Uma Hieraquia para Classificação de Protocolos Otimistas de Sincronização em Simulação Distribuída

Renata S. Lobato
Dep. de Ciências de Computação e Estatística, IBILCE, UNESP
Rua Cristóvão Colombo, 2265
São José do Rio Preto, Brasil, 15054-000
renata@dcce.ibilce.unesp.br

and

Marcos J. Santana e Regina H. C. Santana
Dep. de Ciências de Computação e Estatística, ICMC, USP
Av. do Trabalhador São-Carlense, 400 Caixa Postal 668
São Carlos, Brasil, 13560-970
{mjs, rcs}@icmc.usp.br

and

Roberta S. Ulson
Dep. de Computação, Faculdade de Ciências, UNESP
Av. Engenheiro Luis Edmundo Carrijo Coube, s/n, Caixa Postal 473
Bauru, Brasil, 17033-360
roberta@fc.unesp.br

Abstract

This paper presents an updated classification for optimistic distributed synchronization simulation protocols. As the distributed simulation performance depends on two main factors, events synchronization costs (the protocol that is used) and process communication costs, the taxonomy can help researchers and distributed simulation users to study these aspects, by grouping the protocols with similar characteristics.

Keywords: Distributed Simulation, Synchronization Protocols

Resumo

Este artigo apresenta uma versão estendida da classificação para protocolos de simulação distribuída otimista. Como o desempenho da simulação distribuída depende de dois fatores principais, o custo da sincronização de eventos (o protocolo utilizado) e o custo da comunicação entre processos, a classificação pode auxiliar pesquisadores no estudo desses aspectos, estando agrupados os protocolos com características semelhantes

Palavras chaves: Simulação Distribuída, Protocolos de Sincronização

1 INTRODUCÃO

O protocolo conservativo *CMB* (Chandy, Misra e Bryant) [14] e o protocolo otimista *Time Warp* [10] são dois protocolos de sincronização em simulação distribuída discutidos amplamente na literatura nos últimos 20 anos. Não existe um consenso sobre qual dos dois apresenta melhor desempenho e por esse motivo tem-se optado por desenvolver protocolos que buscam explorar as vantagens que os protocolos *CMB* e *Time Warp* apresentam. Assim, podem-se inserir características conservativas em um protocolo puramente otimista, ou então tornar mais otimista um protocolo conservativo. A primeira opção, por exemplo, deu origem a muitas variantes do protocolo otimista *Time Warp*.

Um novo problema surgiu para aqueles que procuravam uma alternativa entre *CMB* e *Time Warp*, envolvendo a tomada de decisão sobre qual dos protocolos otimistas utilizar para a resolução de problemas.

Uma avaliação entre esses protocolos fica facilitada quando se consegue organizá-los, agrupando os protocolos que possuem características semelhantes. Com uma classificação desse tipo, pode-se avaliar um grupo ao invés de um protocolo específico.

As classificações apresentadas na literatura são redundantes e confusas, o que levou à proposta de uma nova taxonomia para facilitar o estudo dos protocolos otimistas que inserem noções de conservadorismo no *Time Warp* [12]. Essa classificação é expandida neste artigo.

Este artigo é organizado da seguinte forma: a seção 2 faz uma revisão sobre classificações de protocolos otimistas, a seção 3 estende as classificações já publicadas pelo autor [25] [12], e a seção 4 mostra as conclusões do trabalho.

2 REVISÃO E CRÍTICA DE CLASSIFICAÇÕES EXISTENTES

As pesquisas na área de simulação distribuída têm convergido para variações híbridas de protocolos de sincronização que implementam uma abordagem intermediária entre os extremos conservativo e otimista, uma vez que a escolha por um determinado protocolo é uma tarefa bastante complexa. Esses protocolos utilizam uma forma de otimismo controlado, aproveitando os benefícios da visão otimista (explorar mais paralelismo) sem serem afetados pelos seus problemas (sobrecarga causada por *rollbacks* e consumo de memória), ou adicionam mais otimismo em alguma abordagem conservativa.

Na literatura de simulação distribuída estão descritos vários protocolos para sincronização que são basicamente variantes dos protocolos *CMB* e *Time Warp*. A quantidade de protocolos leva à necessidade de uma classificação que permita estudá-los de uma forma mais organizada. A construção de uma classificação concisa e hierárquica pode facilitar as análises comparativas de desempenho entre os protocolos. O uso da classificação pode identificar os aspectos mais importantes do protocolo *Time Warp* e que merecem atenção em um estudo de desempenho através da técnica de modelagem.

2.1 Classificação de Das

A classificação apresentada por Das [5] divide os protocolos em Híbridos ou Adaptativos. Os protocolos Híbridos abrangem os protocolos que adicionam otimismo a protocolos conservativos, como por exemplo Simulação Especulativa e SRADS [5], *Breathing Time Buckets* [29], *Filtered Rollback* [5], e os protocolos que buscam controlar o otimismo no *Time Warp*, como por exemplo *Moving Time Windows* [28], *Bounded Time Warp* [5], MIMDIX [20], *Breathing Time Warp* [30], *Composite ELSA* [5], *Local Time Warp* [20] e *Filter* [16].

Os protocolos Adaptativos ajustam-se entre a abordagem otimista e a abordagem conservativa, com o objetivo de minimizar o tempo de execução da simulação distribuída. São subdivididos em Gerais, Estado Local e Estado Global. Os protocolos Gerais têm como exemplo *Moving Time Windows* e MIMDIX. Os protocolos com base em informações referentes ao Estado Local do processo lógico englobam *Penalty-based Throttling* [22], *Adaptive Time Warp* [7], Protocolo Probabilístico baseado em Custos [5], *Probabilistic Distributed Discrete Event Simulation Protocol* [8], *Local Adaptive Protocol* [5], *Probabilistic Direct Optimism Control* (PADOC) [7] e *Adaptive Bounded Time Window* [5]. Os protocolos que utilizam informações referentes ao Estado Global da simulação distribuída agrupam *Adaptive Memory Management Protocol*, *Near Perfect State Information* (NPSI) e *Cancelback Protocol* [7].

