1. Introdução
   1. Tema

Este trabalho aborda um método para a criação de Checkpoints não coordenados em sistemas de simulação computacional distribuída com o intuito de evitar a criação de Checkpoints sem valor. Desta maneira possibilitando que o sistema se recupere de falhas por violação de tempo, tendo checkpoints úteis para realizar a operação de Rollback.

* 1. Contextualização

Simulações computacionais tem o propósito de permitir a representação do comportamento de um sistema com o uso de um modelo de representação. Um modelo incorpora propriedades do sistema que representa, possuindo então a capacidade de “imitar” seu comportamento.

Tendo cada vez mais integração entre áreas de conhecimento distintas, por exemplo aplicações médicas com uso de operações matemáticas, motivou-se os estudos para o uso de modelos heterogêneos de simulação. Modelos heterogêneos permitem que os elementos da simulação diferenciem quanto sua linguagem de programação, sua interface e na maneira em que trocam mensagens [Reynolds 1988].

Com o objetivo de reduzir o tempo da simulação, o modelo da simulação pode ser dividido em elementos distintos que podem ser executados em sistemas computacionais distribuídos [FUJ 1999].

Na Simulação computacional, checkpoints são criados para garantir a consistência da simulação naquele instante de tempo e evitar o desperdício de computação caso ocorra falha por violação de tempo, assim ao invés de voltar a computação do início ela é retomada de um último checkpoint consistente.

Segundo [ELN 2002], checkpoints não coordenados permitem a cada processo a máxima autonomia ao decidir quando criar um checkpoint. A maior vantagem deste modelo é que cada processo cria checkpoint quando acha mais conveniente. Porém checkpoints não coordenados possuem desvantagens, uma delas é o efeito dominó, outra desvantagem é a criação de checkpoints inúteis e por fim o problema que pode ser necessário um coletor de lixo para apagar os checkpoints que não serão mais úteis.

Problema...

Este trabalho foi desenvolvido tomando como base a arquitetura de co-simulação heterogênea denominada *Distributed Cosimulation Backbone* [Mello, 2005], ou simplesmente DCB. O DCB foi inspirado na *High Level Architeture* (HLA). A HLA é um padrão IEEE para arquitetura de sistemas de simulação distribuídos [FUJ 2000]. O DCB é distribuído, uma vez que permite que os modelos estejam distribuídos logicamente e fisicamente, é heterogêneo, permitindo que seus elementos sejam desenvolvidos em linguagem de programação ou possuam interfaces diferentes e seus elementos assíncronos seguem a política de checkpoints quanto a recuperação pós falha.

* 1. Objetivos
     1. Geral  
        Desenvolver um mecanismo de identificação de estados seguros para a criação de checkpoints não coordenados no DCB reduzindo a probabilidade de desperdício de processamento com a geração de checkpoints inúteis.
     2. Específicos
* Estudar técnicas de identificação de estados seguros.
* Identificar estratégias de criação de Checkpoints a partir de estados seguros.
* Implementar uma técnica de criação de Checkpoints em base de estados seguros no DCB.
* Verificar a qualidade da técnica por meio de estudos de caso.
  1. Justificativa
  2. Estrutura do Trabalho  
     ...

1. Referencial Teórico
   1. Simulação  
      Basicamente todos os modelos de simulação são a especificação de um sistema físico (ou pelo menos alguns de seus componentes) transformados em estados e eventos. Realizar uma simulação computacional significa ‘imitar’ a ocorrência dos eventos reais a partir de um modelo computacional que o representa. Numa simulação contínua, as mudanças de estado ocorrem continuamente no tempo, enquanto em uma simulação discreta a ocorrência de um evento é instantânea e fixo a um ponto selecionado no tempo [Ferscha 1995]. No presente trabalho sempre que for falado em simulação, estará se referindo à simulação discreta de eventos.
   2. Simulação Distribuída

A simulação distribuída lida com a execução de módulos da simulação em ambientes computacionais distribuídos conectados a uma rede local (LAN) ou uma rede de longa distância (WAN). Em ambos os casos a execução de um único modelo de simulação, provavelmente composto de diversos Processos Lógicos (PL), é distribuída entre os múltiplos computadores [FUJ 1999].

O principal fator que leva as simulações a serem distribuídas em ambientes distintos é que ela pode reduzir o tempo de simulação drasticamente, uma vez que cada Processo Lógico possa cumprir sua função paralelamente aos outros.

A sincronização na simulação distribuída tem como objetivo geral garantir que os eventos disparados pelos Processos Lógicos ocorram em seus devidos tempos de evento de modo ordenado. Cada evento ocorre em um instante de tempo de simulação, o tempo de evento. A ordem de execução de eventos internos de um PL utiliza como referência um tempo local de simulação, ou *Local Virtual Time* LVT. Já a ordem de execução de eventos externos precisa ser controlada por um tempo global reconhecido em todos os PL’s do modelo, chamado de *Global Virtual Time* GVT.

* + 1. Simulação Síncrona

Na simulação síncrona, também chamada de simulação conservadora, elementos não podem retroceder no tempo no caso de alguma falha ou violação, fazendo com que o protocolo que implementa as regras de controle para a troca de mensagens deve garantir uma ordem segura de ocorrência de eventos durante uma simulação [~~Mello 2005~~].

