{OK} é o tempo adicional gasto quando é obtido um PR e T_{VAR}^{NOK} o tempo adicional gasto quando não é obtido o PR.

As parcelas componentes da esperança do tempo gasto no estabelecimento dos PRs podem ser calculadas pelas seguintes equações, onde N é o número de processos:

$T_{FIXO} = 3 \cdot t_m + (2 \cdot N - 2) \cdot t_{me}$, quando não há suporte para *broadcast* de mensagens;

$T_{FIXO} = 3 \cdot t_m + (3 \cdot N - 4) \cdot t_{me}$, quando há suporte para *broadcast* de mensagens;

$T_{VAR}^{OK} = T_{VAR}^{NOK} = T_{PRP}$, onde T_{PRP} é o tempo que um processo gasta para salvar os dados de um ponto de recuperação. Na realidade, T_{VAR}^{OK} e T_{VAR}^{NOK} não são iguais, mas a diferença entre eles é pequena; o tempo T_{VAR}^{OK} é maior pois inclui a ativação do PR na memória estável.

Para o tempo gasto no processo de retorno, tem-se a equação:

$$E(T_{REC}) = T_{CP} \times \left(\frac{2 - P_{PR}}{2 \times P_{PR}}\right) + T_{DET} + \frac{T_{FIXO}}{P_{RET}} + \left[\left(\frac{1 - P_{RET}}{P_{RET}}\right) \times T_{VAR}^{NOK} + T_{VAR}^{OK}\right],$$

onde P_{RET} é a probabilidade de que uma tentativa de retorno seja bem sucedida; T_{DET} é o tempo entre a manifestação da falha e sua detecção; T_{VAR}^{OK} é o tempo gasto em uma tentativa de retorno bem sucedida e T_{VAR}^{NOK} em tentativas sem sucesso (deve haver nova tentativa).

Os resultados obtidos são mostrados na forma de gráficos. Na figura 1, são apresentadas as curvas de desempenho em função (a) da periodicidade das falhas, *TBF* e (b) da periodicidade da tomada dos PRs, T_{CP}.

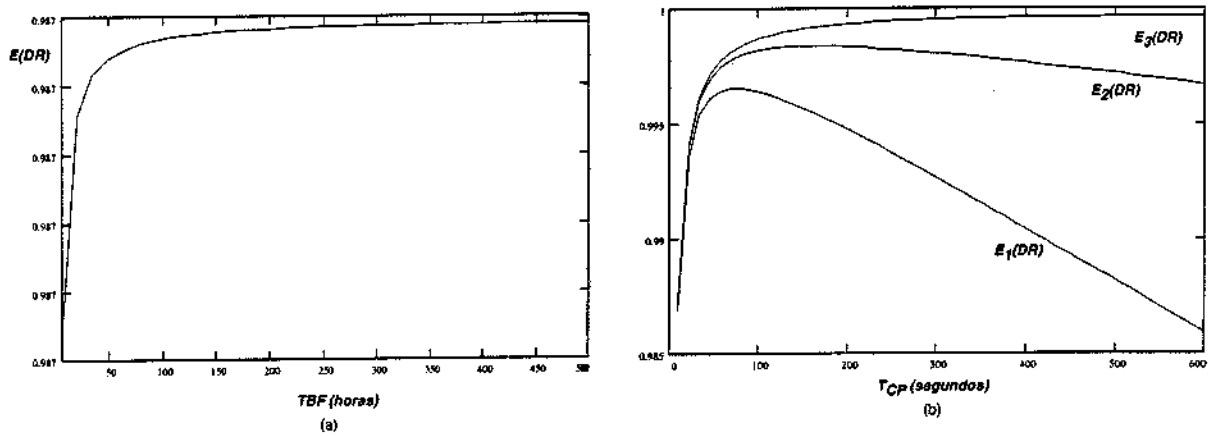

Figura 1 - Curvas de desempenho do algoritmo síncrono

O desempenho cresce, de forma assintótica, com o aumento de TBF. O valor da assíntota pode ser calculado pela aplicação de limite à equação de desempenho:

$$\lim_{TBF \to \infty} E(DR) = 1 - \frac{E(T_{PR})}{T_{CP}}$$

Na figura 1 foram apresentadas três curvas de desempenho. Cada curva foi traçada usando um valor diferente de *TBF*. A curva $E_1(DR)$ corresponde ao menor valor de *TBF* e a curva $E_3(DR)$ é aquela com o maior valor de *TBF*. Todas estas curvas apresentam um ponto de máximo, o qual pode ser calculado igualando a zero a derivada da equação de desempenho. Como resultado obtém-se:

$$T_{CP\,max} = \sqrt{\left[E(T_{PR}) \times (TBF - K2)\right] / K1}$$

$$\text{onde } K1 = \left(\frac{2 - P_{PR}}{2 \times P_{PR}}\right) \text{ e } K2 = T_{DET} + \frac{T_{FIXO}}{P_{RET}} + \left[\left(\frac{1 - P_{RET}}{P_{RET}}\right) \times T_{VAR}^{NOK} + T_{VAR}^{OK}\right].$$