2.2 Classificação de Srinivasan

A classificação e a classe de protocolos propostos por Srinivasan foram publicadas em trabalhos distintos e abordam de forma independente os protocolos que limitam o otimismo no Time Warp [27] e os protocolos que se adaptam às mudanças no estado da simulação [26]. Os protocolos são divididos em: Janelas, Espaço, Penalidade, Conhecimento, Probabilísticos e Estado Global.

Os protocolos com base em Janelas permitem que somente os eventos com valor de marca de tempo dentro do intervalo da janela de tempo sejam executados. São exemplos *Moving Time Windows*, *Window based Throttling* [22], *Bounded Time Warp* e *Breathing Time Warp*.

Os protocolos baseados no Espaço utilizam limites espaciais ao invés de limites temporais para limitar o otimismo. Os processos lógicos são divididos em grupos, e *Time Warp* é utilizado para efetuar o sincronismo dentro de cada grupo. Os grupos interagem de forma conservativa. Pode-se citar como exemplo *Local Time Warp* e *SRADS*.

De acordo com o autor, nos protocolos baseados em Penalidade utiliza-se o comportamento da simulação para penalizar (bloquear) processos lógicos enquanto outros são favorecidos (podem continuar a execução). Como exemplo tem-se *Penalty-based Throttling*.

Os protocolos com base em Conhecimento utilizam informação sobre a ocorrência de *rollbacks* para restringir a propagação de computação provavelmente incorreta. Exemplos incluem *Filter* e WOLF [20].

Os protocolos Probabilísticos fazem uma previsão probabilística sobre o comportamento dos processos lógicos. Como exemplo tem-se MIMDIX.

Os protocolos com base no Estado Global da simulação analisam a simulação distribuída como um todo para limitar o excesso de otimismo do *Time Warp*. Exemplos são *Adaptive Memory Management* e os protocolos NPSI.

Srinivasan [26] também propõe que o controle do otimismo no *Time Warp* seja classificado, devido ao fato de que alguns protocolos introduzem atrasos entre as execuções dos eventos. Esses protocolos foram denominados *Asynchronous Adaptive Waiting Protocols* (AAWP). Como exemplos pode-se citar *Penalty-based Throttling*, *Adaptive Time Warp*, *Local Adaptive Protocol*, protocolos NPSI, *Breathing Time Warp* e UDS.

3 CLASSIFICAÇÃO PARA PROTOCOLOS OTIMISTAS EM SIMULAÇÃO DISTRIBUÍDA

Este trabalho sugere uma extensão para as classificações de Das e Srinivasan, buscando agrupar de forma natural a grande variedade de protocolos propostos ao longo dos anos [25] [12]. Essa classificação permite uma melhor visualização dos protocolos otimistas, consistindo em uma abordagem genérica e hierárquica.

Além disso, nenhuma das classificações apresentadas considera o fato de que existem diferentes algoritmos que podem ser seguidos para efetuar determinadas tarefas, como por exemplo o salvamento de estados e o cancelamento de mensagens. Esses e outros algoritmos, como mostrado na literatura, podem influenciar no desempenho do protocolo e, por isso, serão considerados na classificação apresentada nesta seção.

A classificação de Das apresenta-se inconsistente ao separar os Protocolos que Controlam Otimismo dos Protocolos Adaptativos. Ambos procuram limitar o otimismo do *Time Warp*, podendo executar este trabalho de forma estática (não adaptativa) ou dinâmica (adaptativa), que Das não menciona. Além disso, a noção de protocolo Híbrido abrange também aqueles que adicionam otimismo a protocolos conservativos, ou seja, protocolos cujas características principais são as da abordagem *CMB*. Esses protocolos, por apresentarem características intrínsecas de abordagens conservativas, como por exemplo o uso de mensagens nulas, não são considerados [12].

Das também classifica *Local Adaptive Protocol* [5] como um protocolo que se utiliza do estado da simulação para limitar o otimismo. Apesar de cumprir esse papel, esse protocolo apresenta *deadlocks* e mensagens nulas, como no *CMB*, e também não será considerado na taxonomia proposta. A classificação de Srinivasan cria grupos de protocolos com base em Penalidade e Estado Global e não considera protocolos que fazem uso de informações referentes ao estado local do processo lógico. A característica básica de todos esses protocolos de sincronização é a utilização de informações relativas ao estado da simulação para definir se os processos lógicos podem ou não executar eventos, ou seja, têm como base o Estado da simulação. O grupo dos protocolos com base no Conhecimento também utiliza informação relativa ao estado da simulação distribuída, porém essa informação é difundida para outros processos lógicos. Nesse caso, a denominação mais adequada é Difusão. Srinivasan também não separa os protocolos adaptativos dos não adaptativos.

Além disso, ambas as classificações tratam o protocolo *Adaptive Memory Management* como sendo um protocolo baseado no estado global da simulação. É correto afirmar que esse protocolo, e também o protocolo *Cancelback*, limitam o otimismo do *Time Warp* observando o estado da simulação, porém essa limitação é feita de maneira indireta. O objetivo principal consiste em efetuar o gerenciamento da memória disponível para salvar estados, mas ao imporem aos processos lógicos um espaço limitado na memória disponível estão também prevenindo que os mesmos avancem muito no tempo de simulação. Isso justifica a criação de uma nova classe na taxonomia, como explicado nas seções seguintes, como apresentado em [12].

A classificação para os protocolos otimistas foi desenvolvida em uma estrutura concisa que fornece uma visão global dos protocolos. Essa classificação é organizada da seguinte forma: de acordo com as Características de Implementação do protocolo de sincronização (figura 1), de acordo com as Características do Protocolo de Sincronização propriamente dito (figura 2) [25] e de acordo com o Modo de Limitar o Otimismo (figura 3).

