Ferramentas de Visualização para Análise de Desempenho de Sistemas de Computação Distribuída

Leonardo L. Fernandes*, Ricardo Anido

[1] Instituto de Computação (IC) – Universidade Estadual de Campinas (UNICAMP)
Avenida Albert Einstein, 1251, Caixa Postal 6176
13083-970 Campinas, SP - Brasil

leonardo.fernandes,ranido@ic.unicamp.br

Abstract. *This paper presents a set of profiling tools for performance analysis of MPI-based parallel systems. From trace files we generate some new forms of visualization of the analyzed system execution. We propose animations that represent system execution and message traffic, as well as statistical graphics for the whole duration of the application.*

Resumo. *O presente artigo apresenta um conjunto de ferramentas de* profiling *para análise de desempenho de sistemas de computação paralela baseados na MPI. A partir de arquivos de trace são geradas algumas novas formas de visualização do funcionamento do sistema analisado. São propostas animações que representam o funcionamento do sistema e as trocas de mensagens, bem como gráficos estatísticos referentes a todo o tempo de execução da aplicação estudada.*

1. Introdução

A utilização de sistemas de processamento paralelo apresenta um crescimento significativo nos últimos anos. Com a diminuição do custo dos equipamentos e melhoria das tecnologias de comunicação, os *clusters* tornam-se a melhor opção para a maioria das aplicações que demandam alto desempenho. Em comparação com computadores de grande porte, os *clusters* modernos apresentam custos de implantação e manutenção significativamente menores. Além disso, pode-se ampliar a capacidade de processamento de um *cluster* simplesmente acrescentando novos equipamentos ao sistema.

Com a ampla utilização de *clusters*, o desenvolvimento de programas eficientes para sistemas multiprocessados é fundamental. Este tipo de programa, pela própria natureza dos sistemas distribuídos, tende a ser bastante complexo. Muitas vezes, centenas ou mesmo milhares de processos diferentes estão ativos no sistema. Os processos envolvidos precisam trocar mensagens entre si constantemente para realizar suas tarefas. Nos sistemas sem memória compartilhada, como é o caso dos *clusters*, a troca de mensagens é a única forma de comunicação e sincronização entre os processos. Tal complexidade dificulta significativamente o projeto e implementação das aplicações de processamento

*Bolsista CNPQ - Brasil

paralelo, bem como a identificação de eventuais ineficiências dos programas desenvolvidos.

Profiling é o termo utilizado para definir técnicas que facilitam o acompanhamento do funcionamento de programas computacionais. Tais técnicas têm por objetivo auxiliar aqueles que desenvolvem as aplicações a compreender melhor o comportamento do programa em execução. Como o próprio nome sugere, as ferramentas de *profiling* buscam traçar um "perfil" do funcionamento do programa, estabelecendo quais as rotinas e operações que consomem o maior tempo, quanto tempo consomem, quais rotinas acionam quais outras, como se dá o fluxo de dados, entre outras características, buscando sempre facilitar a análise do sistema.

Estas técnicas são especialmente importantes em sistemas com muitos processos, onde mesmo os analistas mais experientes encontram dificuldades para identificar eventuais ineficiências. Os sistemas distribuídos nos quais existe processamento paralelo costumam apresentar um nível de complexidade considerável, portanto representam uma importante aplicação para sistemas de *profiling*.

O processo de *profiling* em geral envolve a coleta de um determinado conjunto de dados em tempo de execução e a exibição de tais dados de forma que facilitem a identificação de problemas relativos a performance da aplicação em estudo. Especialmente em sistemas complexos com muitos processos envolvidos é fundamental uma apresentação dos dados que facilite a identificação de ineficiências.

Uma das principais interfaces para comunicação utilizadas em sistemas paralelos é a *Message Passing Interface* (MPI) [Message Passing Interface Forum, 1994]. Trata-se de um conjunto de rotinas que realizam tarefas de comunicação entre os equipamentos dos *clusters*, tais como envio e recebimento de mensagens, *broadcast*, *multicast* e sincronização. A MPI oferece também uma interface para *profiling* que simplifica a obtenção de informações em tempo de execução.

