Aumentando a Escalabilidade da Ferramenta de Visualização de Programas Paralelos Pajé Através de um Sistema de Gerenciamento de Memória*

Diego Luís Kreutz
Laboratório de Sistemas de Computação - LSC
Curso de Ciência da Computação - UFSM
kreutz@inf.ufsm.br

Benhur Stein
Departamento de Eletrônica e Computação
Universidade Federal de Santa Maria – UFSM
benhur@inf.ufsm.br

Resumo

Uma das tarefas mais importantes e úteis no processo de desenvolvimento de programas paralelos é a depuração, onde se inclui a visualização e análise do comportamento que uma aplicação apresentou durante a sua execução. Neste contexto surgiram as ferramentas de visualização de programas paralelos. Estas, por sua vez, esbarram em problemas como a necessidade de manipular enormes quantidades de dados e ao mesmo tempo oferecer uma visualização compreensível e eficiente. Um problema que surge é a pouca disponibilidade de memória em sistemas computacionais convencionais. Esta memória pode não ser o suficiente para alocar todos os objetos visualizáveis. Este artigo busca justamente atacar este problema na ferramenta Pajé. O objetivo é possibilitar que esta ferramenta gerencie e manipule arquivos de rastros de praticamente qualquer tamanho independentemente das limitações do sistema computacional utilizado.

1. Introdução

O desenvolvimento acelerado de tecnologias de rede e da capacidade de processamento e armazenamento de dados dos computadores, vem impulsionando cada vez mais áreas como sistemas de computação de alto desempenho e sistemas distribuídos. Estas áreas tiveram um impulso forte com barateamento desses sistemas advindo do surgimento e concepção dos primeiros aglomerados de computadores.

A computação de alto desempenho tem por objetivo apresentar soluções para problemas complexos e que necessitem de um grande poder computacional. Entre as áreas que derivam problemas desse gênero podem ser citadas: matemática, física, química, biologia, engenharia, inteligência artificial e plataformas de simulação. Simulações como a previsão do tempo e a simulação do comportamento da matéria são dois exemplos clássicos de sistemas que requerem uma capacidade de processamento macisso.

Juntamente com essa grande demanda por alto desempenho e o desenvolvimento dos aglomerados de computadores cresceu o desenvolvimento de ambientes e ferramentas para o projeto, programação e otimização de aplicações paralelas. Neste sentido surgiram ferramentas como ParaGraph [7], Paradyn [11], AIMS [15] e Pajé [13]. Estas tem por objetivo possibilitar aos desenvolvedores a visualização do comportamento apresentado pela aplicação paralela durante sua execução, para descobrir erros de programação, gargalos, problemas de execução em geral e apresentar visualmente pontos passíveis de otimização no programa paralelo.

Estas ferramentas fornecem uma visualização do comportamento dinâmico que uma aplicação apresentou durante sua execução. No entanto, a quantidade de informações necessárias para realizar esta visualização pode ser bastante grande, podendo ultrapassar os limites de memória disponível em um sistema computacional normalmente utilizado. Por isso, um dos maiores problemas enfrentados por ferramentas como Pajé é a tarefa de lidar com a potencialmente grande quantidade de informações geradas pelo rastreamento de uma aplicação paralela. Um arquivo de rastros pode conter alguns *gigabytes* de informações. Manter toda essa quantidade de dados em memória é impraticável utilizando-se sistemas computacionais comuns. Estes atualmente contém geralmente entre 128 e 1024 *megabytes* de memória principal. A memória disponível no sistema é um limitante no caso de visualizações de arquivos de rastros muito grandes.

O objetivo deste artigo é apresentar uma solução, desenvolvida e implementada na ferramenta Pajé, capaz de lidar com esse problema, tornando possível a visualização eficiente e prática de arquivos de rastros de praticamente qualquer tamanho. Além disso, possibilita um maior controle ao usuário sobre o processo de leitura e controle dos dados visualizáveis.

As próximas seções apresentam algumas das ferramen-

*Fomento FAPERGS/CNPq: processos 00/50643.6 e 380049/03-1

tas de visualização de aplicações paralelas mais conhecidas e utilizadas, a ferramenta Pajé, o sistema de gerenciamento de memória desenvolvido, alguns dos resultados obtidos e a conclusão.

2. Ferramentas de Visualização da Execução de Programas Paralelos

Com a necessidade crescente de sistemas de computação de alto desempenho e a intensificação das pesquisas nessa área começaram a surgir ferramentas e ambientes para auxiliar o projeto, desenvolvimento e otimização de programas paralelos.

Os ambientes de programação e bibliotecas de comunicação vêm sendo concebidos e desenvolvidos a ritmos realmente rápidos. Seus objetivos são fornecer recursos para o desenvolvimento de aplicações paralelas e distribuídas, que guiam e auxiliam o processo de implementação e otimização. Como exemplo de ambientes e bibliotecas de programação em plena fase de desenvolvimento e evolução: PVM[1] [14], MPI[2] [6], Topsys [2], ParaScope [3] e DECK[3] [1].