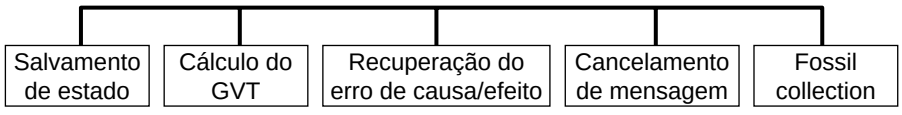

Figura 1: Organização de Protocolos segundo as Características de Implementação

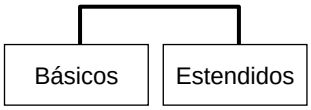

Figura 2: Organização de Protocolos segundo as Características do Protocolo de Sincronização

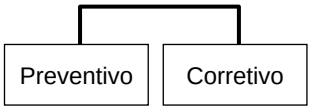

Figura 3: Organização de Protocolos segundo o Modo de Limitar o Otimismo

Essa classificação considera apenas os protocolos que têm como base a limitação do otimismo no *Time Warp*, o que exclui protocolos como, por exemplo, *Local Adaptive Protocol*, que está incluído nas classificações de Das e Srinavasan, e os protocolos que adicionam otimismo a protocolos conservativos classificados por Das.

A organização que considera as características de Implementação é discutida na seção 3.1, estando justificada pelo fato de que não existe um consenso na área de simulação distribuída com relação à melhor maneira de se implementar protocolos baseados no Time Warp [1].

A organização com base no Protocolo de Sincronização apresenta os protocolos divididos em duas categorias, Protocolo Básico e Protocolos Estendidos. O primeiro corresponde ao mecanismo proposto por Jefferson [10] enquanto os Protocolos Estendidos correspondem a variantes do *Time Warp* que procuram reduzir o excesso de otimismo da idéia original de Jefferson. A seção 3.2 apresenta essa classificação.

A organização com base no Modo de Limitar o Otimismo justifica-se pelo fato de que alguns protocolos limitam o otimismo sem fazer análise do estado da simulação ou do processo lógico (por exemplo, com respeito ao número de *rollbacks* e eficiência), mas adaptam-se durante a execução. A seção 3.3 discute essa abordagem.

3.1 Organização Segundo as Características de Implementação

A proposta original do *Time Warp* define a noção de tempo virtual e fornece as diretrizes de como efetuar a sincronização entre os processos lógicos usando a abordagem otimista com *rollbacks* e salvamento de estados. Porém, no trabalho de Jefferson não estão especificadas, para citar alguns exemplos, a forma pela qual as antimensagens devem ser enviadas, a maneira como o *rollback* deve ser efetuado ou ainda como e quando salvar as informações relativas aos estados dos processos lógicos e como efetuar o cálculo do GVT (*Global Virtual Time*). Isso implica que os projetistas e pesquisadores contam com uma gama variada de opções, além de poderem criar suas próprias soluções no desenvolvimento de seus ambientes de simulação que fornecem suporte ao *Time Warp*. As seções a seguir descrevem cada uma das classes apresentadas na classificação da figura 1.

3.1.1 Organização com Base no Salvamento de Estados

O *Time Warp* é um protocolo de sincronização otimista que permite que a simulação distribuída execute até que ocorra um erro de causa e efeito. Quando isso ocorre, *Time Warp* executa um *rollback* para retornar a simulação para um estado consistente. A execução do *rollback*, juntamente com o retorno da simulação a um estado consistente, exige alguns procedimentos que podem afetar o desempenho da simulação.

O procedimento de salvamento de estados no *Time Warp* pode ser executado por uma das seguintes abordagens (figura 4) [12]:

Copy state saving: todo o estado do processo é salvo após a execução de cada evento;

Sparse state saving: todo o estado do processo lógico é salvo periodicamente e qualquer estado pode ser recuperado através da restauração do último estado salvo antes do evento afetado pelo *rollback* e pela re-execução de eventos intermediários. Pode ser fixo (o intervalo de tempo entre os salvamentos de estado é constante durante toda a execução da simulação) ou adaptativo (o intervalo de tempo entre os salvamentos de estado é escolhido dinamicamente). Alguns métodos descritos na literatura são o Método de Rönngren Baseado no Tempo de Execução, o Método de Rönngren Baseado no Consumo de Memória, o Método de Fleischmann-Wilsey e o Método para de Chung-Xu [4] [17] [23] [24];

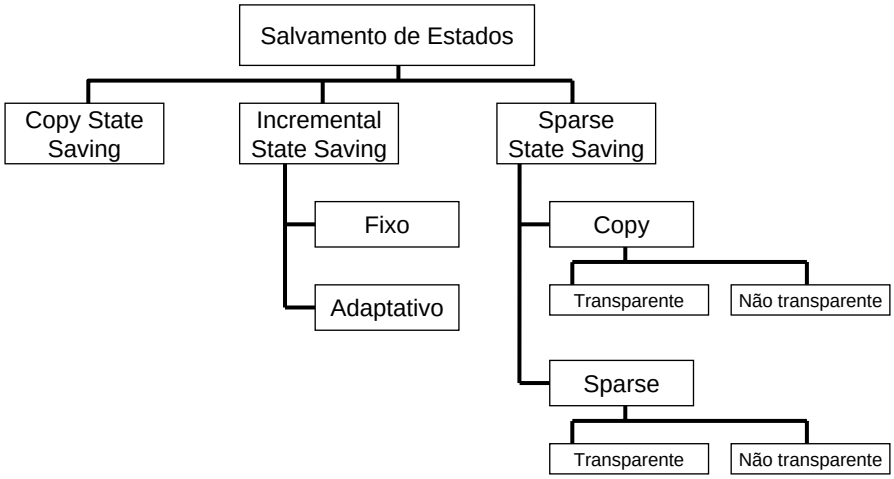

Figura 4: Organização por Salvamento de Estados

Incremental state saving: somente as partes do estado do processo lógico que foram modificadas são salvas. Pode ser dividido em transparente ou não transparente. No caso não-transparente, todas as variáveis de estado que foram modificadas devem ser identificadas pelo usuário da simulação e atualizadas. No caso transparente, as variáveis de estado que sofreram modificação são atualizadas automaticamente pelo sistema de simulação. A implementação pode optar pelo caso *Copy* e salvar as partes modificadas do estado a cada alteração, ou determinar intervalos de tempo para efetuar o salvamento, identificando-se ao caso *Sparse*. Compose (*Conservative Optimistic and Mixed Parallel-oriented Simulation Environment*) é um exemplo de utilização de *Incremental state saving* [13]

3.1.2 Organização com Base no Cálculo do GVT

O grande problema associado à tarefa de salvar estados consiste na utilização do espaço de memória disponível. Dessa forma, outro aspecto importante na implementação de um sistema *Time Warp* reside na adoção de alguma técnica para descartar os estados que foram salvos mas que não serão mais necessários. O valor do GVT garante que os estados salvos com marcas de tempo menores podem ser descartados, pois não ocorre *rollback* para estados anteriores ao GVT.

