Cortes Consistentes em Aplicações Distribuídas

Islene Calciolari Garcia
Luiz Eduardo Buzato

Relatório Técnico IC–98-17

Abril de 1998

Cortes Consistentes em Aplicações Distribuídas*

Islene Calciolari Garcia
Luiz Eduardo Buzato

Resumo

A avaliação de predicados sobre estados globais consistentes é requerida em uma vasta classe de soluções para problemas em sistemas distribuídos. A construção de cortes consistentes de maneira assíncrona permite maior eficiência e flexibilidade a sistemas de monitorização. Consideramos a abordagem assíncrona para aplicações construídas sobre o Modelo de Processos e Mensagens e sobre o Modelo Objetos e Ações. Propomos uma arquitetura de *software* para sistemas de monitorização e analisamos a construção progressiva de *checkpoints* globais consistentes.

Abstract

The evaluation of predicates against consistent global states is required to solve a large class of problems in distributed systems. The asynchronous construction of consistent cuts allows greater efficiency and flexibility to monitoring systems. We consider the asynchronous approach for applications built upon the Process and Message Model and upon the Object and Action Model. We propose a software architecture for monitoring systems and analyze the progressive construction of consistent global checkpoints.

1 Introdução

A avaliação de predicados sobre estados globais consistentes [6] é requerida em uma vasta classe de soluções para problemas em sistemas distribuídos, como detecção de *deadlocks*, perda de ficha, coleta de lixo, recuperação com retrocesso, além de monitorização e reconfiguração em geral [2]. A construção assíncrona de cortes consistentes é menos intrusiva e permite uma maior eficiência e flexibilidade a sistemas de monitorização.

Aplicações distribuídas são freqüentemente construídas utilizando-se o Modelo de Processos e Mensagens (MPM) e existem vários algoritmos na literatura para a detecção assíncrona de cortes consistentes nesse modelo [11].

Aplicações confiávies podem ser construídas utilizando-se o Modelo de Objetos e Ações Atômicas (MOA) [16]. Um estado global consistente de uma aplicação construída sobre o MOA pode ser obtido facilmente através de uma ação atômica que adquira *locks* de leitura sobre todos os objetos da aplicação. A construção assíncrona de cortes consistentes no MOA [8] também é possível e pode ser feita através do mapeamento de algoritmos desenvolvidos para o MPM.

*Instituto de Computação, UNICAMP, Caixa Postal 6176, 13083-970 Campinhas, SP. Este trabalho recebe apoio financeiro da FAPESP, processo número 95/1983-8 para Islene Calciolari Garcia e número 96/1532-9 para o Laboratório de Sistemas Distribuídos do Instituto de Computação/UNICAMP.

A monitorização de aplicações distribuídas levando-se em consideração todos os estados da aplicação é extremamente custosa. Para cada processo ou objeto podemos selecionar estados de interesse, denominados *checkpoints*. A escolha arbitrária desses estados, no entanto, não garante a participação destes em um *checkpoint* global consistente [12]. Algoritmos *quase–síncronos* [11] apresentam um compromisso entre a autonomia e obrigatoriedade de se tirar *checkpoints*. Levando em consideração esses algoritmos, este trabalho analisa a construção progressiva de *checkpoints* globais consistentes a serem apresentados ao monitor. Acreditamos que essa abordagem contribui para um melhor entendimento do comportamento de um algoritmo e pode vir a ser útil para o desenvolvimento de novos algoritmos.

Na Seção 2 descrevemos os modelos computacionais MPM e MOA. Na Seção 3 analisamos a consistência de aplicações distribuídas nesses dois modelos e na Seção 4 apresentamos correlações existentes entre o MPM e o MOA, bem como limitações para o mapeamento de algoritmos. A Seção 5 propõe uma arquitetura de *software* para monitorização e a Seção 6 enumera alguns resultados obtidos através da análise da construção progressiva de *checkpoints* globais consistentes. A Seção 7 conclui o artigo.

2 Modelos Computacionais

Uma aplicação distribuída no modelo de processos e mensagens (MPM) é composta por um conjunto de n processos $(p_1, \ldots, p_n)$ que executam de maneira estritamente seqüencial e que se comunicam exclusivamente através de troca de mensagens. A execução de um processo p_i é subdivida em eventos $(e_i^0, e_i^1, \ldots)$ e σ_i^ι representa o estado interno do processo p_i imediatamente após a execução de seu ι-ésimo evento.

