Proposta de um Formato Binário para Rastro Pajé

Vinícius A.Herbstrith, Lucas Mello Schnorr
{vaherbstrith, schnorr}@inf.ufrgs.br
Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

Resumo

Este artigo apresenta um formato binário para arquivos de rastro em Pajé, almejando principalmente a otimização da leitura através da estrutura binária, de formar a proporcionar um ganho de desempenho para ferramentas que fazem uso do formato para análise e visualização de rastros. A proposta inclui a implementação de dois conversores entre o formato textual e binário e um componente que promove a leitura do arquivo de rastro binário integrado na ferramenta de visualização `PajeNG` e os ganhos em desempenho da leitura do formato binário em relação ao formato textual.

1. Introdução

Com a difusão de aplicações paralelas e distribuídas surge a necessidade da criação de ferramentas que auxiliam na depuração e análises de desempenho destas aplicações. Para tal, se faz a seleção de eventos representativos que devem ser registrados durante a execução em arquivos de rastro de execução [7]. Dentre as informações que compõem estes eventos, temos identificação, tempo do registro, estados de variáveis e outros dados que sejam de interesse para a análise da aplicação em questão.

O formato de rastro Pajé é largamente utilizado para a análise e visualização de rastros de aplicações. O atributo de destaque do formato é a extensibilidade, possibilitando seu uso nos mais variados ambientes de execução, com demandas e requisitos diferenciados, sem ser necessário modificar componentes internos da ferramenta Pajé. Como ponto negativo do formato, temos a sua representação textual, que implica em impactos negativos no desempenho tanto no momento de registro de eventos, gerando uma maior intrusão, como durante a leitura dos arquivos de rastro enquanto processados para a sua visualização e análise. Com o crescente aumento do poder de paralelismo dos computadores de hoje em dia [2], combinado com aplicações mais complexas que geram um número muito grande de eventos durante sua execução, os pontos negativos do Pajé acabam se sobressaindo ainda mais, apresentando uma fraca escalabilidade.

Este artigo apresenta um formato binário para o arquivo de rastro Pajé, como forma de obter uma otimização na leitura de arquivos Pajé. A seção 2 trata da geração dos arquivos em binário, fazendo uso da biblioteca de rastros libRastro [1] e Poti [6], e a implementação da leitura destes arquivos na ferramenta de visualização de rastros PajeNG [5]. A seção 2.1 apresentam com mais detalhes a representação dos eventos Pajé em seu novo formato binário. Por fim, a seção 3 apresenta os resultados dos experimentos feitos sobre a leitura de arquivos de rastros e a comparação dos tempos do formato binário contra a representação Pajé textual padrão.

2. Proposta e implementação

A proposta inclui a implementação de dois conversores e um componente de leitura binário para a ferramenta de visualização PajeNG. Estes três componentes são detalhados na Figura 1 em pontilhado, onde temos os dois conversores (*paje2rst* e *rst2paje*), desenvolvidos dentro da biblioteca Poti com a linguagem C, e o leitor binário (*PajeRastroReader*), desenvolvido na ferramenta PajeNG em C++. Os outros componentes são aqueles que já fazem parte da PajeNG.

As informações do arquivo binário são salvas usando a biblioteca LibRastro, uma biblioteca de geração de arquivos de rastro. A LibRastro tem como principais características uma baixa intrusão na coleta do rastro e flexibilidade nas definições de eventos, características estas que coincidem com o nosso objetivo e requisitos do formato Pajé. Temos então, um arquivo de rastro gerado na LibRastro que contém informações equivalentes ao rastro original no formato Pajé.

A biblioteca Poti – responsável por implementar o formato de rastro Pajé – foi utilizada para a implementação da criação do rastro no formato binário, usando como suporte a libRastro. Foram feitas modificações na biblioteca com o objetivo de gerar a mesma saída Pajé da Poti no

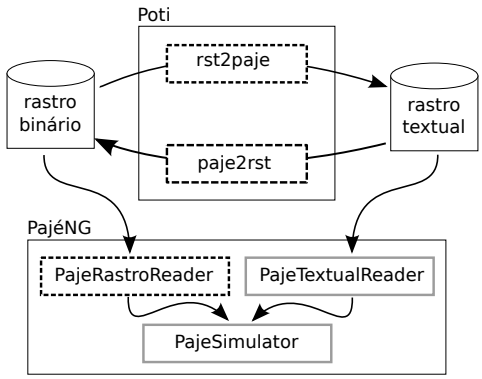

Figura 1. Componentes Poti implementados.

novo formato binário proposto. É possível gerar o arquivo com dois métodos distintos. Um deles é a tradução direta do arquivo textual Pajé fazendo uso da libRastro, nomeado `standard`. O segundo método, chamado `reference`, busca obter uma otimização quanto ao tamanho do arquivo gerado através do uso de referências do tipo int short para dados do tipo string que costumam se repetir durante o arquivo de rastro, gerando um arquivo binário de tamanho reduzido.