Outra razão importante para efetuar o cálculo do GVT em uma simulação distribuída consiste em oferecer suporte para a simulação distribuída interativa (DIS - *Distributed Interactive Simulation*). É permitido que apenas os eventos seguros (suas marcas de tempo são menores do que o GVT e, portanto, não irão sofrer *rollback*) podem liberar informação para o mundo externo.

O fator mais importante e que torna o cálculo do GTV uma tarefa difícil é o fato de existirem mensagens/antimensagens circulando pela rede de comunicação. Essas mensagens/antimensagens devem ser consideradas no momento de se determinar qual a menor marca de tempo da simulação distribuída.

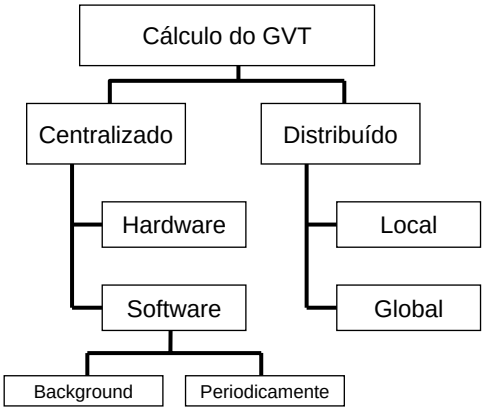

Figura 5: Organização com Base no Cálculo do *GVT*

Existem diferentes abordagens para o cálculo do valor do GVT (figura 5). Pode-se dividir essas abordagens em centralizadas ou distribuídas. Na abordagem centralizada o valor do GVT é calculado por um processo lógico (*software*) ou *hardware* específico [6]. A vantagem de se ter um *hardware* específico para efetuar tal tarefa é que existe pouca influência no desempenho da simulação, porém tem-se um aumento no custo e na complexidade.

A abordagem de um processo lógico específico (*software*) é a mais utilizada na literatura, e nesse caso tem-se o problema de minimizar o impacto do algoritmo de cálculo no desempenho da simulação distribuída. Esse algoritmo pode ser executado em *background* (concorrendo assim com os demais processos lógicos) ou pode ser iniciado periodicamente (por exemplo, quando um processo lógico necessita efetuar *fossil collection* para poder continuar a execução dos eventos). Nesse caso os processos lógicos devem esperar o novo valor do GVT [30].

Uma vantagem da abordagem distribuída para o cálculo do GVT é a ausência de sobrecarga em um único processo lógico, pois o trabalho é distribuído por todos. Essa abordagem também pode ser aplicada de forma local, onde os processos lógicos vizinhos calculam seu GVT que depois é comparado aos demais GVTs das outras vizinhanças e o menor é eleito o GVT, ou então de forma global, onde um único valor de GVT é calculado considerando-se todos os processos lógicos. *Speed GVT* [30] é um exemplo de abordagem distribuída global.

3.1.3 Organização com Base na Recuperação de Erro de Causa e Efeito

Quando ocorre um erro de causa e efeito na simulação distribuída, o protocolo de sincronização precisa corrigí-lo e reiniciar a simulação a partir de um ponto seguro no tempo. Existem duas formas de se corrigir um erro de causa e efeito (figura 6). Na proposta de Jefferson [10] os valores dos estados pelos quais o processo lógico passa são salvos e a simulação pode ser restaurada para um estado seguro através do *rollback*, que copia de volta os valores do estado salvo para as variáveis da simulação. GTW [9] e *Warped* [19] são exemplos de sistemas de simulação distribuída *Time Warp* que utilizam o *rollback* e salvamento de estados para recuperar erros de causa e efeito.

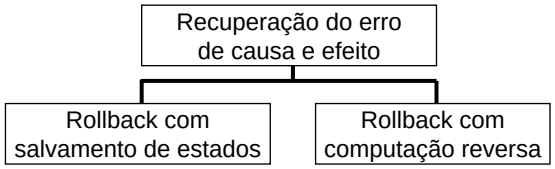

Figura 6: Organização com Base na Recuperação de Erro de Causa e Efeito

Outra forma de se recuperar um erro de causa e efeito consiste em implementar Computação Reversa, proposta por Carothers [3] [2]. Como exemplo tem-se o sistema ROSS (*Rensselaer's Optimistic Simulation System*) [1]. Nessa técnica, o *rollback* é realizado pela execução das operações inversas relativas às operações individuais que são efetuadas durante a execução de um evento. O sistema garante que as operações inversas restauram o estado da simulação para os valores que antecedem a execução do evento. A vantagem dessa técnica reside na pequena quantidade de informação de controle que precisa ser salva, em relação ao tamanho de todo o estado [2].

3.1.4 Organização com Base no Cancelamento de Mensagens

Outro fator importante que deve ser levado em consideração nas implementações baseadas no *Time Warp* relaciona-se às mensagens que são enviadas de forma errada quando ocorre um erro de causa e efeito. Essas mensagens devem ser canceladas quando um *rollback* é executado, através do envio explícito de antimensagens (Cancelamento Indireto) ou do cancelamento das mensagens enviadas diretamente nas filas — dos outros processos lógicos, em caso de memória compartilhada, ou na fila de saída do próprio processo, no caso de memória distribuída (Cancelamento Direto) (figura 7). Em arquiteturas de memória compartilhada, quando a execução de um evento resulta no envio de um novo evento para um processo lógico remoto, um apontador para este novo evento é mantido juntamente com o evento executado, na estrutura de dados do processo emissor. Isso elimina a necessidade do envio explícito de antimensagens [2]. Quando se está utilizando uma arquitetura com memória distribuída, o cancelamento ocorre no buffer de envio (NIC - *Network Interface Card*), como no *Early Cancellation* [15].