Uma aplicação distribuída no modelo de objetos e ações (MOA) é composta por um conjunto de n objetos $(o_1, \ldots, o_n)$ que interagem através de invocação de métodos atômicos (implementados através de ações atômicas). Ações atômicas estão organizadas segundo *estruturas de ações* determinadas pela semântica da aplicação [15]. A estrutura de ações básica é formada por uma única ação atômica. Estruturas mais complexas podem ser formadas utilizando-se combinações seqüenciais ou concorrentes de estruturas de ações atômicas. Consideramos que as ações atômicas são implementadas utilizando o protocolo *two phase commit* [16] e que os objetos são controlados por um *gerente de ações* que se encarrega da distribuição de *locks* [8]. Denominamos σ_i^ι o estado interno do objeto o_i imediatamente após a execução da ι-ésima ação atômica que adquiriu *lock* de escrita sobre ele.

Outro modelo confiável utilizado para construção de aplicações distribuídas é o Modelo de Processos e Conversações [16]. A dualidade entre o MPC e MOA serviu como ponto de partida para as correlações que encontramos entre o MPM e o MOA (Seção 4).

3 Consistência em Aplicações Distribuídas

O conceito de consistência em aplicações distribuídas está fortemente relacionado ao conceito de precedência [14]. Apresentamos a relação de precedência entre eventos no MPM como definida por Lamport [10] e uma relação análoga para a precedência entre ações atômicas no MOA [8].

Definição 3.1 *Um evento e_i^ι precede outro evento e_j^γ ($e_i^\iota \rightarrow e_j^\gamma$) se:*

MPM	**MOA**
execução de processos	estrutura de ações
estados de processos	estados de objetos
eventos	ações atômicas
precedência entre eventos	precedência entre ações

Tabela 1: Correlações entre MPM e MOA

1. *ambos ocorreram no mesmo processo e* $\iota < \gamma$*;*
2. e_i^ι *e* e_j^γ *correspondem, respectivamente, ao envio e recepção de uma mensagem* m*;*
3. *existe um evento* e'' *tal que* $(e \rightarrow e'')$ *e* $(e'' \rightarrow e')$*.*

Definição 3.2 *Uma ação atômica* a *precede outra ação* a' *(*$\mathsf{a} \prec \mathsf{a}'$*) se:*

1. a *deve ocorrer antes de* a' *segundo a estrutura de ações da aplicação;*
2. a *leu um estado gravado por* a'*;*
3. a *leu um estado de objeto* σ_i^ι *e* a' *foi responsável pela gravação de* $\sigma_i^{\iota+1}$*;*
4. *existe uma ação atômica* a'' *tal que* $(\mathsf{a} \prec \mathsf{a}'')$ *e* $(\mathsf{a}'' \prec \mathsf{a}')$*.*

Um corte é consistente no MPM se para todo par de eventos e e e' a seguinte relação é válida:

$$(e \in C) \wedge (e \rightarrow e') \Rightarrow e' \in C$$

De maneira análoga, um corte é consistente no MOA se para todo par de ações atômicas:

$$(\mathsf{a} \in C) \wedge (\mathsf{a}' \prec \mathsf{a}) \Rightarrow (\mathsf{a}' \in C).$$

Em ambos os modelos, podemos construir estados globais consistentes a partir de cortes consistentes. No MPM, utilizamos os estados de processos obtidos após a execução dos eventos na fronteira do corte. No MOA, utilizamos os estados de objeto mais recentes gravados pelas ações atômicas que pertencem ao corte.

4 Mapeamento entre MPM e MOA

A Tabela 1 mostra as correlações que encontramos entre o MPM e o MOA baseando-nos nas relações de precedência e consistência apresentadas na Seção 3. O mapeamento de algoritmos, no entanto, requer uma análise cuidadosa das diferenças entre os dois modelos.