O desenvolvimento de ferramentas de visualização apropriadas para a análise de desempenho de sistemas concorrentes representa uma área ainda em desenvolvimento. Encontram-se disponíveis algumas ferramentas comerciais [Etnus LLC., 2003, Nagel et al., 1996] e também alguns trabalhos acadêmicos na área [Heath and Etheridge, 1991, Calzarossa et al., 1998, Reed et al., 1993, Carothers et al., 1997].

Objetiva-se possibilitar uma análise diferenciada dos sistemas através das diversas formas de visualização. É apresentado um conjunto de ferramentas de visualização para sistemas de computação paralela baseados em MPI. É utilizado para validação o SORTTS [Müller Junior and Anido, 2002], um sistema automatizado de rastreamento de objetos usando câmeras de vídeo.

Na próxima seção, é descrita brevemente a metodologia utilizada na coleta dos dados que são utilizados pelas ferramentas de visualização. A seguir são apresentadas as ferramentas de visualização desenvolvidas. Por fim é apresentado o sistema SORTTS e são descritos alguns resultados da utilização das ferramentas descritas na instrumentação desta aplicação.

2. Coleta de Dados

Uma parte fundamental do *profiling* de um sistema é a coleta de dados em tempo de execução. Devem ser obtidas em tempo de execução informações referentes às comunicações e ao funcionamento de cada processo.

Tais informações devem ser obtidas de maneira precisa e sincronizada, para que se obtenha um conjunto de dados confiável. Além disso, a "intrusão" das ferramentas de profiling deve ser a menor possível, para que a obtenção dos dados não interfira significativamente no funcionamento da aplicação em estudo.

Para as ferramentas descritas neste artigo, é utilizada a extensão a MPI chamada MPE [Chan et al., 1998]. Esta extensão inclui bibliotecas de *profiling* e facilidades para a geração de arquivos de *trace*. Com o auxílio da MPE, é gerado um arquivo de *trace* no formato CLOG. O formato CLOG é baseado em eventos, ou seja, trata-se de um conjunto de eventos com os respectivos rótulos de tempo.

A MPE permite a instrumentação de todas as rotinas MPI automaticamente, sem que se altere o código original. Também é possível a definição de outros estados e eventos através de chamadas a funções específicas no código da aplicação estudada.

O arquivo CLOG pode posteriormente ser convertido para o formato SLOG-2 com o conjunto de aplicações `slog2sdk` [Chan, 2000]. O formato SLOG-2 é baseado em objetos Java denominados *drawables*, e não em eventos como o formato CLOG. Tais objetos podem representar um ou mais estados de processos ou mensagens do sistema. Um *drawable* indica, entre outras características, em que processo acontece e quais os tempos de início e fim no caso de tratar-se de um estado ou os processos de origem e destino e os tempos de envio e entrega no caso de mensagens. O formato foi desenvolvido para ser utilisado por ferramentas de visualização. Uma característica fundamental do formato SLOG-2 é sua escalabilidade. É utilizada uma estrutura baseada em árvores binárias que classifica os objetos no arquivo de acordo com sua posição no tempo. Tal estrutura possibilita uma navegação eficiente mesmo em arquivos de alguns gigabytes de tamanho. Os arquivos SLOG-2 são processados pelas ferramentas propostas.

3. Ferramentas de Visualização

Esta seção apresenta as ferramentas de visualização desenvolvidas. São demonstradas as ferramentas de animação e de gráficos estatísticos de execução de programas paralelos. Todas as ferramentas utilizam arquivos no formato SLOG-2 como entrada.

3.1. Animação de Processos

Esta ferramenta, ilustrada na figura 1 possibilita que se acompanhe a execução de um sistema de computação paralela.

Cada quadrado na figura representa um processo diferente. A legenda a direita da figura apresenta o significado de cada uma das cores. São mostrados pela ferramenta o tempo de execução representado a cada instante e alguns botões de controle.

A legenda de cores representa todas as chamadas a rotinas MPI e estados definidos pelo usuário através do uso da MPE.