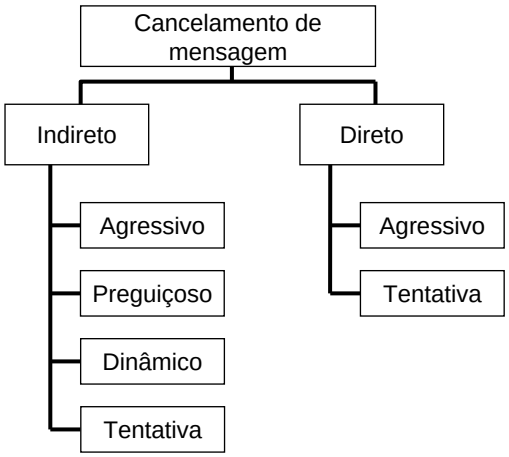

Figura 7: Organização com Base no Cancelamento de Mensagens

No Cancelamento Indireto (com uso de antimensages) as abordagens utilizadas são Cancelamento Agressivo, Cancelamento Preguiçoso, Cancelamento Dinâmico e Cancelamento por Tentativa [11]. No Cancelamento Direto (sem uso de antimensagens, pode-se utilizar as abordagens Agressivo e Tentativo.

Quando Cancelamento Agressivo é adotado e um processo lógico sofre um *rollback* para o tempo t, são enviadas antimensagens imediatamente para anular todas as mensagens enviadas com marca de tempo maior do que t. No Cancelamento Preguiçoso o processo lógico espera para ver se a reexecução dos eventos executados fora de ordem produz as mesmas mensagens. Em caso afirmativo não é necessário enviar antimensagens e no caso contrário as antimensagens são enviadas.

O Cancelamento Dinâmico permite que cada processo lógico decida qual estratégia de Cancelamento, Agressivo ou Preguiçoso, vai utilizar.

No Cancelamento em Tentativa, utilizado no *Tentative Time Warp*, um grupo de mensagens enviadas para um processo lógico pode ser cancelado por uma única antimensagem.

3.1.5 Organização com Base em Fossil Collection

As versões tradicionais do *Time Warp* implementam *fossil collection* através da comparação da marca de tempo da informação que foi salva com o valor do GVT. Tudo o que for anterior a esse tempo pode ser descartado e a memória utilizada pode ser liberada, uma vez que os *rollbacks* só irão ocorrer no máximo até o valor do relógio global. Isso implica que o cálculo do GVT deve ser efetuado regularmente (ou sempre que um processo lógico necessita espaço em memória) para que os processos lógicos consigam liberar espaço de memória e a simulação possa prosseguir. Essa abordagem pode ser considerada conservativa uma vez que os processos lógicos somente liberam a memória ocupada depois de ter certeza de que os estados não serão mais necessários para garantir a correta execução dos eventos da simulação e isso só acontece quando o valor de GVT é calculado [12]. Um processo lógico que utiliza mais espaço em memória do que os demais e precisa liberá-lo pode afetar o desempenho da simulação com sucessivos pedidos para que o cálculo do GVT seja efetuado. A figura 8 mostra a classificação dos protocolos segundo a forma de efetuar *fossil collection*.

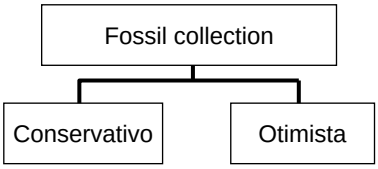

Figura 8: Organização com Base em Fossil Collection

A abordagem OFC (*Optimistic Fossil Collection*) [32] permite que os processos lógicos efetuem a tarefa de liberar espaço de memória de maneira independente, sem trocar informação com os demais processos. Cada um deles faz uma previsão do valor do GVT, com isso pode-se determinar uma estimativa do número x de estados de forma que a probabilidade de que um *rollback* atinja um número de estados maior do que x seja um fator de risco. Esse fator de risco é utilizado para determinar a agressividade da tarefa de efetuar *fossil*

collection. Dessa forma, a liberação de espaço em memória é feita de forma otimista, pois cada processo lógico considera que os estados mais antigos não serão mais necessários e, ao contrário da abordagem conservativa, não fica esperando o cálculo do GVT.

3.2 Organização Segundo as Características do Protocolo de Sincronização

Essa classificação agrupa os protocolos conforme apresentado na figura 2. O protocolo Básico consiste no *Time Warp* proposto por Jefferson [10] sem nenhuma modificação ou otimização quanto ao otimismo. Os protocolos Estendidos consistem em variantes do básico *Time Warp* que procuram reduzir o número de erros de causa e efeito e assim melhorar o desempenho da simulação distribuída.

Os protocolos Estendidos procuram aproveitar as vantagens do *Time Warp* básico como, por exemplo, explorar o paralelismo inerente à aplicação e também eliminar ou ao menos amenizar suas desvantagens (como o excesso de otimismo que pode levar a computações errôneas). A idéia básica consiste em limitar esse excesso de confiança de que a computação de um evento está sendo efetuada corretamente no tempo e assim diminuir a quantidade de erros de causa e efeito e até mesmo a necessidade por espaço livre em memória para armazenar os estados salvos. Assim como o *Time Warp* básico, todos os protocolos que estendem a funcionalidade do *Time Warp* fazem uso das Características de Implementação descritas anteriormente.

Os protocolos Estendidos podem ser divididos em três grandes grupos, que envolvem aqueles que introduzem técnicas para limitar o otimismo na sincronização entre os processos lógicos, aqueles que se preocupam em otimizar o consumo do espaço de memória disponível para que os processos lógicos efetuem salvamento de estados (limitando assim indiretamente o otimismo), e aqueles que se preocupam com o escalonamento de processos lógicos que executam em um mesmo elemento de processamento (o processo lógico com menor chance de sofrer *rollback* pode executar) (figura 9). Essas duas últimas visões dos protocolos podem ser utilizadas em conjunto com a limitação do otimismo [12].

