Monster: uma Ferramenta para Monitoração de Clusters Voltada ao Escalonamento de Processos

Daniel Kikuti, Paulo Sérgio Lopes de Souza

Universidade Estadual de Ponta Grossa (UEPG)
Departamento de Informática (DeInfo)
Caixa Postal 15.064 – 91.501-970 – Ponta Grossa – PR – Brasil
danielkikuti@yahoo.com.br, pssouza@uepg.br

Abstract. *This paper describes the Monster, a clusters monitoring tool aiming the processes scheduling. This tool groups different metrics used in scheduling policies, allowing visualize and evaluate clusters performance. Monster provides online information, through its friendly graphical user interface, and after its execution, through its trace files. Experimental analysis of the tool demonstrates that it has a stable behavior, and it achieves its purposes, aiding in scheduling policies development and assisting in cluster's load balancing.*

Resumo. *Este artigo descreve o Monster, uma ferramenta de monitoração de clusters voltada ao escalonamento de processos. Esta ferramenta agrupa várias métricas usadas em políticas de escalonamento, permitindo visualizar e avaliar o desempenho de clusters. Monster fornece informações em tempo real, através de sua interface gráfica amigável, e posterior a sua execução, através dos arquivos de rastro. A análise da ferramenta por meio de experimentos demonstrou que a mesma, possui comportamento estável, e que alcançou seus objetivos, auxiliando no desenvolvimento de políticas de escalonamento e contribuindo com o balanceamento de carga no cluster.*

1. Introdução

O uso de grades computacionais e clusters na resolução de problemas que demandam alto poder de processamento vêm se tornando freqüente nos últimos anos [Souza et al., 1999]. Entre as vantagens freqüentemente citadas no uso de clusters destacam-se: a grande quantidade de recursos disponíveis, maior confiança resultante da multiplicidade destes recursos e a boa relação entre o custo para se montar um cluster e a possibilidade de redução do tempo de execução das aplicações. No entanto, para usar estes recursos eficientemente é necessário conhecer a carga de cada *host* (máquina) em tempo de execução. Para obter essas informações freqüentemente faz-se necessário o uso de ferramentas de monitoração.

A atividade de monitoração é importante porque permite ao usuário do sistema analisar a utilização dos recursos computacionais, encontrar gargalos de desempenho, identificar rotinas de programas que podem ser otimizadas e principalmente analisar o comportamento de suas aplicações frente a estimativas teóricas.

Este artigo descreve a especificação de um módulo do AMIGO[1] [Souza et al. 1999] com finalidade de monitoração de um sistema distribuído. O AMIGO é um ambiente para escalonamento de processos com o intuito de agrupar algoritmos de escalonamento de maneira dinâmica e flexível, relacionando estes algoritmos com as aplicações dos usuários.

Atualmente existem diversas ferramentas de monitoração no meio acadêmico e comercial [Johnson, 2003][Kikuti, 2003], porém os objetivos para qual foram propostas variam de acordo com os problemas que desejam identificar e tratar. A ferramenta de monitoração especificada neste artigo é voltada ao escalonamento de processos. Sua finalidade é apresentar alguns índices de carga que auxiliem no balanceamento dinâmico de carga, visto que uma forma de melhorar a performance em sistemas distribuídos é fazer com esta carga seja distribuída de forma balanceada.

Na seção seguinte serão apresentados alguns aspectos teóricos relacionados à monitoração e escalonamento de processos. A seção 3 descreve a modelagem da ferramenta utilizando statecharts. Na seção 4 são apresentadas algumas métricas disponíveis na ferramenta. A seção 5 apresenta alguns resultados obtidos com a ferramenta. Por fim são apresentadas algumas considerações finais e trabalhos futuros na seção 6.

2. Definindo a Ferramenta de Monitoração

Como já mencionado, o Monster (MONitor de cluSTER) é um módulo do AMIGO e possui um objetivo específico: oferecer informações sobre a utilização de recursos computacionais através da monitoração. Espera-se com isso possibilitar uma distribuição adequada da carga de trabalho sobre a plataforma distribuída.