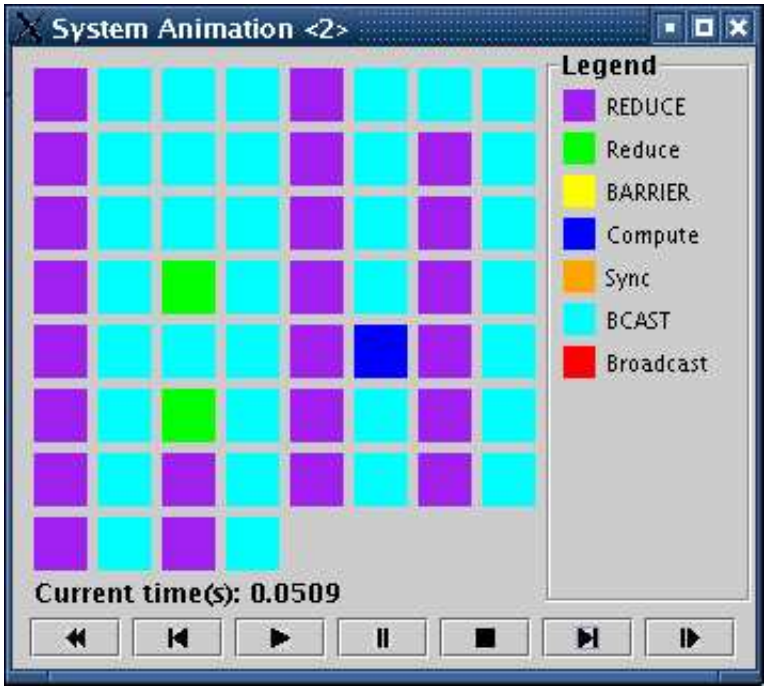

Figura 1: A ferramenta de animação de processos.

Os botões de controle permitem a exibição da animação em velocidade normal, buscando aproximar-se ao tempo real ou em *slow motion*, possibilitando uma melhor visualização do funcionamento. Além disso, pode-se utilizar os botões de avanço e retrocesso rápidos para uma navegação eficiente por toda a animação de forma que se possa encontrar rapidamente pontos de interesse da execução.

O número de quadros de animação por segundo de tempo de execução do sistema em estudo é configurável. Isto permite que seja possível um maior detalhamento em etapas importantes da animação.

Muitas ferramentas de análise de desempenho [Nagel et al., 1996, Zaki et al., 1999, Heath and Etheridge, 1991] utilizam gráficos de linhas de tempo ou diagramas de Gannt para representar os sistemas em execução. Em comparação com este tipo de gráfico, a animação apresenta vantagens como a identificação de repetições de comportamentos de processos ao longo do tempo, bem como a identificação de grupos de processos que se comportam de maneira semelhante. Pode-se ainda analisar as diferenças em comportamentos de processos pertencentes a tais grupos. Ainda em comparação com os gráficos de linha de tempo, a animação permite uma melhor percepção de detalhes em trechos específicos da execução, ao passo que linhas de tempo permitem uma melhor visão global de todo o sistema.

3.2. Estatísticas de Tempo de Espera

Uma das principais questões na performance de sistemas distribuídos é o tempo de espera de um processo pelos demais. É proposta uma forma de visualização que possibilita identificar claramente esta característica dos processos. Através da definição de estados de espera como o recebimento de uma mensagem, é calculado o tempo de espera de cada

processo pelos demais. Este tempo é então comparado com o tempo total de execução e com o tempo total de espera de cada processo.

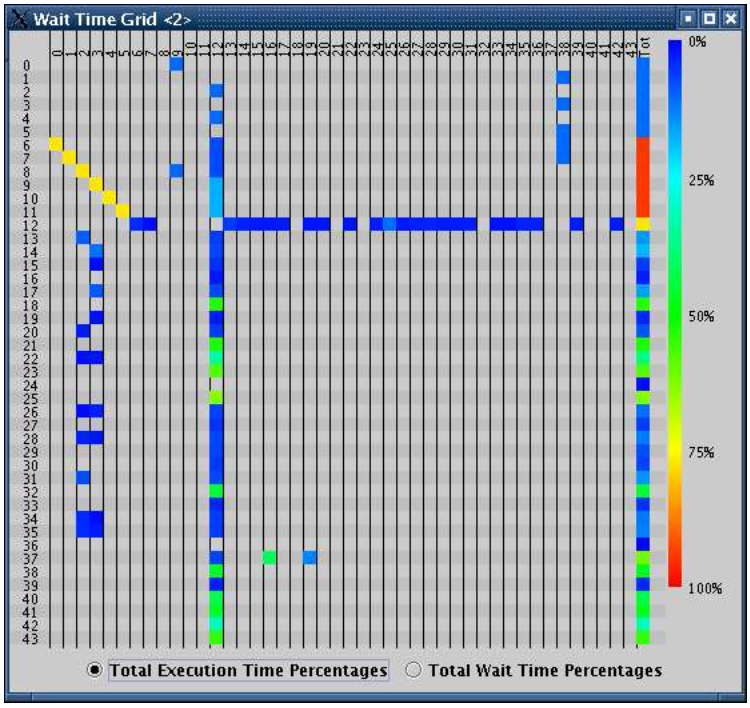

Figura 2: Tempo de espera relativo ao tempo total de execução.