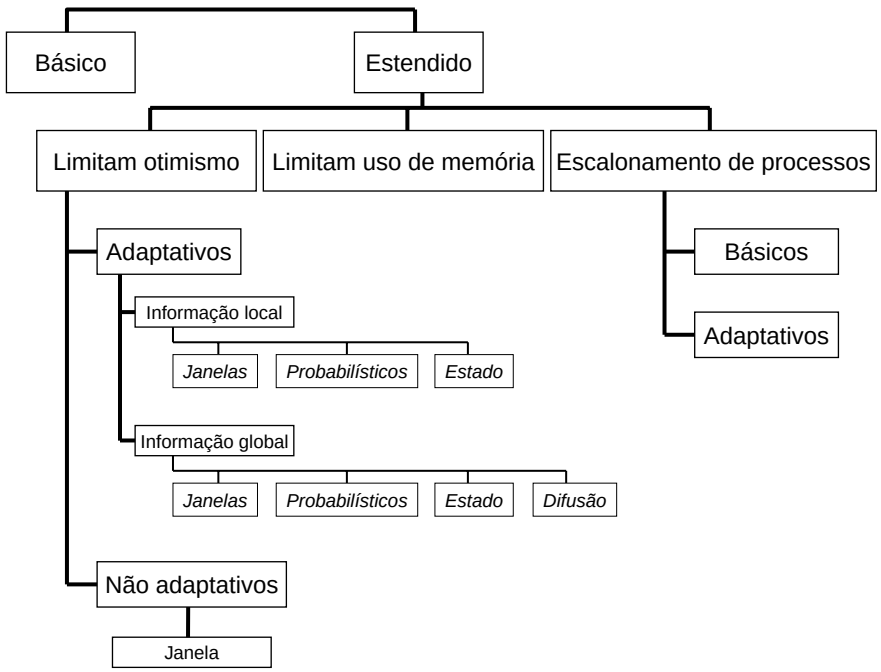

Figura 9: Organização segundo as Características do Protocolo de Sincronização

3.2.1 Protocolos que Limitam o Otimismo

Esses protocolos podem ser classificados de acordo com a maneira pela qual limitam o otimismo e pela forma com que a técnica utilizada para limitação se comporta no decorrer da simulação distribuída (isto é, se existe adaptação em relação a mudanças no comportamento da simulação ou não) resultando em protocolos Adaptativos ou Não Adaptativos [5].

A maioria dos protocolos existentes pode ser classificada como adaptativa, isto é, utilizam a situação corrente da simulação distribuída para decidir quando o otimismo deve ser controlado. Os vários protocolos Adaptativos diferem no conjunto de informações que são utilizadas para avaliar o comportamento da simulação até o momento, isto é, quais dados e como esses dados são utilizados. Dessa forma, os protocolos

podem ser divididos em Locais (a tomada de decisão leva em consideração somente as informações locais ao processo lógico) ou Globais (a decisão em um processo lógico considera informações relativas a outros processos lógicos).

Os protocolos Não Adaptativos limitam o otimismo de forma estática, isto é, não existe alteração durante a execução da simulação distribuída. Alguns protocolos baseados em janelas se encaixam nesta categoria.

Dependendo da implementação, diferentes protocolos podem ser classificados como Locais ou Globais ou mesmo como sendo Não Adaptativos, como mostrado na figura 9 e discutido a seguir.

Protocolos Baseados em Janelas de Tempo: permitem que os processos lógicos executem somente aqueles eventos cujas marcas de tempo são inferiores a um determinado valor limite (limite da janela de tempo). Exemplos de implementações não adaptativas incluem *Bounded Time Warp*, *Moving Time Windows* [7] e *Window-based Throttling* [21]. Exemplos de protocolos onde os limites das janelas se adaptam dinamicamente às condições da simulação incluem *Adaptive Time Warp* [7], *Breathing Time Warp* [29], *Adaptive Bounded Time Window* [5], UDS [20] e *Adaptive Throttle Scheme* [31]. Os protocolos baseados em janelas diferem com relação ao método utilizado para determinar o tamanho da janela e se os processos lógicos compartilham a mesma janela (informação global) ou se existe uma janela particular para cada um (informação local). Como exemplo de protocolo com janelas locais tem-se *Breathing Time Warp* e protocolos com janelas globais têm-se *Window-based Throttling* e *Bounded Time Warp*;

Protocolos Probabilísticos: esses protocolos se baseiam em análises probabilísticas para prever o comportamento futuro dos processos lógicos e assim tomar decisões sobre a sincronização dos mesmos. Como exemplo pode-se citar MIMDIX, Protocolo Probabilístico baseado em Custos, Protocolo *Probabilistic Distributed Discrete Event Simulation* e o Protocolo PADOC;

Protocolos Baseados no Estado: dependendo das condições da simulação os processos lógicos são impedidos de executar eventos e assim não avançam seus LVTs (*Local Virtual Time*). Como exemplo tem-se *Penalty based Throttling*, *Length-based Blocking* e *Composite Elsa*, nos quais as informações de estado são utilizadas localmente por cada processo lógico, e *State Query Time Warp* e os Protocolos Adaptativos NPSI [27], nos quais as decisões baseiam-se no estado global da simulação. É importante ressaltar que o protocolo *Penalty based Throttling* é definido por Das como baseado em Estado local, e por Snirivasan como baseado em Penalidade. Nesta classificação, entende-se que a penalidade sofrida por um processo lógico está associada a informações relativas ao seu estado e, portanto, justifica-se a inclusão do protocolo como baseado em estado.

Protocolos Baseados em Difusão: têm como princípio básico a difusão de informação entre os processos lógicos, com o objetivo de parar computações errôneas. Pode-se considerar que tais protocolos são de abordagem global, porque a informação difundida pode afetar todos os processos da simulação, e adaptativos, porque seu comportamento pode mudar conforme a simulação apresenta um número maior ou menor de erros de causa e efeito. Como exemplo tem-se os protocolos *Wolf*, *Filter* e *Sfilter* [16]. Os protocolos *Wolf* e *Filter*, classificados por Snirivasan como baseados em conhecimento, são aqui classificados como baseados em difusão. Das os classifica apenas com limitantes do otimismo no *Time Warp*, sem aprofundar a maneira pela qual esta limitação acontece.