Para complementar estes ambientes, auxiliando o processo de programação, depuração e otimização de aplicações, surgiram ferramentas de visualização e depuração como o TraceView [9], ParaGraph [7], Paradyn [11], AIMS [15], Panorama [10], SvPablo [5], PARAVER [12] e Pajé [13, 4]. Estas ferramentas têm por objetivo ajudar os programadores a encontrar problemas na aplicação, otimizar o desempenho do programa, simular o comportamento da execução da aplicação e localizar erros de programação.

A idéia central destas ferramentas está na visualização da execução de aplicações paralelas [8]. Essas ferramentas de visualização podem ser divididas basicamente em duas classes: as que utilizam rastros de execução e as que utilizam estatísticas de execução. A técnica de rastros de execução baseada em eventos é a mais utilizada. Isso se deve ao fato da qualidade da representação da execução de uma aplicação através da coleta e registro de eventos.

No entanto, um problema que surge quando da utilização de rastros de execução é a quantidade de informações que podem ser geradas em um processo de rastreamento de um programa paralelo, sendo que as aplicações podem gerar desde apenas algumas centenas, até milhares de *megabytes* de dados de rastreamento. Isso representa uma quantidade de dados bastante grande e de difícil manipulação para um sistema computacional comumente utilizado. A quantidade de memória disponível, para uma análise mais rápida e eficiente dos dados coletados, pode ser insuficiente. Neste ponto tem-se nitidamente um possível problema de escalabilidade no processo de visualização baseada em eventos.

[1] Parallel Virtual Machine

[2] Message Passing Interface

[3] Distributed Execution and Communication Kernel

Entre as ferramentas de visualização conhecidas, existem algumas que são capazes de lidar com esse problema. Podem ser citadas ferramentas como ParaGraph [7], AIMS [15] e Paradyn [11] que fornecem visualizações baseadas em eventos; e ferramentas como o SvPablo [5] que utilizam estatísticas de execução ao invés de eventos.

O primeiro conjunto de ferramentas simplemente simula e descarta os dados, ou seja, o usuário não tem a possibilidade de avançar ou recuar na linha de tempo da simulação. Já o segundo conjunto de ferramentas apresenta apenas dados estatísticos da execução do programa paralelo. Neste caso, normalmente não haverá uma grande quantidade de dados a ser manipulada, já que os dados são resumidos. Em muitos casos, esse resumo pode não permitir a identificação de problemas e detalhes da aplicação, causando assim uma redução de sua efetiva utilidade.

Tanto o primeiro como o segundo conjunto de aplicações podem dificultar a tarefa de visualização do usuário. A falta de detalhes da visualização por estatísticas e a falta de flexibilidade da simulação passiva da visualização por eventos não permitem ao usuário facilmente retornar a um ponto desejado da visualização sem prejudicar a análise da execução da aplicação. Isso pode tornar o processo de desenvolvimento e visualização de aplicações paralelas pouco intuitivo e eficiente para a detecção de falhas e otimizações possíveis. A ferramenta Pajé procura amenizar esses problemas através de características como interatividade, escalabilidade e extensibilidade [13].

3. A Ferramenta Pajé

Pajé é uma ferramenta de visualização baseada em eventos. Ela permite a visualização de programas paralelos através da análise e interpretação dos registros dos eventos gerados durante suas execuções. Início de execução, trocas de mensagens, estado dos fluxos de execução e sincronizações são alguns exemplos de tipos de eventos que podem ser registrados em um arquivo de rastros para uma posterior reconstrução do comportamento que a aplicação teve durante sua execução.

Esta ferramenta possui algumas características que a tornam facilmente adaptável a mudanças, escalável e interativa. Um exemplo é a sua arquitetura interna [13]. Pajé é organizado na forma de um grafo de componentes independentes que se comunicam exclusivamente através de protocolos extensíveis bem definidos. A figura 1 ilustra melhor essa arquitetura.

O simulador é o módulo responsável pela construção dos objetos visualizáveis a partir dos registros armazenados no arquivo de rastros. Ele também é um dos componentes mais complexos da ferramenta. Isso por que é o responsável pela identificação das dependências temporais dos dados do arquivo de rastros e a montagem correta dos objetos visua-

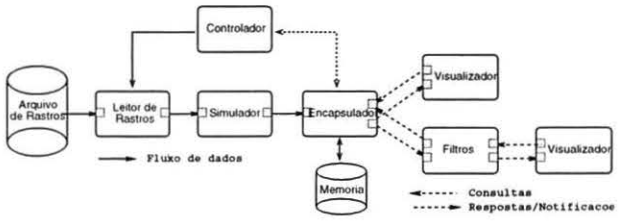

Figura 1. Grafo dos componentes de Pajé.

lizáveis segundo uma hierarquia de controle e herança.