A figura 2 apresenta percentagens de tempo de espera referentes ao tempo total de execução. Cada linha i do gráfico representa um processo. Cada coluna j representa um processo pelo qual se espera uma determinada porcentagem de tempo. A cor de uma determinada célula $[i, j]$ no gráfico representa a percentagem do tempo de execução que o processo i espera pelo processo j. O gradiente de cores a direita do gráfico, permite que se identifique facilmente quais processos esperam por mais tempo e por quais processos. Além disso, também se torna simples identificar que processos fazem com que os demais esperem muito. Desta forma, é possível identificar, por exemplo, uma má distribuição da carga de processamento se vários processos ficarem esperando muito tempo por alguns poucos. A última coluna do gráfico da figura 2 representa o tempo total de espera de cada processo em comparação ao tempo total de execução, possibilitando a identificação dos processos que esperam mais de uma forma geral.

Na figura 3 é apresentado um gráfico semelhante, com a diferença de que os dados são relativos ao tempo total de espera de cada processo, e não mais ao tempo de execução. Esta opção é importante pois possibilita uma maior diferenciação entre os tempos de espera. Especialmente em casos onde o tempo de espera é pequeno em relação ao tempo total de execução, a representação da figura 3 é mais precisa.

As duas formas de visualização são na verdade complementares. Pode-se identificar, por exemplo, quais os processos que esperam por mais tempo na representação geral e a seguir, verificar os dados destes mesmos processadores em mais detalhes.

Tempos de espera inferiores a 0,1% do tempo de execução e do tempo de espera são excluídos das representações. Isto é feito por que tempos de espera tão pequenos em

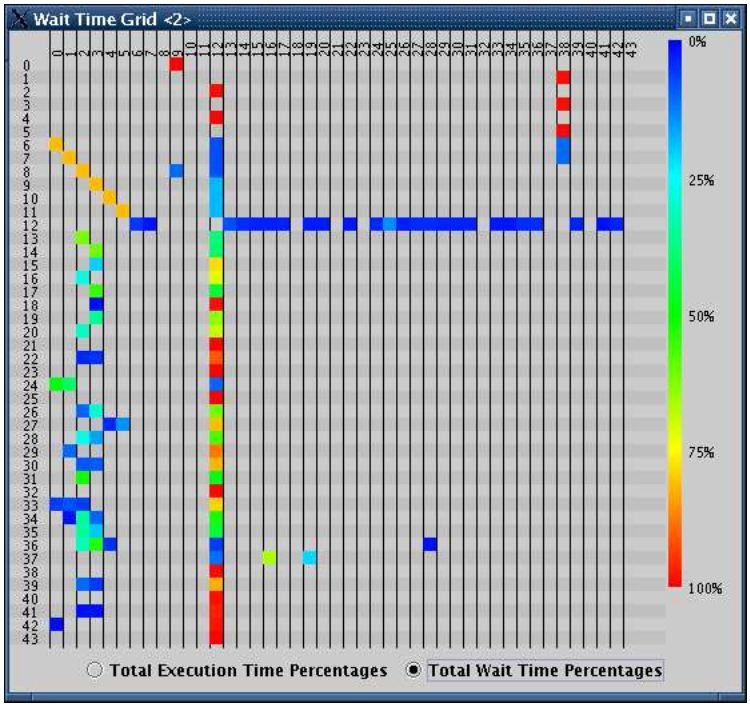

Figura 3: Tempo de espera por cada processador relativo ao tempo total de espera.

geral representam trocas rápidas de mensagens comuns em etapas de sincronização que tendem a preencher todo o gráfico com informação irrelevante. A apresentação destes dados dificultaria a visualização da informação realmente importante que são os casos em que o tempo de espera é significativo.