3.2.2 Protocolos que Controlam o Uso do Espaço de Memória

Essa classificação inclui as técnicas de limitar o uso do espaço de armazenamento disponível, isto é, o progresso dos processos lógicos no tempo de simulação é controlado de acordo com a disponibilidade de espaço em memória (para salvar estados). Esses protocolos podem ser utilizados em conjunto com os protocolos que limitam otimismo.

Ressalta-se que Das classifica os protocolos de gerenciamento de memória como protocolos adaptativos baseados no estado global da simulação. Nesta classificação, preferiu-se agrupar os protocolos de controle da memória disponível aos processos lógicos em um grupo à parte. Todos eles são adaptativos e tomam decisões com base no estado da simulação (falta de espaço para armazenar novos estados), porém são os únicos que se preocupam com a memória finita da plataforma.

A aplicação desses protocolos é atrativa devido a vários motivos. Primeiro, porque a utilização de espaço de armazenamento é um ponto crítico nos protocolos otimistas. Segundo, porque eles efetuam um controle indireto sobre o progresso da simulação, evitando *rollbacks*. Exemplos desses mecanismos são *Adaptive Memory Management Protocol* e *Cancelback Protocol* [7].

3.2.3 Protocolos que Utilizam Escalonamento de Processos Lógicos

A principal forma de se evitar que os processos lógicos no *Time Warp* executem eventos de forma totalmente otimista é através do controle direto desse otimismo. O tratamento indireto desse problema também tem sido estudado na literatura, como é o caso do escalonamento de processos lógicos.

Em simulações distribuídas que envolvem um grande número de processos lógicos, é provável a atribuição de mais de um processo lógico para um mesmo elemento de processamento. Assim, pode-se utilizar um algoritmo de escalonamento de processos lógicos de forma que aqueles com menores chances de executarem eventos fora de ordem tenham prioridade.

Esses algoritmos de escalonamento podem ser divididos em Básicos ou Estendidos. Os básicos caracterizam-se pelo fato de escalonarem sempre o processo lógico com menor valor de LVT (*Lowest-Local-Virtual-Time-First*) ou aquele com menor valor de marca de tempo do próximo evento a ser executado (*Lowest-Timestamp-First*).

Os algoritmos estendidos procuram avaliar qual processo lógico tem menor chance de sofrer *rollback* ao executar o próximo evento. Esse processo lógico é escolhido para executar. Exemplos incluem *Adaptive Control based Scheduling*, *Service Oriented Scheduling*, *Probabilistic Scheduling*, *State based Scheduling*, *Grain Sensitive Scheduling* e *Aggressiveness/Risk Effects based Scheduling* [18].

3.3 Organização Segundo o Modo de Limitar o Otimismo

Os processos lógicos podem moldar o seu comportamento em virtude das características da simulação, como por exemplo o aumento do número de *rollbacks*. Pode-se classificar os protocolos limitantes do *Time Warp* em Preventivos e Corretivos. Os primeiros se propõem a prevenir a ocorrência de rollbacks, como por exemplo *Length-based Blocking* e os protocolos baseados em Janelas Não Adaptativos. Os protocolos Corretivos, como o próprio nome indica, corrigem o comportamento do processo lógico em função do aumento do número de erros de causa e efeito. Como exemplo têm-se *Penalty-based Throttling* e os protocolos Adaptativos.

4 CONCLUSÃO

Existem duas classificações na literatura que buscam agrupar os protocolos de sincronização (apresentadas na seção 2). Porém, essas classificações não levam em consideração as características da implementação, que podem influenciar em um estudo de desempenho entre protocolos de sincronização. Além disso, uma delas classifica protocolos que inserem otimismo em protocolos conservativos, e ambas tratam os protocolos de gerenciamento de memória juntamente com outros protocolos, apesar das características únicas que apresentam. Agrupadas e avaliadas as formas de se limitar o otimismo, a classificação apresentada em [25] [12] foi detalhada.

Os principais protocolos de sincronização descritos na literatura e que têm como objetivo principal limitar o excesso de otimismo no *Time Warp* foram divididos em três grandes grupos, que se preocupam com as Características de Implementação, Mecanismo de Sincronização e o Modo de Limitar o Otimismo. Cada um desses grupos apresenta uma hierarquia de subgrupos.

Agradecimentos

Os autores agradecem à FUNDUNESP, pelo apoio financeiro, e ao Programa de Mestrado em Ciência da Computação do Departamento de Computação e Estatística da Universidade Federal de Mato Grosso do Sul, do qual a Profa. Renata Lobato é colaboradora.

Referências

[1] C. D. Carothers, D. Bauer, and S. Pearce. ROSS: a high-performance, low memory, modular time warp system. In *Proceedings of the fourteenth workshop on Parallel and distributed simulation*, pages 53–60, Bologna, Italy, 2000. IEEE Computer Society.

[2] C. D. Carothers, K. S. Perumalla, and R. M. Fujimoto. The effect of state-saving in optimistic simulation on a cache-coherent non-uniform memory access architecture. In *Proceedings of the 1999 winter simulation conference*, pages 1624–1633, Phoenix, Arizona, United States, Dec 1999.

[3] C. D. Carothers, K. S. Perumalla, and R. M. Fujimoto. Efficient optimistic parallel simulations using reverse computation. In *Proceedings of the thirteenth workshop on Parallel and distributed simulation*, pages 126–135, Atlanta, Georgia, United States, 1999. IEEE Computer Society.

[4] M. J. Chung and J. Xu. An overhead reducing technique for Time Warp. In *Proceedings of the sixth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, pages 95–102, Texas, United States, Oct 2002. IEEE Computer Society.

[5] S. R. Das. Adaptive protocols for parallel discrete event simulation. In *Proceedings of the 28th conference on Winter simulation*, pages 186–193, Coronado, California, United States, 1996. ACM Press.

[6] L. M. D'Souza, X. Fan, and P. A. Wilsey. Modifications to the pGVT algorithm to eliminate acknowledgement messages and improve the GVT broadcast frequency. In *Proceedings of the World Congress on Systems Simulation: Conference on Parallel and Distributed Simulation*, pages 288–292, Coronado, California, United States, Sep 1997.

[7] A. Ferscha. Probabilistic adaptive direct optimism control in Time Warp. In *Proceedings of the 9th workshop on Parallel and distributed simulation*, pages 120–129, Lake Placid, New York, United States, 1995. IEEE Computer Society.