No MPM a relação de precedência coincide com o fluxo de informação entre os processos. Isso permite que vários algoritmos presentes na literatura para cortes consistentes adicionem informações de precedência e controle às mensagens da aplicação [2, 3, 11] e obtenham um canal de comunicação consistente com as relações de precedência. No MOA, é possível a relação de precedência na ausênica de um fluxo de informação direto entre os objetos. O item 2 da Definição 3.2 permite, por exemplo, que uma ação a que adquiriu *lock* de leitura sobre σ_i^ι e que gravou σ_a^α preceda uma ação a' que gravou $\sigma_i^{\iota+1}$ e σ_b^β. Existe uma relação de precedência entre σ_a^α e σ_b^β apesar de não ter havido fluxo de informação entre eles. Para preencher essa lacuna é utilizada a colaboração do gerente de ações

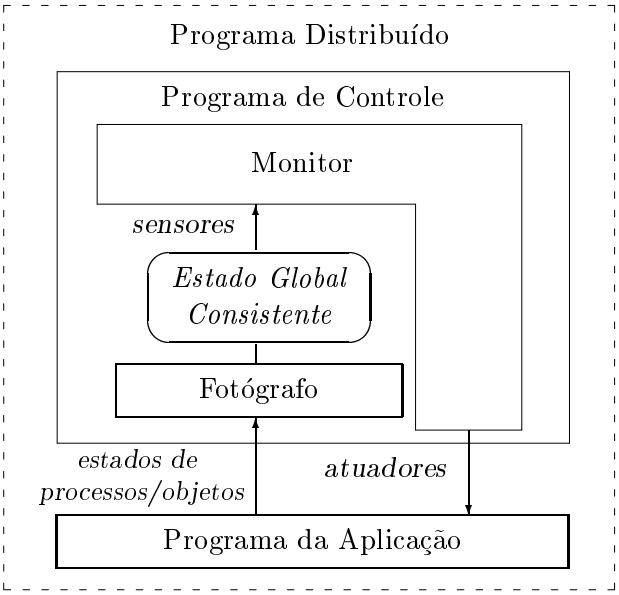

Figura 1: Arquitetura de *Software* para Monitorização

do objeto [8]. O gerente acumula informações das ações que obtiveram *lock* de leitura e as incorpora ao estado do objeto no próximo *lock* de escrita.

Uma mensagem m estabelece uma relação de precedência imediata (não devida à transitividade) entre exatamente dois eventos de processos distintos. Em contrapartida, uma ação atômica `a` pode estabelecer uma relação de precedência imediata entre vários pares de estados de objetos. Tais estados podem não ser apenas os participantes de `a`, mas também estados que participaram em ações atômicas que precederam `a` estruturalmente (dentro de uma mesma estrutura de ações). Isso implica que o conjunto de objetos para os quais um estado σ_i estabeleceu uma relação de precedência imediata não pode ser determinado no momento de sua gravação. Essa característica pode impedir o mapeamento de um algoritmo do MPM para o MOA, como é o caso do algoritmo proposto por Baldoni et al. [3].

A existência deste mapeamento é, portanto, fundamental para a análise e construção assíncrona de cortes consistentes no MOA a partir de algoritmos desenvolvidos para o MPM. As características dos dois modelos, citadas acima, fornecem um arcabouço teórico para a classificação de algoritmos do MPM em mapeáveis e não mapeáveis.

5 Arquitetura de *Software* para Monitorização

A atividade de monitorização deve causar o mínimo impacto sobre a aplicação subjacente. Propomos uma arquitetura que divide um programa distribuído em dois níveis: nível da aplicação e o nível de controle (Figura 1). Um subsistema denominado *fotógrafo* se encarrega de receber estados dos componentes da aplicação e montar assincronamente estados globais consistentes. O monitor avalia predicados globais [7] sobre esses estados e reage sobre a aplicação.

A utilização de todos os estados da computação para a tarefa de monitorização pode ser proibi-

tiva. Uma outra abordagem consiste em se colocar filtros junto aos objetos ou processos e escolher estados de interesse (*checkpoints*) para envio ao fotógrafo. Tais estados podem ser, por exemplo, estados onde um predicado local se tornou válido e pode vir a influenciar um predicado global. Essa abordagem garante um máximo de autonomia aos processos ou objetos, mas tem o inconveniente de não garantir a participação desses *checkpoints* em *checkpoints* globais consistentes [12]. Caso um *checkpoint* participe de pelo menos um *checkpoint* global consistente ele é denominado *útil*; caso contrário, é denominado *inútil*.