Pode-se obter, entre outros detalhes, o valor preciso de cada tempo de espera de um determinado processo clicando em qualquer célula da linha correspondente com o mouse. A seção 3.4 descreve a apresentação de detalhes dos processos.

Não foram encontradas nas ferramentas pesquisadas visualizações que demonstrem este nível de detalhamento do tempo de espera de processos.

3.3. Animação do Tráfego de Mensagens

Esta animação apresenta o tráfego de mensagens do sistema. Trata-se de uma tabela que apresenta todos os processos, conforme mostrado na figura 4. Uma área colorida representa a transmissão de uma mensagem no momento representado. As linhas representam os processos de origem das mensagens, ao passo que as colunas representam os processos que as recebem. As cores representam a velocidade de transmissão efetiva da mensagem.

Neste artigo, considera-se como velocidade de transmissão efetiva a quantidade de bits transmitida pelo tempo transcorrido entre o início do envio da mensagem pelo processo transmissor e o final do seu recebimento pelo destinatário. Desta forma, uma chamada a rotina de recebimento de mensagem tardia por parte de um processo é uma causa provável de uma transmissão efetiva lenta. Procura-se com esta medida levar em conta o tempo de resposta de processos que recebem mensagens, bem como o tamanho das mensagens em bits.

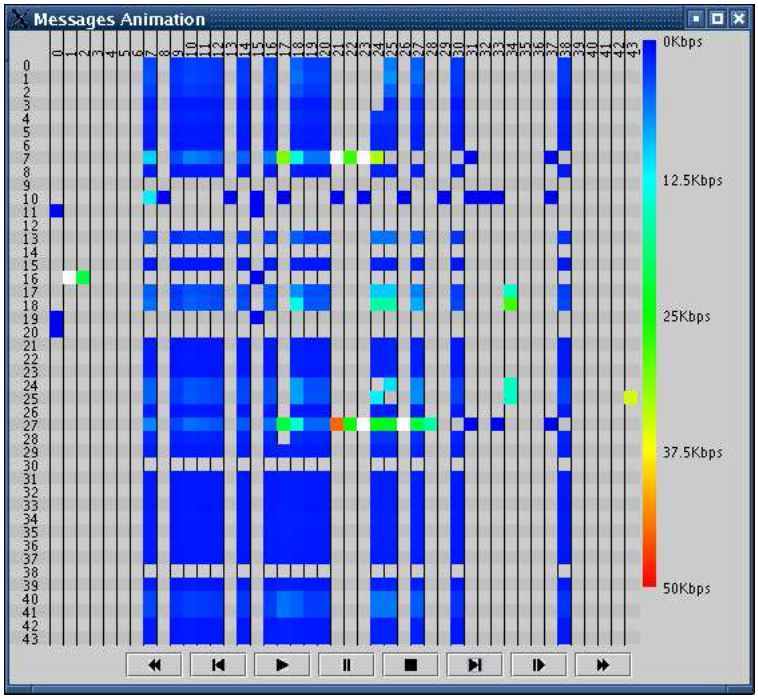

Figura 4: Tráfego de mensagens.

Um gradiente de cores facilita a visualização da eficiência da comunicação. Tanto um canal de comunicações ruim quanto o recebimento tardio de mensagens tendem a diminuir a eficiência da transmissão. Desta forma, ambos os problemas podem ser detectados através da ferramenta.

É disponibilizado ainda um gráfico semelhante contendo a taxa de transmissão efetiva média entre cada par de processos durante todo o tempo de execução, tornando possível a identificação de gargalos do sistema.

Freqüentemente a velocidade de transmissão efetiva dos dados apresenta uma variação grande. Isto acontece por uma série de motivos, como a existência de mensagens de diferentes tamanhos e diferenças na disponibilidade dos processos para receber mensagens. Manter um gradiente de cores que abranja todas as velocidades de transmissão diferentes muitas vezes não é o ideal. Por exemplo, uma aplicação poderia conter um subconjunto de mensagens bastante lentas e outro subconjunto extremamente rápido. Cada subconjunto, em um gradiente de cores grande, tenderia a se aproximar de uma das extremidades do gradiente, dificultando a identificação da variação da performance dentro de cada subconjunto. Para solucionar este tipo de problema, é possível ajustar os limites do gradiente para valores máximo e mínimo definidos pelo usuário. Neste caso, mensagens com velocidades superiores ao limite máximo do gradiente são mostradas em branco e mensagens com velocidades inferiores ao mínimo representável são mostradas em preto. Pode-se ainda fazer uma pausa na animação e ajustar o gradiente de cores de acordo com o quadro mostrado.