A dependência de dados pode facilmente ser exemplificada pelas comunicações. Quando o simulador recebe um registro indicando o início de uma troca de mensagem ele cria um objeto novo que representará a comunicação entre o emissor e o receptor. O que ocorre é que podem existir centenas ou até milhares de registros entre o tempo do registro que indica o início do processo de comunicação e o registro que indica o final. Logo, o simulador deve armazenar o objeto, que representa a comunicação, até receber todos os eventos referentes a esta troca de mensagem.

Devido a essa dependência de dados, há a necessidade de o simulador receber serialmente todos os registros do arquivo de rastros, do início ao fim. Não havendo a possibilidade de iniciar a simulação em um local diferente do indicador de primeiro registro do arquivo de rastros.

O encapsulador, dentro do fluxo de dados da ferramenta, é o responsável pelo processo de armazenamento dos objetos visualizáveis em memória. Sua função, na versão original de Pajé, é alocar em memória todos os objetos que recebe do simulador. A idéia é que as requisições, feitas sob demanda pelos componentes a frente no grafo, sejam atendidas da melhor forma possível. O armazenamento dos dados em memória permite que o usuário avance ou recue no tempo da visualização de maneira fácil e eficiente.

No entanto, a quantidade de memória disponível no sistema pode não ser o suficiente para armazenar todos os dados em memória. Neste contexto entra o objetivo central deste trabalho. Isso por que na versão original de Pajé, a partir da escolha e abertura do arquivo de rastros, todos os eventos são lidos, simulados e alocados em memória. Se a memória disponível não for o suficiente a ferramenta é finalizada pelo sistema operacional por violação de memória.

Esse foi um dos maiores motivos deste trabalho. A idéia é manter blocos de dados em memória e utilizar pontos de referência para armazenar estados consistentes da visualização e possibilitar a análise e visualização de arquivos de rastros de praticamente qualquer tamanho. Os estados consistentes da visualização são necessários para recuperar um determinado estado da simulação.

É justamente nesse ponto que entra o papel dos pontos de referência. Estes irão ser os responsáveis, entre outras coisas, pelo armazenamento do estado interno do simulador e do leitor de rastros.

Para tanto, foi desenvolvido e incluido um novo módulo no grafo de componentes. Na versão atual, existe um novo componente de gerenciamento de memória. Seu papel é manter comunicação e controle sobre o processo de leitura, simulação, alocação de dados em memória e permitir ao usuário um maior controle sobre o processo de visualização.

Este novo módulo pode ser visto no grafo de comonentes da figura 2. Suas funcionalidades estão diretamente relacionadas ao módulo de controle de Pajé. Através de notificações recebe alertas e ativa processos de controle de memória e manipulação do fluxo de dados da visualização. Nesse novo grafo de componentes o controlador ganha novas funcionalidades e aumenta sua comunicação e controle sobre módulos como o leitor, simulador e o encapsulador.

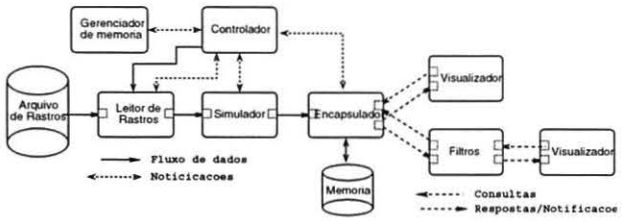

Figura 2. Novo grafo dos componentes de Pajé.

3.1. O processo de geração de imagens consistentes do sistema

Agora o controlador também é o componente responsável pela geração e controle dos pontos de referência. O gerenciador de memória apenas estabelece os parâmetros e delimitações do fluxo de dados.

Após um estudo e identificação dos objetos, estruturas, dados e controles necessários chegou-se a uma arquitetura que seria capaz de suportar estas novas funcionalidades em Pajé. O processo de geração e controle dos pontos de referência é ilustrado na figura 3.

O processo de geração de um ponto de referência consiste em o controlador enviar uma mensagem ao leitor e uma mensagem ao simulador marcando o início do processo de geração da imagem do sistema naquele dado instante de tempo. A partir dessa mensagem o leitor e o simulador codificam seus dados internos e enviam um objeto contendo os dados codificados ao controlador. Este grava os dados recebidos e o tempo de simulação atual em um arquivo que representará o ponto de referência. Por fim, é incluido uma entrada na lista de pontos de referência do controlador indicando o nome do arquivo e o tempo do último evento processado quando da gravação do respectivo ponto de referência.