Algoritmos *quase–síncronos* [11] garantem que os processos podem escolher *checkpoints* livremente, mas também podem vir a ser *forçados* a tirar *checkpoints* adicionais. Vários desses algoritmos garantem a ausência de *checkpoints* inúteis.

Consideramos que o monitor deve ter uma *visão progressiva* da aplicação, no sentido de que o fotógrafo sempre lhe apresenta um estado global mais recente que o anterior. Dentro do nosso conhecimento, não houve uma proposta genérica para a construção progressiva de *checkpoints* globais consistentes. A próxima Seção resume os resultados que obtivemos utilizando essa abordagem [9].

6 Construção Progressiva de *Checkpoints* Globais Consistentes

Denominamos $\hat{\sigma}_i^\iota$ um *checkpoint* relacionado a um estado ordinário σ_i^γ, $\iota \leq \gamma$, de um processo ou objeto da aplicação. A Definição 6.1 estabelece a condição necessária para uma aplicação permitir a construção progressiva de *checkpoints* globais consistentes. O espaço limitado impede a inclusão de mais detalhes sobre a construção progressiva de *checkpoints* globais consistentes, informações adicionais e a prova do Teorema 6.1 podem ser encontradas em [9].

Definição 6.1 Visão Progressiva da Aplicação *Seja um checkpoint global consistente* $\hat{\Sigma} = (\hat{\sigma}_1^{\iota_1}, \hat{\sigma}_2^{\iota_2}, \ldots, \hat{\sigma}_n^{\iota_n})$ *e seja* $S = (\hat{\sigma}_1^{\iota_1+1}, \hat{\sigma}_2^{\iota_2+1}, \ldots, \hat{\sigma}_n^{\iota_n+1})$ *o conjunto dos checkpoints que sucedem os checkpoints em* $\hat{\Sigma}$. *Um subconjunto não vazio de checkpoints em* S *pode ser substituído em* $\hat{\Sigma}$ *de maneira a formar um novo checkpoint global consistente.*

Para qualquer algoritmo que não permita *checkpoints* inúteis é possível a determinação, com custo $O(n^2)$ para tempo e espaço, de um subconjunto maximal de S a ser substituído em $\hat{\Sigma}$. Para um algoritmo específico, esse custo pode ser mais baixo. Um outro resultado interessante obtido a partir do estudo progressivo da aplicação está expresso no Teorema 6.1.

Teorema 6.1 *Uma aplicação permite visão progressiva se e somente se não permite checkpoints inúteis.*

Esse resultado garante que *qualquer* algoritmo que não admite *checkpoints* inúteis pode ser utilizado para monitorização. Além disso, acreditamos que novos algoritmos poderão ser desenvolvidos e provados utilizando esse resultado. Notamos que vários algoritmos na literatura provam que não admitem *checkpoints* inúteis através da apresentação de um *checkpoint* global consistente para cada *checkpoint* local [3, 4]. Nosso resultado permite provas mais flexíveis.

A nossa abordagem também permite a detecção de *checkpoints* inúteis, garantido que todos os *checkpoints* úteis serão utilizados pelo fotógrafo em algum corte consistente.

7 Conclusões

A obtenção de estados globais consistentes é fundamental para a solução de uma série de problemas em sistemas distribuídos e é mais eficiente quando feita de maneira assíncrona.

Apresentamos relações de precedência e consistência para aplicações construídas sobre o Modelo de Processos e Mensagens (MPM) e sobre o Modelo de Objetos e Ações (MOA) e discutimos o mapeamento de algoritmos para cortes consistentes do MPM para o MOA.

Propomos uma arquitetura para monitorização de sistema que separa a aplicação do monitor. Um processo intermediário, denominado fotógrafo se encarrega de montar estados globais consistentes e apresentá-los progressivamente ao monitor.

A análise da construção progressiva de *checkpoints* globais consistentes trouxe dois resultados interessantes: (i) qualquer algoritmo que não permite *checkpoints* inúteis pode ser utilizado para a monitorização e (ii) a visão progressiva é equivalente à ausência de *checkpoints* inúteis. Consideramos que esse último resultado pode ser valioso para a elaboração de novos algoritmos.