[8] A. Ferscha and G. Chiola. Self-adaptive logical processes: the probabilistic distributed simulation protocol. In *Proceedings of the 27th Annual Simulation Symposium*, pages 78–88, La Jolla, California, United States, 1994. IEEE Computer Society.

[9] R. M. Fujimoto. *Parallel and Distributed Simulation Systems*. John Wiley and Sons, Inc., 2000.

[10] D. R. Jefferson. Virtual Time. *IEEE Transactions on Programming Languages and Systems*, 7(3):404–425, 1985.

[11] N. Kalantery. Tentative Time Warp. In *Proceedings of the 3rd International Euro-Par Conference*, volume 1300 of *LNCS*, pages 458–467. Springer-Verlag, 1997.

[12] R. S. Lobato, R. S. Ulson, M. J. Santana, and R. H. C. Santana. A revised taxonomy for time warp based distributed synchronization protocols. In *Proceedings of the 8th IEEE Internation Symposium on Distributed Simulation and Real Time Applications*, Budapest, Hungary, 2004. IEEE Computer Society. To appear.

[13] R. A. Meyer, J. M. Martin, and R. L. Bagrodia. Slow memory: the rising cost of optimism. In *Proceedings of the fourteenth workshop on Parallel and distributed simulation*, pages 45–52, Bologna, Italy, 2000. IEEE Computer Society.

[14] J. Misra. Distributed discrete-event simulation. *ACM Computing Surveys*, 18(1):39–65, 1986.

[15] R. Noronha and N. B. Abu-Ghazaleh. Early cancellation: an active nic optimization for time-warp. In *Proceedings of the sixteenth workshop on Parallel and distributed simulation*, pages 43–50, Washington, D.C., United States, 2002. IEEE Computer Society.

[16] A. Prakash and A. Subramanian. An efficient optimistic distributed simulation scheme based on conditional knowledge. In *Proceedings of the 6th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 1992.

[17] F. Quaglia. Event history based sparse state saving in Time Warp. In *Proceedings of the twelfth workshop on Parallel and distributed simulation*, pages 72–79, Banff, Alberta, Canada, 1998. IEEE Computer Society.

[18] F. Quaglia and V. Cortellessa. Grain sensitive event scheduling in time warp parallel discrete event simulation. In *Proceedings of the fourteenth workshop on Parallel and distributed simulation*, pages 173–180, Bologna, Italy, 2000. IEEE Computer Society.

[19] R. Radhakrishnan, L. Moore, and P. A. Wilsey. External adjustment of runtime parameters in time warp synchronized parallel simulators. In *Proceedings of the eleventh International Parallel Processing Symposium*, pages 260–266, Geneva, Switzerland, 1997. IEEE Computer Society.

[20] H. Rajaei, R. Ayani, and L.-E. Thorelli. The local time warp approach to parallel simulation. In *Proceedings of the seventh workshop on Parallel and distributed simulation*, pages 119–126, San Diego, California, United States, 1993. ACM Press.

[21] P. L. Reiher, F. Wieland, and D. Jefferson. Limitation of optimism in the time warp operating system. In *Proceedings of the 21st conference on Winter simulation*, pages 765–770, Washington, D.C., United States, 1989. ACM Press.

[22] P. F. Reynolds, C. F. Weight, and J. R. Fidler. Comparative analyses of parallel simulation protocols. In *Proceedings of the 21st conference on Winter simulation*, pages 671–679. ACM Press, 1989.

[23] R. Rönngren, L. Barriga, and R. Ayani. An incremental benchmark suite for performance tuning of parallel discrete event simulation. In *Proceedings of the 29th Hawaii International Conference on System Sciences*, 1996.

[24] H. M. Soliman. On the selection of the state saving strategies in Time Warp parallel simulators. *TRANSACTIONS of the Society for Computer Simulation International*, 16(1):32–36, Mar 1999.

[25] R. Spolon, M. J. Santana, and R. H. C. Santana. Distributed simulation, Time Warp and its variations: Taxonomy and performance evaluation issues. In *Proceedings of the 13th European Simulation Multiconference*, pages 220–227, Warsaw, Poland, 1999. The Society for Modeling and Simulation, Europe Council.

[26] S. Srinivasan and P. F. Reynolds, Jr. Adaptive algorithms vs. Time Warp: an analytical comparison. In *Proceedings of the 27th conference on Winter simulation*, pages 666–673, Arlington, Virginia, United States, 1995. ACM Press.

[27] S. Srinivasan and P. F. Reynolds, Jr. NPSI adaptive synchronization algorithms for PDES. In *Proceedings of the 27th conference on Winter simulation*, pages 658–665, Arlington, Virginia, United States, 1995. ACM Press.

[28] J. S. Steinman. SPEEDES: Synchronous parallel environment for emulation and discrete event simulation. In *Proceedings of the SCS Multiconference on Advances on Parallel and Distributed Simulation*, pages 95–103. ACM Press, 1991.

[29] J. S. Steinman. Breathing Time Warp. In *Proceedings of the seventh workshop on Parallel and distributed simulation*, pages 109–118, San Diego, California, United States, 1993. ACM Press.

[30] J. S. Steinman, C. A. Lee, L. F. Wilson, and D. M. Nicol. Global virtual time and distributed synchronization. In *Proceedings of the 9th workshop on Parallel and distributed simulation*, pages 139–148, Lake Placid, New York, United States, 1995. IEEE Computer Society.

[31] S. C. Tay, Y. M. Teo, and S. T. Kong. Speculative parallel simulation with an adaptive throttle scheme. In *Proceedings of the eleventh workshop on Parallel and distributed simulation*, pages 116–123, Lockenhaus, Austria, 1997. IEEE Computer Society.

[32] C. H. Young, N. B. Abu-Ghazaleh, and P. A. Wilsey. OFC: A distributed fossil-collection algorithm for Time-Warp. In S. Kutten, editor, *Proceedings of the 12th International Symposium on Distributed Computing*, volume 1499 of *LNCS*, page 408, Andros, Greece, 1998. Springer-Verlag.