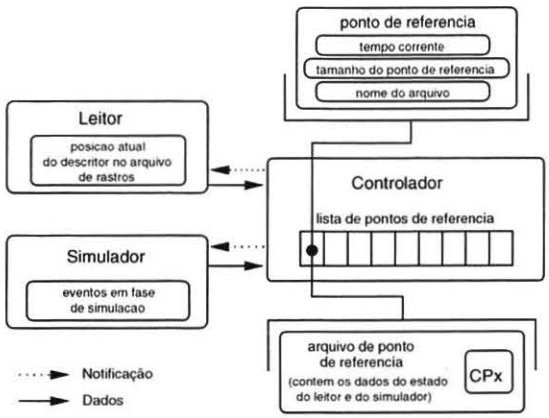

Figura 3. Visão geral do processo de geração de pontos de referência.

Como pode ser facilmente deduzido, cada ponto de referência ocupa algum espaço de armazenamento. Se o usuário estiver visualizando um arquivo de rastros grande provavelmente serão gerados milhares de pontos de referência. Não havendo espaço físico para o armazenamento temporário de todos os pontos de referência incorre-se novamente num problema de limite de memória. Para evitar este problema, é reservado espaço em disco, do montante livre, para a mani-pulação temporária dos pontos de referência. Caso seja preenchido todo o espaço reservado é iniciado um processo de reciclagem, ou seja, são eliminados pontos de referência, intercaladamente, de forma a liberar espaço para a alocação de novos pontos de referência procurando prejudicar o míni-mo possível o desempenho da ferramenta.

A figura 4 ilustra uma seqüência de pontos de referência candidatos a remoção em ordem de prioridade. Quanto menor o identificador maior as chances de ser eliminado. Quando o sistema solicitar a alocação de um novo ponto de referência e o espaço físico reservado para o armazenamento estiver esgotado o ponto de referência com índice 0 será eliminado. E assim sucessivamente até que todo o arquivo de rastros tenha sido simulado e não haja mais a necessidade de gravação de novos pontos de referência. Neste caso, há uma pequena perda de desempenho no processo de visualização da ferramenta Pajé. Mas essa perda é pequena visto que continuam a existir muitos pontos de referência, a uma distância um pouco maior. Além disso, o sistema de gerenciamento de memória do encapsulador, que será visto mais adiante, ameniza essa perda.

3.2. Visualização utilizando os pontos de referência

Um dos objetivos da utilização de pontos de referência foi justamente possibilitar ao usuário a realização de saltos de visualização de uma forma eficiente e prática, principal-

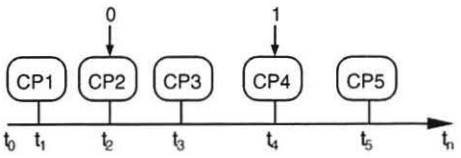

Figura 4. Ilustração dos pontos de referência candidatos a eliminação em ordem de prioridade.

mente para o caso de arquivos de rastros enormes, que não poderiam ser armazenados na memória limitada disponível no sistema computacional em uso.

Para ilustrar melhor o funcionamento do processo de visualização com o uso de pontos de referência é apresentada a figura 5. Neste figura estão em destaque duas regiões de dados. Uma delas é a visualização atual, ou seja, a faixa de dados que está atualmente disponível para o módulo de visualização. A outra área em destaque é a faixa de dados que representa a visualização desejada. Esse é um caso típico onde o usuário deseja retornar no tempo da visualização.

Observando a imagem pode-se perceber que o usuário deseja visualizar uma região entre os pontos de referência de tempo T_1 e T_2, que corresponde ao conjunto de dados entre os tempos T_x e T_y. Enquanto que os dados da visualização atuais representam o intervalo de tempo T_z a T_w.

Para proceder essa operação, atualização dos objetos visualizáveis, o controlador irá escolher o ponto de referência CP1 como ponto de partida. A partir do tempo T_1, que corresponde ao tempo do ponto de referência CP1, serão lidos todos os eventos do arquivo de rastros até ser lido o primeiro evento que possui um carimbo de tempo igual ou superior a T_y.

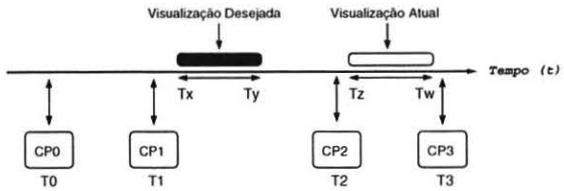

Figura 5. Alternando entre faixas de dados para a visualização.

Devido as novas características e funcionalidade do encapsulador, existe ainda uma chance de os dados entre os tempos T_x e T_y já estarem em memória. Pois, eles estão relativamente próximos ao tempo de visualização atual T_z. Isso por que o encapsulador tenta manter o máximo de dados em memória a cada instante da visualização. Ou seja, descarta faixas de dados somente quando for necessário.

4. O Sistema de Gerenciamento de Memória

Pajé em sua versão original tenta alocar todos os objetos visualizáveis, gerados a partir dos registros do arquivo de rastros, em memória. Isso pode não ser possível em sistemas computacionais restritos.