É possibilitado ao usuário ter acesso a mais detalhes sobre um processador específico através de um clique do mouse em qualquer uma das linhas que representam processadores, conforme apresentado na seção 3.4.

3.4. Estatísticas de Processos

Ao clicar na área correspondente a um determinado processo em qualquer uma das ferramentas anteriores, o usuário tem acesso a uma janela com alguns detalhes do respectivo processo. São representados o tempo consumido em cada estado em relação ao tempo total, bem como o tempo de espera por cada um dos demais processos em relação ao tempo total de espera.

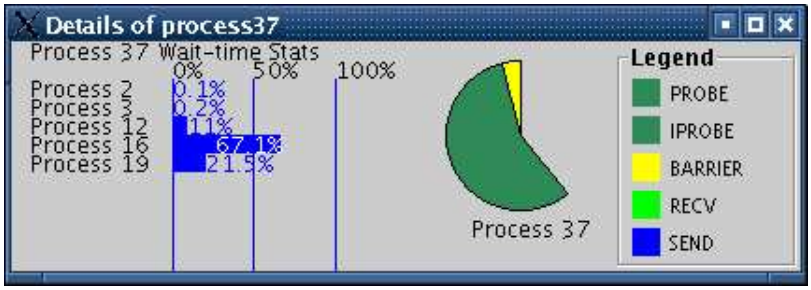

Figura 5: Janela apresentando detalhes de um processo específico.

A figura 5 mostra os detalhes de um determinado processo. O gráfico em barras a esquerda demonstra a porcentagem de tempo consumida esperando por cada processo. O processo representado na figura consome um tempo excessivo esperando por mensagens do processo 16. O gráfico em setores à direita da mesma figura representa a percentagem de tempo consumida por cada rotina MPI. Eventuais estados definidos pelo usuário através da MPE também são incluídos neste gráfico.

4. Aplicação

Foram utilizadas as ferramentas descritas neste artigo para avaliar a performance de uma aplicação. A aplicação estudada é um sistema de rastreamento de objetos utilizando câmeras de vídeo. Particularmente, os objetos rastreados são aqueles envolvidos em uma partida de futebol.

O sistema utiliza processamento paralelo devido a grande demanda computacional necessária para rastrear os objetos em tempo real. A aplicação chama-se SORTTS (*Soccer Real-Time Tracking System*) e foi desenvolvida no Instituto de Computação da UNICAMP.

A aplicação encontra-se dividida em três módulos: o módulo de leitura, o módulo de rastreamento e o módulo de iniciação e acompanhamento. O módulo de leitura é responsável pela leitura das imagens a partir das câmeras e por enviar estas imagens ou partes delas para outros processos. O módulo de rastreamento envolve processos que realizam rastreamento de objetos em imagens completas e em imagens reduzidas. O módulo de inicialização e acompanhamento é o responsável por iniciar e interromper os rastreadores de imagens reduzidas de acordo com as coordenadas obtidas através dos rastreadores de imagens completas.

Através da animação dos processos, foi possível identificar os diferentes grupos de processos devido ao seu comportamento diferenciado. O sistema funciona com os seis

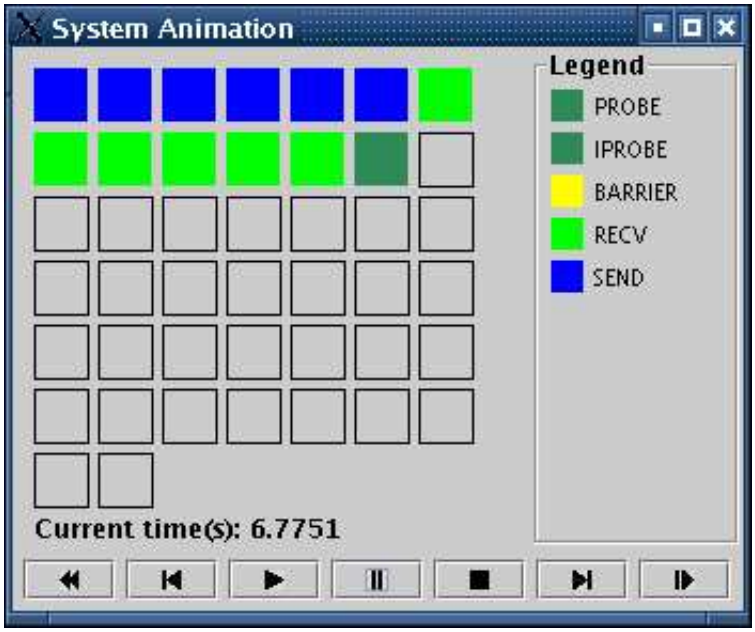

Figura 6: A ferramenta de animação de processos mostrando a execução do SORTTS.