A avaliação de desempenho em um sistema usando a técnica de monitoração é feita através de monitores. Um monitor é uma ferramenta com a finalidade de coletar dados mostrando-os como resultado. Os monitores geralmente são classificados quanto ao nível de implementação – podendo ser por software, hardware, firmware ou híbridos; quanto ao mecanismo de disparo – orientado a eventos (o monitor capta informações quando ocorrem mudanças no estado do sistema) ou tempo (coleta de informações de forma periódica); e quanto à forma que os resultados são apresentados – em tempo real (on-line) ou posterior a execução (batch) [Jain 1991].

A ferramenta de monitoração apresentada neste artigo se enquadra nas classificações acima como sendo um monitor para sistemas distribuídos por software, com mecanismo de coleta de dados ativado por tempo, e com capacidade de apresentação dos mesmos tanto em tempo real quanto posterior.

O objetivo desta ferramenta é monitorar a utilização de recursos computacionais nos diversos *hosts* que compõem o *cluster*. A coleta de informações será ao nível do sistema operacional e sendo esta uma ferramenta de monitoração por software, haverá disputa de recursos com outros processos também em execução. Esta intrusão não pode influenciar significativamente no tempo despendido pelas demais aplicações do *cluster*. Caso isto ocorra, os resultados obtidos não refletirão a carga real do sistema e conseqüentemente perde-se o foco dos objetivos propostos.

[1] AMIGO é um acrônimo para dynAMical flexIble schedulinG envirOnment, um ambiente dinâmico e flexível para escalonamento de processos.

A opção pelo desenvolvimento de um monitor por tempo deve-se à quantidade de métricas e informações coletadas. Se a freqüência de ocorrência dos eventos que estão sendo monitorados é alta, então o uso de um monitor orientado a eventos gerará uma intrusão significativa no sistema, por outro lado, um monitor por tempo passa a responsabilidade de controle de overhead para o usuário.

Quanto à apresentação dos dados, é interessante poder observar qual o estado do *cluster* em tempo de execução das aplicações e ter condições de analisar os resultados após a execução das mesmas, podendo aplicar métodos estatísticos nos arquivos de rastro gerados para organizar os dados e fazer previsões futuras. Para que isto seja possível, o arquivo de rastro deve estar em um padrão reconhecido. Neste caso adotamos o padrão XML (*Extensible Markup Language*) por se tratar de um arquivo texto estruturado e por apresentar características de portabilidade.

O escalonamento de processos pode ser visto como uma forma de distribuir processos de forma organizada e eficiente de acordo com os recursos computacionais disponíveis. O escalonamento pode ser estático, quando definido explicitamente no código fonte ou durante a compilação do programa a ser escalonado, ou dinâmico quando modificado a qualquer momento enquanto estiver executando. O AMIGO proporciona este ambiente dinâmico e suas políticas de escalonamento necessitam de um módulo capaz de informar a utilização de recursos do sistema para auxiliar em suas decisões.

Monitorar um sistema distribuído com várias máquinas é uma tarefa complexa se comparada à monitoração de apenas uma máquina. O monitor de um sistema distribuído deve ser constituído de vários componentes que trabalham separados e concorrentemente. Essa abordagem é detalhada na seção a seguir onde é mostrada a modelagem da ferramenta usando statecharts.

3. A Ferramenta Monster

O Monster é composto de duas camadas, uma responsável pela coleta de dados e outra responsável por apresentar estes dados ao usuário. Um dos pontos fundamentais na modelagem do monitor é a representação da interação entre a camada superior, responsável pela visualização dos dados (Front End), e a camada inferior, que coletará os dados que irão compor as diferentes métricas para avaliação de desempenho (Back End).

Para representar os aspectos comportamentais da ferramenta, bem como troca de mensagens e concorrência, foram utilizados statecharts. A Figura 1 apresenta uma visão geral da ferramenta, onde existem dois processos (front end e back end) e uma linha tracejada representando concorrência entre estes dois processos. Note que o back end também está subdividido por linhas tracejadas, isto significa que ele é composto por diversos processos[2] que estão em algum estado em determinado período de tempo.