É principalmente por esse motivo e deficiência que nasceu a idéia de criar um módulo de gerenciamento de memória e ao mesmo tempo apresentar uma interface ao usuário que permitisse algum controle e manipulação maior sobre o processo de visualização utilizando os pontos de referência.

Com isso em mente nasceu o sistema de gerenciamento de memória e a interface da figura 6, que possibilitam o controle e visualização de arquivos de rastros grandes, com quantidades de dados que excedam os limites físicos da memória do sistema computacional em uso.

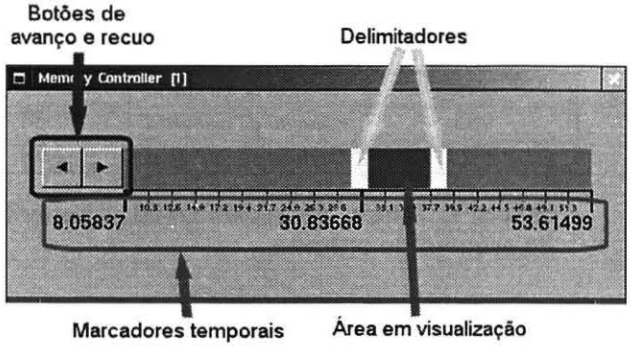

Figura 6. Visão gráfica do módulo de gerenciamento de memória.

Na imagem pode-se verificar a existência de vários controladores, como: os botões de avanço e recuo manual, os limitadores, a área em visualização e os marcadores. Estes controladores são características e recursos que auxiliam a visualização de arquivos de rastros enormes. No decorrer das próximas seções os controladores serão melhor introduzidos.

4.1. Funcionalidades

Inicialmente, quando o arquivo é aberto, são processados os primeiros 10.000 eventos registrados no arquivo. Esse número é modificável e variável de acordo com a capacidade do sistema computacional em uso. A partir desse momento o controle e a manipulação da leitura e dos dados visualizáveis passa para o usuário. Cabendo apenas ao sistema o atendimento as requisições e movimentações solicitadas pelo operador da ferramenta.

A partir da ativação das funcionalidades do gerenciador de memória e da leitura da primeira faixa de eventos, o usuário pode utilizar os recursos deste novo módulo para controlar a visualização do arquivo de rastros. Entre as funcionalidades presentes pode-se citar: a naveração aleatória, aumento e diminuição da área de visulização e automatização de processos de avanço e recuo no tempo.

Navegação aleatória. Esta facilidade, presente no gerenciador de memória, permite que o usuário navegue realizando saltos de visualização passando de uma região X_1 para uma região X_n sem ter que passar pelas regiões intermediárias $(X_2, X_3, X_4, ..., X_{n-1})$. Essa característica fornece uma maior liberdade ao operador da ferramenta permitindo que regiões críticas ou regiões que mantêm alguma relação entre si sejam facilmente visualizáveis e acompanháveis. Muitas vezes falhas e otimizações não são detectáveis pela falta de recursos apropriados no processo de visualização da execução de uma aplicação paralela. Com esta opção de visualização aleatória é no mínimo facilitado a observação de dados temporalmente distantes mas que tem uma relação ou dependência bastante próxima.

Aumento e redução da área de visualização. A área em visualização, do controlador de memória, possui uma funcionalidade especial. É através dela que o usuário tem a possibilidade de definir qual será a área de dados disponível para a janela de visualização a cada instante. Simplesmente clicando sobre um dos delimitadores, esquerdo ou direito, e arrastando para a direita ou esquerda o usuário pode aumentar ou diminur a faixa de dados em visualização. O limite do aumento da faixa de dados em visualização é a disponibilidade de memória do sistema. Logo, o usuário somente conseguirá aumentar a quantidade de dados em memória até atingir o máximo de utilização da memória do sistema. Após esse fato o sistema não permitirá mais o aumento da quantidade de dados em memória disponívies à janela de visualização.

Avanço e recuo automatizado. Este recurso muitas vezes sequer é percebido pelo usuário. Ao invés deste realizar o avanço e recuo na faixa de dados diponível no arquivo de rastros, ocorre um processamento automático. Quando o usuário chega ao final da visualização dos dados disponíveis a janela de visualização, o controlador detecta e ativa o sistema de leitura e simulação para que forneçam mais dados à janela de visualização.

4.2. Utilizando ao máximo a memória disponível no sistema

O principal problema abordado neste trabalho é a possível e provável falta de memória para manter todos os objetos visualizáveis em memória. Isso para arquivos de rastros muito grandes, milhares de *megabytes*. Certamente é desejável ao menos a utilização de praticamente toda a memória disponível e ociosa. Por isso, foi incrementado o módulo de encapsulamento, responsável pela manutenção dos objetos visualizáveis em memória, de forma a manter e

gerenciar o máximo de dados possível em memória.