primeiros processos no módulo de leitura, os seis seguintes no módulo de rastreamento de imagens completas e com o décimo terceiro processo no módulo de iniciação e acompanhamento. Os demais processos atuam no rastreamento de imagens reduzidas. A figura 6 demonstra um quadro da animação.

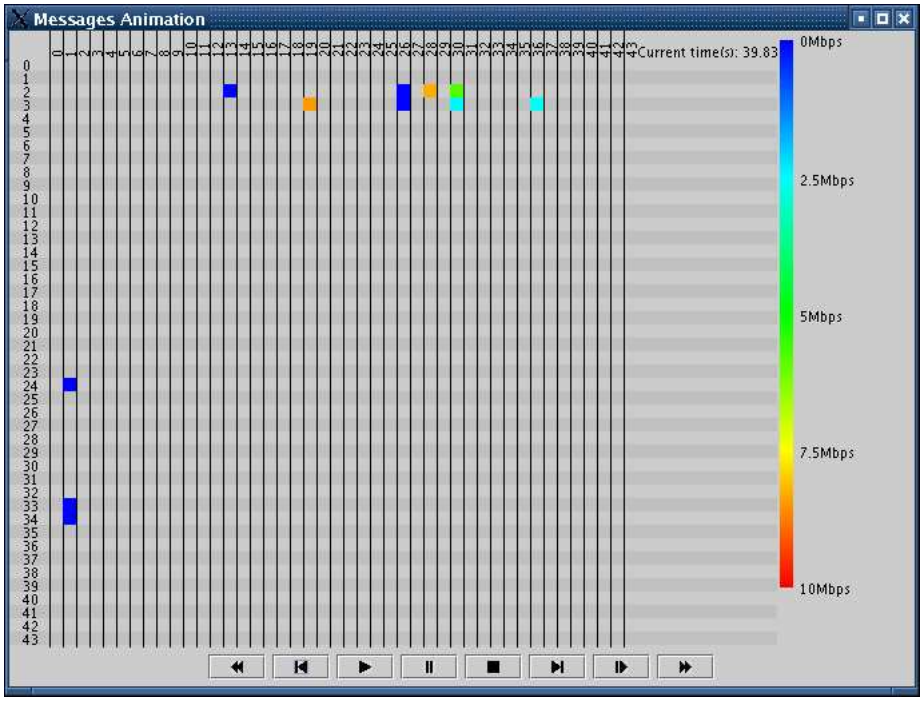

Figura 7: Tráfego de Mensagens da aplicação.

O maior tráfego de mensagens, conforme mostrado na figura 7, ficou entre os seis

primeiros processos e os processos de 13 a 43. Salvo algumas exceções, o tráfego observado ficou abaixo de 10Mbps, em uma rede com capacidade nominal de 100Mbps. Esta observação se confirma no gráfico que representa o tráfego agregado de toda a execução, mostrado na figura 8. A comunicação mais eficiente foi dos processos zero a cinco para os processos 6 a 11, pois estes enviavam imagens completas, formando mensagens grandes que efetivamente podem aproveitar o potencial da rede. A velocidade de comunicação efetiva nestes casos ficou em torno de 80Mbps.