6 Desempenho relativo do algoritmo assíncrono

Da mesma forma que a equação de desempenho do algoritmo síncrono, aparecem duas parcelas na equação de desempenho: uma devida ao processo de estabelecimento dos PRs e outra devida ao retorno a um PR, conforme equação que segue:

$$E(DR) = \left(1 - \frac{T_{PRV}}{1/\lambda} - \frac{T_{PRP}}{T_{CP}}\right)\left(1 - \frac{E(T_{REC})}{TBF}\right),$$

onde T_{PRV} corresponde ao tempo que o processo gasta para salvar o seu estado em memória volátil, T_{PRP} corresponde ao tempo gasto para salvar o estado do processo em memória estável e λ é a taxa média de recebimento de mensagens.

Na equação de *desempenho relativo*, pode-se notar que o tempo gasto nas atividades que degradam o desempenho aparecem divididas pela periodicidade que ocorrem: tempo gasto na escrita em memória volátil pelo tempo entre mensagens; tempo gasto na escrita em memória estável pelo tempo entre salvamentos de PRs; tempo gasto no retorno a um PR pela periodicidade das falhas. Os tempos gastos no salvamento do estado do processo em memória volátil ou memória estável são dependentes do hardware das memória envolvidas. Entretanto,

quanto maior for a relação entre o tempo de acesso à memória estável e o tempo de acesso à memória volátil, tanto melhor pode-se esperar que seja o desempenho.

O tempo gasto no processo de retorno pode ser calculado pela seguinte equação:

$$E(T_{REC}) = \frac{T_{CP}}{2} + T_{DET} + T_{BRC} + N \times (T_{TM} + T_{PM}),$$

onde T_{DET} é o tempo entre a manifestação da falha e a detecção deste fato; T_{BRC} é o tempo gasto para que o processo que falhou informe este fato aos outros processos do sistema; T_{TM} é o tempo gasto em uma rodada de busca de uma linha consistente de recuperação; T_{PM} é tempo gasto para processar as mensagens em uma destas rodadas. Como pode ser observado na equação, são necessárias N rodadas para que seja obtida a linha de recuperação. A equação do cálculo de T_{TM} é a seguinte:

$$T_{TM} = N \times [t_m + (N-2) \times t_{me}]$$

A representação gráfica das equações obtidas para o desempenho relativo pode ser vista na figura 2, mostrando a relação entre o desempenho relativo e (a) a periodicidade das falhas, *TBF*, e (b) a periodicidade da tomada dos PRs, T_{CP}. As curvas em 2(b) foram traçadas para valores diferentes de *TBF*: $E_1(DR)$ para o menor valor e $E_3(DR)$ para o maior valor.

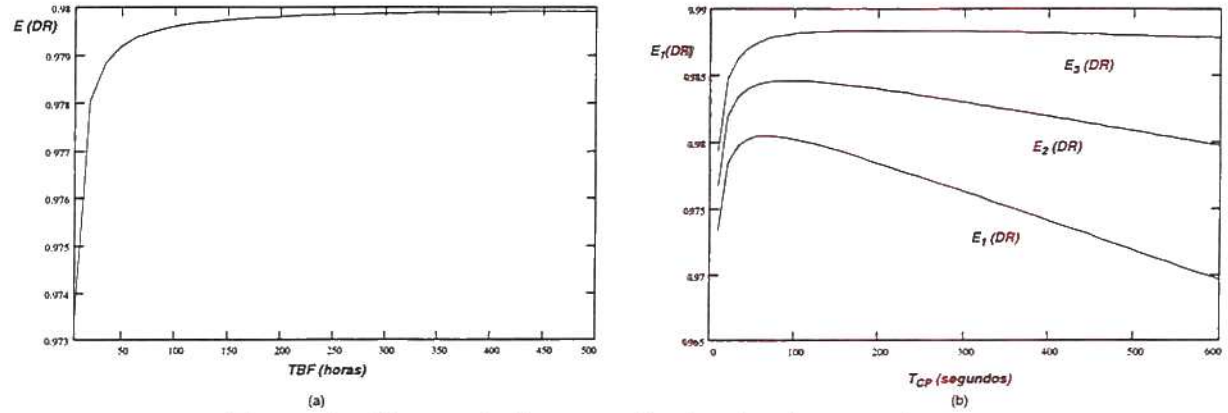

Figura 2 - Curvas de desempenho do algoritmo assíncrono

A forma geral das curvas de desempenho é a mesma que as do algoritmo síncrono. O desempenho cresce de forma assintótica, quando *TBF* aumenta, para um valor que pode ser calculado pelo limite:

$$\lim_{TBF \to \infty} E(DR) = 1 - \frac{T_{PRV}}{1/\lambda} - \frac{T_{PRP}}{T_{CP}}.$$

Através deste resultado pode-se verificar que, mesmo com uma baixa taxa de falhas (*TBF* grande), o desempenho não poderá passar de um valor máximo. Este valor depende do tempo gasto para salvar um PR: volátil e estável; e da periodicidade destes salvamentos: taxa de recebimento de mensagens e taxa de transferência para a memória estável.