Quando a ferramenta é carregada o encapsulador verifica o espaço de memória disponível no sistema. À medida que objetos vão sendo simulados e repassados ao encapsulador, são armazenados em memória, enquanto houver memória disponível. A cada nova alocação de dados o encapsulador verifica a quantidade de *bytes* livres. Caso chegue perto dos 95% de utilização da memória é iniciado o processo de reciclagem de objetos, ou seja, os objetos mais antigos e com a menor probabilidade de serem utilizados em um futuro próximo são eliminados para que os novos objetos possam ser armazenados.

Esse processo tenta manter o máximo de dados em memória de forma a otimizar e agilizar a visualização dos eventos registrados no arquivo de ratros. Com isto é possível realizar saltos de visualização, entre duas regiões distintas e distantes, de forma rápida e bastante eficiente caso os dados dessas duas regiões caibam em memória. Caso não caibam, o usuário pode ainda reduzir a faixa de visualização referente as duas regiões tentando tornar possível sua alocação em memória. Essa situação pode ser um caso típico onde o usuário está analizando duas faixas de dados que mantém uma certa dependência e relação entre si, onde possíveis otimização podem ser realizadas ou possíveis falhas podem estar ocorrendo.

5. Alguns Resultados Obtidos

As próximas seções apresentam alguns dados e estatísticas de resultados obtidos com a implementação do módulo de gerenciamento de memória e do controle de alocação de memória aplicado ao encapsulador.

5.1. Tempo de resposta

Certamente uma das diferenças significativas entre a versão original de Pajé e a versão atual é quanto ao tempo de resposta da ferramenta. Em sua versão original o arquivo de rastros é lido e simulado por completo a partir de sua escolha. Na versão atual são lidos apenas os primeiros 10.000 eventos. Devido a isso o tempo de resposta é significativamente menor quando se está lidando com arquivos de rastros que contenham uma quantidade de registros elevada.

Na tabela 1 pode-se observar que o tempo de inicialização da nova versão é praticamente constante. Os dados da tabela representam a média das vinte execuções coletadas. Essa tabela exibe um comparativo entre o tempo de resposta da versão atual e antiga de Pajé em relação ao tamanho do arquivo de rastros. A quantidade de eventos registrados em um arquivo de rastros de tamanho X pode variar.

O tempo de inicialização e resposta corresponde ao tempo necessário para que a ferramenta esteja à disposição do usuário a partir da seleção do arquivo de rastros de entrada. Como na versão antiga dessa ferramenta é realizada a leitura, simulação e armazenamento do arquivo de entrada por completo, não é possível verificar o tempo de resposta necessário para arquivos de rastros maiores que 256 *megabytes*, como pode ser visto na tabela 1. Isso por que é necessário aproximadamente 8 vezes mais memória, para armazenar os objetos visualizáveis, que o tamanho do arquivo de rastros. Como foi utilizado um sistema computacional com 1024 *megabytes* de memória física real e 1024 *megabytes* de memória virtual, totalizando 2048 *megabytes* de memória disponível ao sistema, não havia memória disponível para armazenar todos os dados de arquivos de rastros maiores que 256 *megabytes*.

Tamanho (MB)	Versão antiga	Versão nova
1	2.31	2.58
17	42.78	2.58
55	122.74	2.58
83	182.75	2.58
479	*violação de memória*	2.58
955	*violação de memória*	2.58
1721	*violação de memória*	2.58

Tabela 1. Tempo de inicialização × tamanho do arquivo de entrada.

5.2. Estatísticas de utilização de memória

Os dados apresentados nesta seção foram basicamente estraídos do comando *top* e *ps*. Comandos que acompanham praticamente qualquer sistema operacional da família Unix. Entre suas funções podem ser citadas o fornecimento de informações, de um processo, referentes a utilização de memória e processador.

Para realizar os testes e comparações formam utilizadas três versões de Pajé. A versão original, que não possui sistema de gerenciamento de memória e as duas novas versões. A primeira delas sem o novo gerenciamento de memória no encapsulador e a segunda com. O gráfico da figura 7 permite visualizar um breve comparativo entre as três versões da ferramenta.

Os dados desse gráfico são apresentados na forma de tamanho de arquivo versus utilização de memória de acordo com a versão de Pajé. Como pode facilmente ser observado, a versão original da ferramenta necessita de sistemas computacionais com enormes quantidades de memória para tornar possível a visualização de arquivos de rastros grandes, com centenas de *megabytes* de dados.

