Implementação e Análise do Algoritmo de Snapshot Distribuído de Chandy-Lamport

Luis Felipe Sena

[1]Universidade Federal da Bahia (UFBA)
Salvador – BA – Brazil

Abstract. *This paper presents a Python implementation and analysis of the Chandy-Lamport distributed snapshot algorithm. The implementation simulates a distributed system with three processes communicating through bidirectional channels, demonstrating the algorithm's ability to capture consistent global states. We discuss the implementation details, challenges encountered, and provide a thorough analysis of the algorithm's behavior through detailed logging and state tracking.*

Resumo. *Este artigo apresenta uma implementação em Python e análise do algoritmo de snapshot distribuído de Chandy-Lamport. A implementação simula um sistema distribuído com três processos comunicando-se através de canais bidirecionais, demonstrando a capacidade do algoritmo de capturar estados globais consistentes. Discutimos os detalhes da implementação, desafios encontrados e fornecemos uma análise completa do comportamento do algoritmo através de logs detalhados e rastreamento de estados.*

Palavras-chave: *Sistemas Distribuídos, Algoritmo de Chandy-Lamport, Snapshot Distribuído, Estados Globais Consistentes.*

1. Introdução

O algoritmo de snapshot distribuído de Chandy-Lamport [Chandy and Lamport 1985] é fundamental para a determinação de estados globais consistentes em sistemas distribuídos. Este trabalho apresenta uma implementação prática do algoritmo, demonstrando sua eficácia na captura de estados globais consistentes em um ambiente simulado com três processos. A importância deste algoritmo reside na sua capacidade de capturar estados consistentes sem interromper a execução normal do sistema distribuído, permitindo debug, checkpointing e análise de propriedades globais.

2. Fundamentação Teórica

O algoritmo de Chandy-Lamport baseia-se em dois conceitos principais:

- Captura do estado local de cada processo
- Registro de mensagens em trânsito nos canais de comunicação

Um estado global consistente deve satisfazer a propriedade fundamental de que, se uma mensagem é registrada como recebida em um estado local de destino, ela deve também ter sido registrada como enviada no estado local da origem. Esta propriedade é garantida pelo algoritmo através do uso estratégico de markers e do protocolo de gravação de mensagens.

O algoritmo utiliza mensagens especiais chamadas ”markers” para coordenar a captura do estado global, garantindo a consistência do snapshot através de três regras fundamentais:

1. Um processo registra seu estado local ao receber o primeiro marker
2. O processo então propaga markers para todos os outros processos
3. O processo começa a registrar mensagens recebidas em cada canal até receber um marker naquele canal

2.1. Propriedades de Consistência

O algoritmo garante as seguintes propriedades:

- **Consistência Causal:** Se um evento e1 causalmente precede um evento e2, e e2 está no snapshot, então e1 também está no snapshot
- **Atomicidade:** O snapshot é registrado de forma atômica, mesmo que os processos capturem seus estados em momentos diferentes
- **Não-Interferência:** O processo de snapshot não interfere na execução normal do sistema

3. Implementação

3.1. Estrutura do Sistema

Nossa implementação utiliza Python com as seguintes características principais:

- Processos implementados como threads independentes usando a biblioteca threading
- Canais de comunicação bidirecionais usando Queue do módulo queue
- Estados locais representados por números inteiros aleatórios (1-100)
- Sistema de logging detalhado usando o módulo logging do Python

3.2. Componentes Principais

3.2.1. Classe Process

A classe Process implementa a lógica principal do algoritmo:

- **Gerenciamento de Estado:**
  - Estado local (self.state)
  - Estado do snapshot (self.snapshot_state)
  - Conjunto de canais em gravação (self.recording_channels)
- **Controle de Execução:**
  - Thread control (self.running)
  - Locks para sincronização (self.lock)
  - Eventos para coordenação (self.snapshot_completed)
- **Manipulação de Mensagens:**
  - Processamento de markers
  - Gravação de mensagens em trânsito
  - Gerenciamento de canais de comunicação

3.2.2. Sistema de Comunicação

O sistema de comunicação é implementado usando:

- **Queue:** Implementa canais FIFO assíncronos
- **Mensagens:** Tuplas (tipo, conteúdo) onde tipo pode ser 'msg' ou 'marker'
- **Delays:** random.uniform(0.1, 0.3) para simular latência de rede realista
- **Controle de Concorrência:** Locks para acesso seguro a recursos compartilhados

3.3. Mecanismo de Snapshot

O processo de snapshot segue as seguintes etapas:

1. **Iniciação:**
    - Processo iniciador grava seu estado
    - Inicia gravação em todos os canais
    - Envia markers para todos os processos
2. **Propagação:**
    - Processos gravam estado ao receber primeiro marker
    - Propagam markers para outros processos
    - Iniciam gravação em canais sem marker
3. **Finalização:**
    - Processos param de gravar canal ao receber marker
    - Reportam snapshot quando todos os canais receberam markers

4. Análise dos Resultados

4.1. Comportamento do Sistema

A análise dos logs demonstra o funcionamento correto do algoritmo:

```
[INÍCIO] P0 iniciou snapshot com estado local = 61
[MARKER] P0 -> P1
[MARKER] P0 -> P2
```

O processo P0 inicia o snapshot e propaga markers:

```
[MARKER] P1 recebeu primeiro marker de P0
[SNAPSHOT] P1 gravou estado local = 47
[MARKER] P1 -> P2
```

4.2. Captura de Estados

Os estados locais capturados foram:

- P0: 61
- P1: 47
- P2: 35

4.3. Mensagens em Trânsito

O sistema capturou corretamente as mensagens em trânsito:

```
[CAPTURA] P1 gravou mensagem do canal P2: MSG from 1
[CAPTURA] P0 gravou mensagem do canal P1: MSG from 0
```

4.4. Análise de Corretude

A implementação satisfaz as propriedades fundamentais do algoritmo:

- **Terminação:** O algoritmo termina em tempo finito, com todos os processos completando seus snapshots
- **Consistência:** O estado global capturado é consistente, preservando a causalidade das mensagens
- **Não-Bloqueio:** Os processos continuam sua execução normal durante o snapshot

5. Conclusão

A implementação demonstrou com sucesso os principais aspectos do algoritmo de Chandy-Lamport:

- Captura consistente de estados locais
- Registro preciso de mensagens em trânsito
- Coordenação eficiente entre processos
- Terminação adequada do algoritmo

Os resultados obtidos validam a eficácia do algoritmo na captura de estados globais consistentes em sistemas distribuídos. A implementação em Python demonstrou ser uma escolha adequada, oferecendo as ferramentas necessárias para simular um ambiente distribuído e implementar o algoritmo de forma clara e eficiente.

Referências

Chandy, K. M. and Lamport, L. (1985). Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75.