A curva de desempenho em função de T_{CP} apresenta um ponto de máximo. Este pode ser calculado igualando-se a zero a derivada da função de desempenho:

$$T_{CP\,max} = \sqrt{[T_{PRP} \times (TBF - K2)] / \left[K1 \times \left(1 - \frac{T_{PRV}}{1/\lambda} \right) \right]}$$

onde $K1 = \frac{1}{2}$ e $K2 = T_{DET} + T_{BRC} + N \times (T_{TM} + T_{PM})$.

7 Análise e Conclusões

A comparação das equações obtidas levaram às conclusões a seguir relatadas.

Para ambos algoritmos, a medida que a taxa de falhas diminui, o fator básico determinante do desempenho é o período da tomada dos PRs (TCP). Entretanto, o desempenho do algoritmo assíncrono piora com o aumento do número de mensagens recebidas, λ.

O aumento do número de processos é mais prejudicial ao desempenho no algoritmo assíncrono; as equações apresentam um fator de redução de desempenho N^3 para o algoritmo assíncrono e N para o síncrono.

Como os processos comunicam-se só através de mensagens, os tempos destas foram considerados. Seus efeitos aparecem em todas as etapas do algoritmo síncrono e na recuperação do algoritmo assíncrono, onde aparece multiplicada pelo fator N^3.

A existência de suporte para *broadcast* (ou uma arquitetura que o privilegie) é importante no estabelecimento dos PRs do algoritmo síncrono e na recuperação do assíncrono.

Além dos exposto, a taxa de recebimento de mensagens só influencia o desempenho do algoritmo assíncrono e pode ser decisivo na escolha do algoritmo. Esta comparação pode ser observada na figura 3, onde foram traçadas as curvas de desempenho para os algoritmos síncrono ($E_S(DR)$) e assíncrono com λ=1 ($E_{A1}(DR)$) e λ=5 ($E_{A2}(DR)$).

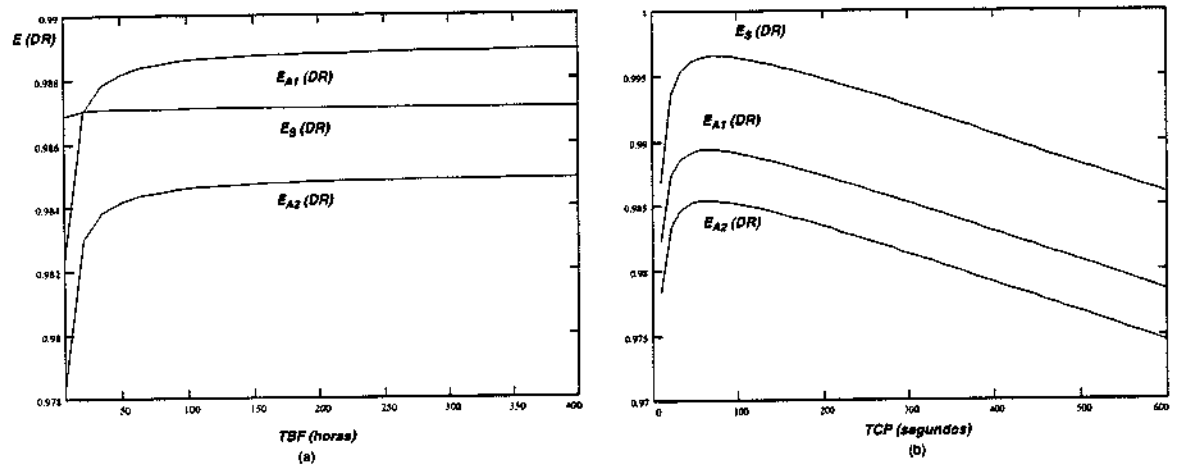

Figura 3 - Curvas de desempenho comparativas

Apesar dos resultados coincidirem com as previsões apresentadas na literatura, é importante que sejam validados através da implementação ou da simulação dos algoritmos, o que ainda não foi feito.

O estudo aprofundado dos algoritmos e das hipóteses adotadas para a modelagem matemática, bem como a dedução de todas as equações, estão disponíveis através da monografia [CEC98] preparada como parte das atividades preliminares ao doutorado no CPGCC.

Referências

[CEC98] Cechin, S. L. Avaliação teórica do desempenho de algoritmos de recuperação por retorno do tipo síncrono e assíncrono. CPGCC da UFRGS. 1998.

[JUA91] Juang, T.; Venkatesan, S. Crash Recovery with Little Overhead. Int'l. Conf. on Distributed Computing Systems. Proceedings. May 1991. Pp.454-461.

[JAL94] Jalote, P. *Fault Tolerance in Distributed Systems*. New Jersey: Prentice-Hall, 1994.

[KOO87] KOO, R; TOUEG, S. Checkpointing and Rollback-Recovery for Distributed Systems. *IEEE Trans. on Software Engineering*, v.SE-13(1):23-31, Jan. 1987.