Como a execução de aplicações em sistemas paralelos e distribuídos é uma tarefa custosa, é de grande interesse promover ao usuário a possibilidade de converter rastros gerados anteriormente. Para isto, foram criadas duas ferramentas dentro da Poti: `rst2paje` e `paje2rst`.

A ferramenta `paje2rst` faz a tradução de arquivos do formato textual Pajé gerados com a Poti para o novo formato binário. Na implementação deste, foram usados Flex [3] e Bison [4], para fazer a análise léxica e sintática da entrada textual, e as funções da nova biblioteca Poti para gerar o arquivo de saída em formato binário equivalente.

A ferramenta `rst2paje` faz a tradução no sentido oposto, dado o arquivo binário gerado na Poti, é gerado o arquivo no formato textual equivalente. Para este foram usadas as funções de leitura de arquivos da própria libRastro juntamente com funções que fazem a organização dos dados de eventos de acordo com o formato textual.

A leitura e análise do novo formato é feita dentro da PajeNG, uma ferramenta de visualização de rastros no formato Pajé. O componente PajeRastroReader foi criado para realizar a decodificação dos arquivos no formato binário proposto, tanto na versão `standard` quanto na versão `reference`.

Para se tirar proveito da estrutura binária do formato, também foi criado o componente PajeRastroTraceEvent que armazena os dados de acordo com os tipos de dados usados na libRastro(int, doubles e char). O objetivo deste componente é eliminar conversões de dados para o formato string, tarefa essa que se torna muito custosa tendo em vista o grande número de eventos que ocorre em um arquivo de rastro. Outra tarefa da classe PajeRastroTraceEvent é a reordenação dos dados dos eventos, que se torna necessária uma vez que a libRastro faz o armazenamento dos dados em vetores de acordo com o tipo de dado, perdendo a relação de ordem que existe entre o evento Pajé e sua definição de cabeçalho, o que nos leva a ter uma decodificação diferenciada.

2.1. Representação em binário do arquivo Pajé

Um arquivo de rastro Pajé se divide em duas partes: definições de cabeçalho e eventos. As definições de cabeçalho contêm uma lista de tipos de eventos a serem registrados e seus parâmetros, definindo números de identificação, nome e os dados que são usados como parâmetros. Logo após o cabeçalho, temos os eventos datados que foram registrados durante o rastro do programa, com o número de identificação do tipo de evento e os parâmetros, de acordo as regras definidas no cabeçalho.

A libRastro, por ser uma biblioteca de criação de rastros, faz todos os seus registros como eventos, onde cada evento possui um valor numérico para identificação do seu tipo, seguido de vetores com os dados que foram registrados.

Abaixo temos um exemplo de uma definição de cabeçalho de um arquivo de rastro Pajé em seu formato textual.

```
%EventDef PajeDefineContainerType 0
%    Alias string
%    Type string
%    Name string
```

Abaixo a versão binária do mesmo trecho de cabeçalho.

```
type: 999
 strings-> {0}{11}{0}{3}{0}{2}{0}
```

No campo `type` é usado o valor 999 para identificar, durante a leitura, que o evento binário deve ser tratado como uma definição de cabeçalho. As informações de nome de campo e tipo de dado do cabeçalho Pajé são representadas por valores inteiros na versão binária, valores estes definidos em uma enumeração que faz parte da `PajeNG` e da Poti. No exemplo, o primeiro valor do vetor de dados é a identificação do evento, que é o mesmo valor usado na representação textual (0, no exemplo), seguido por tuplas nome do campo e tipo de dado. No exemplo, a primeira tupla da sequência é Alias e string, representadas no arquivo binário pelos valores 11, o valor da enumeração para o campo Alias, e 0, valor da enumeração para o tipo de dado `string`. Os quatro valores seguintes do vetor de dados em

binário correspondem às outras duas tuplas, com os valores 3 e 2 representando os campos Type e Name respectivamente e 0 o tipo string.

Abaixo, um exemplo de um evento datado no formato Pajé textual, evento este que corresponde à definição de cabeçalho anterior.

```
0 1 0 HOST
```

Abaixo temos a representação com a libRastro para um evento datado obtido com o conversor `paje2rst` com o método `standard`.

```
type: 0
 strings-> {1} {0} {HOST}
```

Diferentemente da definição de cabeçalho, em eventos a informação do campo `type` se refere à identificação do evento, com o valor 0 no exemplo indicando que o evento é do tipo `PajeDefineContainerType`. Em seguida, temos os vetores com os dados do evento datado agrupados de acordo com o seu tipo, neste caso todos em um vetor de strings.

A seguir temos a representação do mesmo evento datado obtido também através do `paje2rst`, mas usando o método `reference`.

```
type: 888
 strings-> {HOST}
type: 888
 strings-> {0}
type: 888
 strings-> {1}
type: 0
 u_int16_ts-> {2} {1} {0}
```

Os eventos em que o campo `type` tem o valor 888 definem que o evento contém uma string à ser usada como referência. No evento de exemplo temos três dados do tipo string, logo temos três eventos de referência que armazenam as strings. Abaixo destes eventos existe o evento Pajé de fato, que faz referência às strings que são indicadas de acordo com a ordem em que elas aparecem no arquivo como um valor inteiro de 16 bits.