Os dados desse gráfico permitem inferir que se o objetivo é utilizar o mínimo de memória necessária deve-se optar pela nova versão de Pajé sem gerenciamento de memória

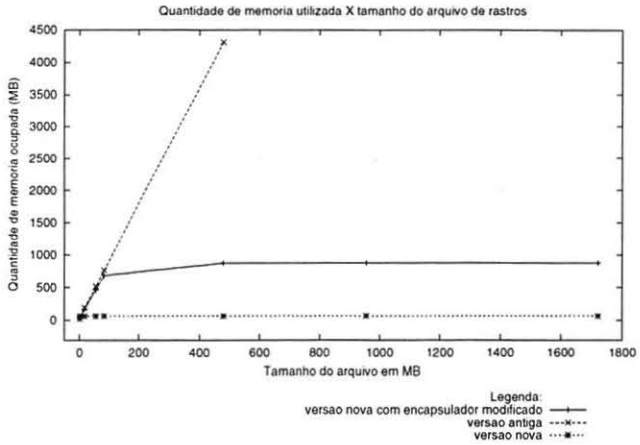

Figura 7. Ocupação de memória de acordo com a versão de Pajé e do tamanho do arquivo de entrada.

no encapsulador. Neste caso, tem-se uma uti-lização de memória praticamente constante e mínima. Por outro lado, caso a finalidade seja eficiência é justificável e recomendável a utilização da nova versão com o encapsulador modificado. Este tentará manter o máximo de objetos possíveis em memória.

O ponto superior do gráfico da figura 7 foi calculado a partir de uma média de utilização de memória dos demais pontos. Serve somente para ilustrar a quantidade aproximada de memória que seria necessária para visualizar um arquivo de rastros de 476 MB de dados utilizando a versão antiga de Pajé.

Em termos de utilização de memória, a nova versão, utilizando o novo sistema de gerenciamento de memória do encapsulador, obtém-se uma melhor utilização da memória disponível no sistema. Somente é contada a memória real. Ou seja, não se utiliza memória virtual por que esta prejudica o desempenho do processo de visualização já que acessos a disco são menos eficientes e mais custosos.

6. Conclusão e Trabalhos Futuros

Aplicações paralelas e distribuídas introduzem novos paradigmas aos desenvolvedores. A forma de programar, pensar, visualizar e otimizar estes programas toma um rumo diferente da tão conhecida programação sequêncial.

Estas aplicações requerem novas técnicas e métodos para a depuração e visualização do comportamento dinâmico que o programa teve durante sua execução. Por isso surgiram ferramentas como Pajé, Paradyn, AIMS e Paragraph. Ferramentas estas que disponibilizam ao usuário meios de encontrar falhas e visualizar locais passíveis de otimização de programas paralelos.

No entanto, o rastreamento de uma aplicação paralela ou distribuída pode gerar desde alguns *megabytes* de dados até dezenas de *gigabytes* de dados. Tudo depende do tempo de execução e da quantidade de informações que se deseja rastrear. Logo, surge a difícil tarefa de se lidar com uma potencial grande quantidade de informações.

Este problema é agravado ainda mais quando o usuário tem a possibilidade de navegar na linha do tempo da execução do programa paralelo que está sendo visualizado, visto que em grande parte das visualizações não é possível manter todos os dados da visualização em memória.

Dentre as ferramentas conhecidas, existe um conjunto de ferramentas de visualização e depuração de programas paralelos que adotam políticas para manipulação de enormes volumes de informações. Essas políticas normalmente são classificadas em simulação dinâmica dos dados do rastreamento ou apresentação resumida em forma de estatísticas de execução. Em ambos os casos há perdas na visualização.

No primeiro caso o usuário não consegue associar regiões de visualização com facilidade. Já que não é possível avançar e recuar no tempo de simulação de forma simples e prática. Em se tratando de visualização baseada em estatísticas tem-se uma perda considerável no nível de detalhes e precisão que o programador terá ao seu dispor para encontrar possíveis problemas e locais que podem tornar a execução da aplicação mais eficiente.

Por isso o objetivo central deste trabalho foi justamente implementar funcionalidades de gerenciamento de memória, na ferramenta Pajé, capazes de manter estados da visualização e possibilitar a visualização de grandes quantidades de dados eficientemente, através da utilização de pontos de referência.

O novo módulo de gerenciamento de memória permite ao usuário controlar não somente os dados em memória mas também possibilita um maior controle sobre o processo de leitura e visualização dos rastros como um todo. O incremento realizado no encapsulador possibilita a utilização de praticamente toda a memória disponível no sistema a fim de aumentar o desempenho de Pajé durante o processo de visualização.

Estes novos recursos e funcionalidades agregados a Pajé permitem a visualização eficiente e prática de arquivos de rastros de praticamente qualquer tamanho. A ferramenta ganha em escalabilidade. Além disso, esta nova versão da ferramenta fornece flexibilidade e robustez ao usuário durante o processo de visualização do comportamento dinâmico que a aplicação apresentou durante sua execução. Funcionalidades como avanço e recuo no tempo da visualização são eficientemente realizáveis.