Por exemplo, um PL(A) de simulação com LVT igual a 10 deseja executar um evento interno com tempo do evento igual a 12. Contudo um PL(B) está preparando uma solicitação ao PL(A) para que ele execute um evento no tempo 11. Um algoritmo de sincronização conservadora deve garantir que o PL(A) execute o evento do tempo 12 somente depois que existam garantias que nenhum outro PL irá solicitar ao PL(A) que execute um evento num tempo menor que 12.

Para garantir a ordem de ocorrência dos eventos, na simulação síncrona, o controle geralmente é mantido pelo relógio global (GVT) [FERSCHA 95]. O GVT síncrono é dado pelo maior dentre todos LVT dos PL’s. Como na simulação síncrona os PL’s solicitam a execução de eventos apenas para instante de tempo maiores que o GVT, garante-se que a ordem será garantida.

* + - 1. Lookahead  
         Uma alternativa que visa oferecer mais desempenho à simulação síncrona é a função de Lookahead. Ela permite que cada PL possa predizer quando gerará eventos no futuro, dando assim uma margem para outros processos avançarem sua computação até este intervalo de tempo. O DCB possui um mecanismo de Lookahead estático implementado, onde se estabelece um limite para este avanço.
    1. Simulação Assíncrona

Também chamada de abordagem otimista, a simulação assíncrona permite que eventos possam ser executados fora de uma ordem temporal permitindo a ocorrência de violações de tempo. Entretanto, a consistência entre a sincronização entre os PL’s é garantida através de políticas de recuperação de estados consistentes (Rollback) em situação de erro.

Tomando como exemplo a figura 2 o estado de PL1 torna-se inconsistente com o envio de uma mensagem de PL0 para PL1 com o tempo de evento igual a 10. Neste caso, o PL1 deve retroceder no tempo até o LVT 10, tempo em que ocorreu a violação de tempo local, e executar novamente os eventos até o LVT 15. As mensagens enviadas a partir de PL1 entre os tempos 10 e 15, se houverem, são denominadas mensagens órfãs. Por este motivo, a recuperação de um estado seguro pode gerar um efeito conhecido como efeito dominó [Elnozahy 2002]. Nele, quando um PL precisa executar os eventos em um período de tempo passado, as mensagens enviadas neste período (mensagens órfãs) podem provocar a necessidade de recuperação de estado seguro (Rollback) em um segundo PL, que por sua vez pode interferir em um terceiro PL, e assim por diante [FERSCHA 95].

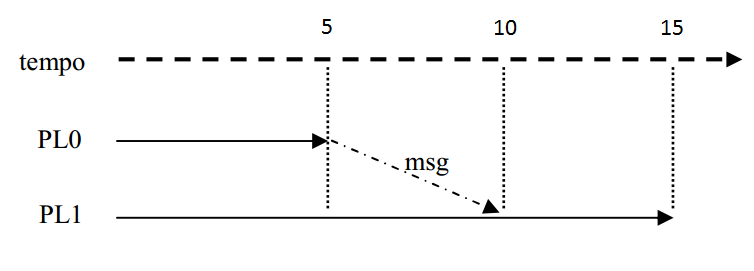


Figura 2: Exemplo de estado inconsistente.

Fonte: Desenvolvida pelo autor.

Na literatura foram encontrados dois principais métodos para realizar Rollback durante uma simulação: o método de criar logs e o método de estabelecer checkpoints.

* 1. Rollback baseado em Log
  2. Rollback baseado em Checkpoint

Quando ocorre uma falha, a política de Rollback baseado em checkpoint restaura o estado do sistema de acordo com o conjunto de checkpoints. Protocolos baseados em checkpoints são menos restritivos e mais fáceis de implementar quanto aos protocolos de Log [Elnozahy 2002].

As técnicas de Rollback podem ser classificadas em três categorias: Checkpoints coordenados, Checkpoints não-coordenados e Checkpoints induzidos a comunicação.

* + 1. Checkpoints Coordenados

Checkpoints coordenados exige que os processos orquestrem seus checkpoints a fim de formar um estado consistente global. Checkpoints coordenados simplificam a recuperação e não são suscetíveis ao efeito dominó, uma vez que todos os processos utilizarão seu último checkpoint.

* + 1. Checkpoints Não-Coordenados

Segundo [ELN 2002], checkpoints não coordenados permitem a cada processo a máxima autonomia ao decidir quando criar um checkpoint. A maior vantagem deste modelo é que cada processo cria checkpoint quando acha mais conveniente. Porém checkpoints não coordenados possuem três desvantagens:

1. É possível ocorrer o efeito dominó, que pode causar grande perda de trabalho útil, podendo retornar até o início da computação.
2. Um processo pode criar checkpoints inúteis que nunca farão parte de estados consistentes. Checkpoints inúteis são indesejáveis pois eles causam overhead e não contribuem com o processo de Rollback.
3. Checkpoints não coordenados forçam cada processo a manter vários checkpoints, e chamar periodicamente um coletor de lixo para apagar os checkpoints que não serão mais úteis.
   * 1. Checkpoints induzidos a comunicação
   1. Mensagem Nula
4. Metodologia
5. Referências

FUJIMOTO, R. M. Parallel and distributed simulation systems. [S.l.]:

Wiley-Interscience Publication, 2000. 299 p.