Estamos trabalhando no desenvolvimento de um ambiente para monitorização utilizando os algoritmos apresentados sobre o Guaraná (`http://www.dcc.unicamp.br/~oliva`), uma arquitetura de *software* reflexiva bastante flexível, projetada para facilitar a reconfiguração e o reuso de objetos do meta-nível [1, 13]. Nosso trabalho está relacionado à arquitetura de *software* para o desenvolvimento de aplicações distribuídas confiávies que está sendo elaborada pelo Laboratório de Sistemas Distribuídos do Instituto de Computação/UNICAMP [5].

Agradecimentos

Este trabalho recebe apoio financeiro da FAPESP, processo número 95/1983-8 para Islene Calciolari Garcia e número 96/1532-9 para o Laboratório de Sistemas Distribuídos do Instituto de Computação/UNICAMP.

Referências

[1] I. C. G. Alexandre Oliva and L. E. Buzato. The reflexive architecture of Guaraná. Technical Report IC-98-14, Instituto de Computação, Universidade Estadual de Campinas, Apr. 1998.

[2] O. Babaoglu. Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms. Technical Report UBLCS-93-1, University of Bologna, 1993.

[3] R. Baldoni, J. M. Helary, A. Mostefaoui, and M. Raynal. Consistent Checkpointing in Message Passing Distributed Systems. Technical Report 2564, INRIA, June 1995.

[4] R. Baldoni, F. Quaglia, and P. Fornara. An Index-Based Checkpoint Algorithm for Autonomous Distributed Systems. Technical Report 07-97, Universitá "La Sapienza", Roma, Italy, Mar. 1997.

[5] L. Buzato, H. Liesenberg, C. R. R. Anido, and M. de Toledo. Uma arquitetura de software para o desenvolvimento de aplicações distribuídas confiáveis. In *First Workshop on Distributed Systems (WoSid'96)*, Salvador, BA, Brasil, May 1996.

[6] M. Chandy and L. Lamport. Distributed Snapshots: Determining Global States of Distributed Systems. *ACM Transactions on Computing Systems*, 3(1):63–75, Feb. 1985.

[7] R. Cooper and K. Marzullo. Consistent Detection of Global Predicates. *SIGPLAN Notices*, 26(12):167, 1991.

[8] I. C. Garcia and L. E. Buzato. Asynchronous Construction of Consistent Global Snapshots in the Object and Action Model. In *Proceedings of the 4th International Conference on Configurable Distributed Systems (To appear)*, Annapolis, Maryland, EUA, May 1998. IEEE. Available as Technical Report IC–98–16.

[9] I. C. Garcia and L. E. Buzato. Progressive Constrution of Consistent Global Checkpoints. Technical report, Instituto de Computação, Universidade Estadual de Campinas, 1998. Em preparação.

[10] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM*, 21(7):558–565, July 1978.

[11] D. Manivannan and M. Singhal. Quasi–Synchronous Checkpointing: Models, Characterization, and Classification. Technical Report OH 43210, Department of Computer and Information Science, The Ohio State University, 1997.

[12] R. H. B. Netzer and J. Xu. Necessary and Sufficient Conditions for Consistent Global Snapshots. *IEEE Transactions on Parallel and Distributed Systems*, 6(2):165–169, 1995.

[13] A. Oliva and L. E. Buzato. An overview of MOLDS: A Meta-Object Library for Distributed Systems. Technical Report IC-98-15, Instituto de Computação, Universidade Estadual de Campinas, Apr. 1998.

[14] R. Schwarz and F. Mattern. Detecting Causal Relationships in Distributed Computations: In Search of the Holy Grail. Technical Report SFB 124-15/92, University of Kaiserslautern, Dec. 1992.

[15] S. K. Shrivastava and S. M. Wheater. Implementing Fault-Tolerant Distributed Applications Using Objects and Multi-Coulored Actions. In *Proceedings of Tenth International Conference on Distributed Computing Systems*, pages 203–210, Paris, France, may 1990.

[16] S. M. Wheater and D. L. McCue. Configuring Distributed Applications using Object Decomposition in an Atomic Action Environment. In J. Kramer, editor, *Proceedings of the International Workshop on Configurable Distributed Systems*, pages 33–44. IEE (UK), Imperial College of Science, Technology and Medicine, UK, March 1992. ISBN 0-85296-544-3.