Por fim, este trabalho teve como intuito atacar a escalabilidade da ferramenta Pajé a nível de visualização e manipulação de arquivos de rastros enormes. Os resultados obtidos se mostraram satisfatórios. Hoje é possível visualizar arqui-

vos de rastros de praticamente qualquer tamanho. Isso sem perder funcionalidades de avanço e recuo no tempo dos dados disponíveis. Além disso, agora o usuário também tem a possibilidade de realizar saltos de visualização e exercer um controle maior sobre o processo de visualização.

Trabalhos Futuros. Devido à necessidade crescente de ferramentas de visualização que auxiliem o usuário a encontrar falhas e locais passíveis de otimização na sua aplicação deseja-se continuar aumentando as funcionalidades, a eficiência e a genericidade da ferramenta Pajé. Dentro desta perspectiva surgem idéias como a implementação de um sistema de processamento em segundo plano. O objetivo é utilizar fluxos de execução extras para a predição e processamento de informações enquanto o processador não estiver sendo requisitado pelo usuário. Ocupando melhor o sistema computacional e antecipando o processamento de futuras solicitações de dados.

Danto continuidade surgem ainda idéias como a criação de fluxos de execução independentes por módulo, visto que atualmente existe um único fluxo de execução, podendo haver um ganho significativo em desempenho da ferramenta como um todo.

Por fim, surge ainda a sugestão de implementação de métodos de filtragem e refinamento de dados e a paralelização da ferramenta. O primeiro permitiria a visualização a diferentes níveis de abstração e o segundo possibilitaria coleta dinâmica de rastros de execução e aumentaria a escalabilidade da ferramenta em termos de capacidade de processamento e eficiência.

Referências

[1] Marcos Barreto, Rafael Ávila, Fábio Oliveira, Ricardo Cassali, and Philippe Navaux. Deck: an environment for parallel programming on clusters of multiprocessors. In *SBAC-PAD*, 2000.

[2] T. Bemmerl and Peter Braun. Visualization of message-passing parallel programs with the topsys parallel programming environment. *Journal of Parallel and Distributed Computing*, 18(2):118–128, 1993.

[3] Keith D. Cooper, Mary W. Hall, Robert T. Hood, Ken Kennedy, Kathryn S. McKinley, John M. Mellor-Crummey, Linda Torczon, and Scott K. Warren. The ParaScope parallel programming environment. *Proceedings of the IEEE*, 81(2):244–263, 1993.

[4] J. Chassin de Kergommeaux and B. de Oliveira Stein. Pajé : an extensible environment for visualizing multithreaded programs executions. *Euro-Par 2000 Parallel Processing, Proc. 6th In-ternational Euro-Par Conference*, 1900:133–140, 2000.

[5] Luiz A. de Rose and Daniel A. Reed. Svpablo: A multi-language architecture-independent performance analysis system. *International Conference on Parallel Processing*, September 1999.

[6] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. MIT Press, Cambridge, Massachusetts, USA, 1994.

[7] M. T. Heath and J. A. Etheridge. Visualizing the performance of parallel programs. *IEEE Software*, 5:29–39, 1991.

[8] Dieter Kranzmueller and Jens Volkert. Why debugging parallel programs needs visualization? In *IEEE Symposium on Visual Languages*, September 2000. `http://wwwmath.uni-muenster.de/cs/u/guidow/CONF/wsvl2000/`.

[9] Allen D. Malony, David H. Hammerslag, and David J. Jablonowski. Traceview: A trace visualization tool. *IEEE Software*, 8(5):19–28, September 1991.

[10] John May and Francine Berman. Panorama: A portable, extensible parallel debugger. In *Proceedings of ACM/ONR Workshop on Parallel and Distributed Debugging*, pages 96–106, San Diego, California, 1993.

[11] Barton P. Miller, Mark D. Callaghan, Joanthan M. Cargille, Jeffrey K. Hollingsworth, R. Bruce Irvin, Karen L. Karavanic, Krishna Kunchithapadam, and Tia Newhall. The paradyn parallel performance measurement tool. *IEEE Computer*, 28(11):37–46, 1995.

[12] V. Pillet, J. Labarta, T. Cortes, and S. Girona. PARAVER: A tool to visualise and analyze parallel code. In *Proceedings of WoTUG-18: Transputer and occam Developments*, volume 44, pages 17–31, Amsterdam, 1995. IOS Press.

[13] Benhur Stein. *Visualisation interactive et extensible de programmes parallèles à base de processus légers*. Thèse de doctorat en informatique, Université Joseph Fourier, France, October 1999.

[14] V. S. Sunderam. PVM: a framework for parallel distributed computing. *Concurrency, Practice and Experience*, 2(4):315–340, 1990.

[15] J. C. Yan. Performance tuning with AIMS — an Automated Instrumentation and Monitoring System for multicomputers. In *Proceedings of the 8th international conference on Supercomputing*, pages 625–633. ACM, 1994.