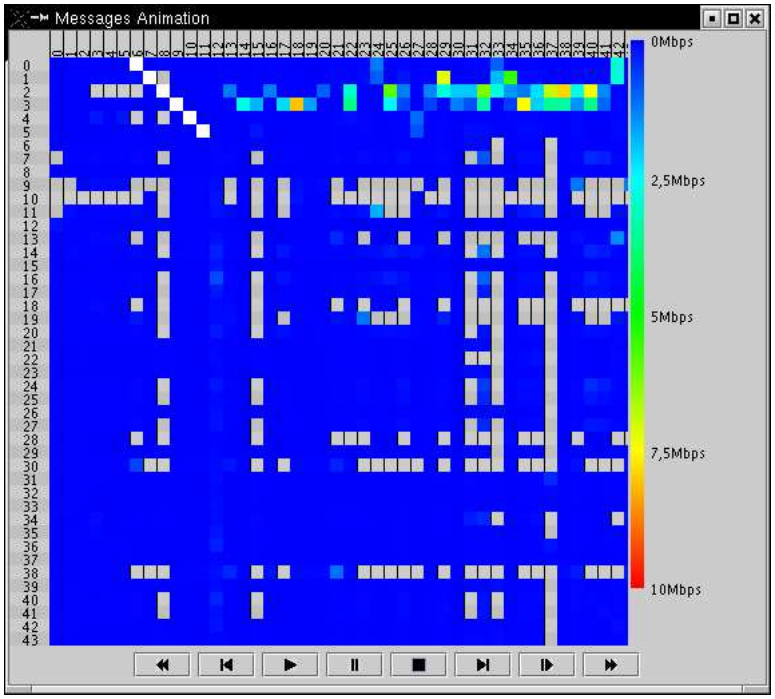

Figura 8: Tráfego de mensagens total da aplicação durante todo o tempo de execução.

O tempo de espera mostrado nas figuras 2 e 3 evidencia a centralização das atividades no processador 12 (módulo de inicialização e acompanhamento), sugerindo que possivelmente este processador esteja sobrecarregado. Novamente os processos 6 a 11 se destacam por esperar bastante pelos seis primeiros. Mas, como foi visto que a comunicação está eficiente, provavelmente o tempo de espera se dê apenas devido ao grande volume de dados transferido.

5. Conclusão

Pode-se concluir a partir da análise das ferramentas desenvolvidas que diferentes formas de visualização evidenciam diferentes características dos sistemas. As ferramentas propostas conseguem atingir seus objetivos, uma vez que foi possível identificar claramente algumas características de uma aplicação real.

Percebe-se também que a área de visualização para análise de desempenho de sistemas paralelos encontra-se em pleno desenvolvimento. Existem ainda inúmeras possibilidades a serem exploradas, algumas das quais pretende-se abordar em trabalhos futuros.

Referências

Calzarossa, M., Massari, L., Merlo, A., Tessera, D., and Vidal, M. (1998). A tool for performance analysis of parallel programs. *Rivista di Informatica*, 28(2):103–111.

Carothers, C., Topol, B., Fujimoto, R. M., Stasko, J., and Sunderman, V. (1997). Visualizing parallel simulations in network computing environments: A case study. In *Proceedings of the 1997 Winter Simulation Conference (WCS'97)*.

Chan, A. (2000). Slog-2 software development kit. `http://www-unix.mcs.anl.gov/perfvis/download/index.htm\#slog2sdk`.

Chan, A., Gropp, W., and Lusk, E. (1998). User's guide for mpe: Extensions for mpi programs. `http://www-unix.mcs.anl.gov/mpi/mpich/docs/mpeman/mpeman.htm`.

Etnus LLC. (2003). Etnus totalview. `http://www.etnus.com/Products/TotalView/index.html`.

Heath, M. and Etheridge, J. (1991). Visualizing the performance of parallel programs. *IEEE Software*, 8(5):29–39.

Message Passing Interface Forum (1994). Mpi: a message passing interface standard. *International Journal of Supercomputer Applications*, 8(special issue on MPI).

Müller Junior, B. and Anido, R. (2002). Object detection with multiple cameras. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, Orlando, FL, USA.

Nagel, W. E., Arnold, A., Weber, M., Hoppe, H., and Solchembach, K. (1996). Vampir: Visualization and analysis of mpi resources. *Supercomputer*, 12(1):69–80.

Reed, D. A., Aydt, R. A., Noe, R. J., Roth, P. C., Shields, K. A., Schwartz, B., and Tavera, L. F. (1993). Scalable performance analysis: The pablo performance analysis environment. In *Proceedings of the Scalable Parallel Libraries Conference*, pages 104–113. IEEE Computer Socirty.

Zaki, O., Lusk, E., Gropp, W., and Swider, D. (1999). Toward scalable performance visualization with Jumpshot. *High Performance Computing Applications*, 13(2):277–288.