Embora um número maior de eventos seja gerado, temos uma redução significativa do tamanho total do arquivo com este método devido a grande ocorrência de dados do tipo string.

3. Resultados sobre o desempenho do `PajeRastroReader`

A análise de desempenho da leitura do formato binário foi feita sobre dois arquivos de rastro de duas aplicações distintas, cada um em três versões, textual – a versão original – , binária e binária reduzida – ambas obtidas com a ferramenta paje2rst. Uma das aplicações é uma fatoração LU que gerou um arquivo de rastro de 0.77 GB, a outra é um cálculo de gradiente conjugado com um rastro de tamanho 2.1 GB. A máquina usada para os testes possui processador Intel i7-4770 CPU (3.40GHz), HD 1TB SATA II (3.0Gb/s) e 8GiB DIMM DDR3(1.6GHz). Foram feitas 30 execuções para cada versão, binárias e textual, dos dois arquivos, de forma a se obter uma amostra significativa para comparação sobre o desempenho do formato proposto.

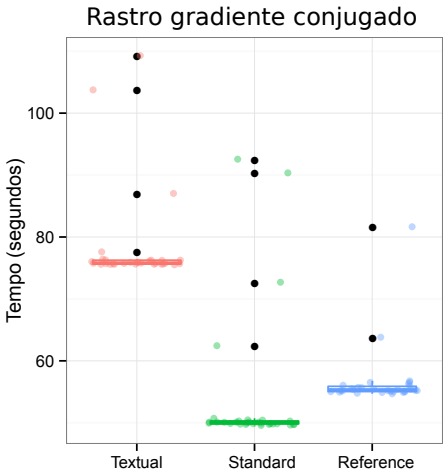

Figura 2. Resultados dos experimentos com o arquivo de rastro gradiente conjugado

Em ambos os gráficos, podemos notar um ganho de desempenho de aproximadamente 32% do formato binário `standard` em relação ao formato textual original. Com a versão `reference` o ganho foi de 27% em relação ao formato textual. Sobre o tamanho do arquivo, a diferença de tamanho entre o arquivo original e o do método `reference` foi de 48% para o arquivo de rastro da fatoração LU e 43% para o rastro do gradiente conjugado.

4. Conclusões

Neste artigo apresentamos o formato binário Pajé. Como esperado, a estrutura do arquivo binário promoveu uma melhora no desempenho da leitura do arquivo de rastro. Esperava-se ganhos maiores com a redução do tamanho do arquivo usando o método `reference`, mas o tempo de processamento das referências acaba sendo maior que a simples leitura dos dados em binário com o método `standard`. Ainda sim, ambos os métodos de geração de arquivos resultam em um ganho de desempenho de leitura sobre o formato textual, reduzindo em até 32% o tempo de leitura para arquivos gerados com o método `standard`.

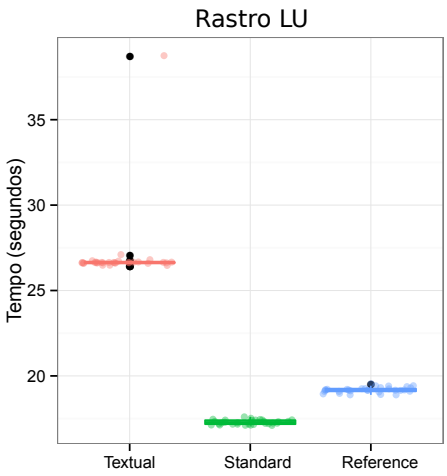

Figura 3. Resultados dos experimentos com o arquivo de rastro LU

A partir destes resultados positivos, podemos buscar ainda outras otimizações para o formato binário. Como possibilidade, temos o uso de técnicas de compactação que tornariam o arquivo de rastro menor e consequentemente um melhor tempo de leitura com técnicas de descompactação eficientes aplicadas durante a leitura do arquivo binário.

Referências

[1] G. J. da Silva, L. M. Schnorr, and B. Stein. Jrastro: A trace agent for debugging multithreaded and distributed java programs. In *Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing*, pages 46–54. Los Alamitos: IEEE Computer Society, 2003.

[2] T. Geller. Supercomputing exaflop target. In *Comm. of the ACM*, page 54(8). ACM New York, NY, USA, August 2011.

[3] V. Paxson. Flex. `http://flex.sourceforge.net`, 2014.

[4] T. G. P. Robert Corbett. Bison. `http://www.gnu.org/software/bison`, 2014.

[5] L. M. Schnorr. Pajeng. `http://github.com/schnorr/pajeng`, 2014.

[6] L. M. Schnorr. Poti. `http://github.com/schnorr/poti`, 2014.

[7] B. Stein. Depuração de programas paralelos. In *I Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul*, pages 151 – 176, Gramado, RS, 2001. Sociedade Brasileira de Computação (SBC).