Esta modularidade presente na modelagem de fato ocorre na prática. A camada back end é um módulo composto de diversos programas independentes entre eles com finalidade de coleta de dados, gravação e envio dos mesmos (caso solicitado), sendo

[2] Sem perda de generalidade, a Figura 1 ilustra apenas três processos. Emum sistema real, este modelo deve ser interpretado como N processos, onde N representa o número de máquinas que forma o *cluster*.

mais detalhado na seção 3.1. A camada front end nada mais é que um visualizador e gerenciador da camada inferior e será abordada na seção 3.2.

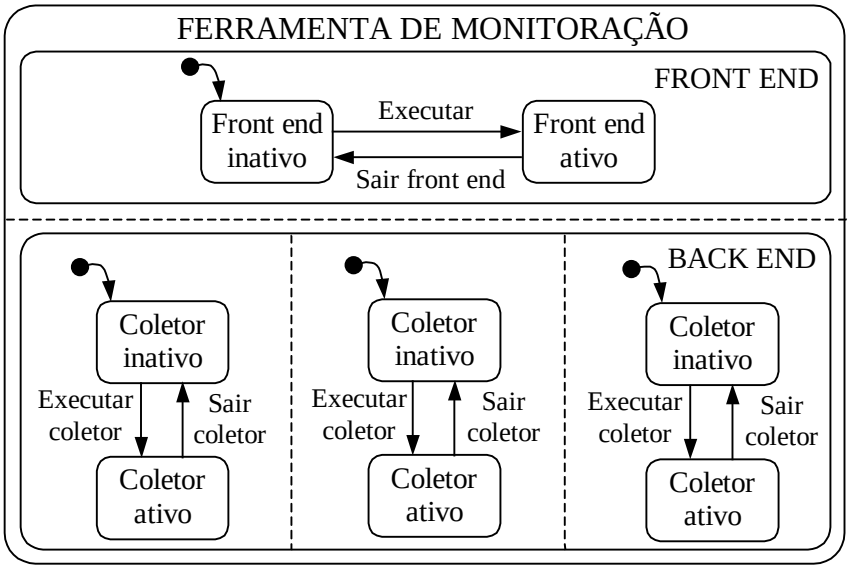

Figura 1. Visão geral da ferramenta

3.1. Back End

O diagrama da Figura 2 é uma explosão do estado coletor ativo apresentado na Figura 1. Inicialmente o processo coletor está no estado inativo, ou seja, o programa encontra-se no disco e ainda não entrou em execução. Quando o usuário ou o Front End coloca o processo em execução, ele sai do estado inativo, passa para o estado ativo.

A primeira tarefa a ser executada quando o coletor entra no estado ativo é verificar os argumentos de entrada. Se o coletor foi disparado por um usuário através de um terminal (console), ele simplesmente passa para a etapa seguinte, caso contrário (Front end no estado ativo) ele envia uma mensagem confirmando início de execução.

A próxima etapa consiste em verificar a configuração do sistema. Esta configuração define o intervalo de tempo em que serão feitas as coletas dos dados, quais dados serão enviados e para quem deverá ser enviado.

Os três próximos estados representam o loop principal do coletor. No estado “coletando dados” serão feitas chamadas ao sistema operacional buscando todos os dados referentes às métricas disponíveis na ferramenta. Após a coleta, é verificado se alguém solicitou envio dos dados (no caso de monitoração online), e caso alguém tenha feito, então a mensagem é enviada. O coletor irá permanecer no próximo estado, dormindo, até que alguma mensagem chegue ou quando interrompido pelo intervalo de tempo definido para coleta.

Este loop irá ser executado até que chegue alguma mensagem. Esta mensagem pode ser alguma mensagem de configuração ou de término de execução. Caso seja uma mensagem de configuração, o coletor irá aplicar a mesma e entrará novamente no loop principal. Caso a mensagem seja para finalizar execução, então o coletor enviará mensagem confirmando término e passará ao estado inativo.

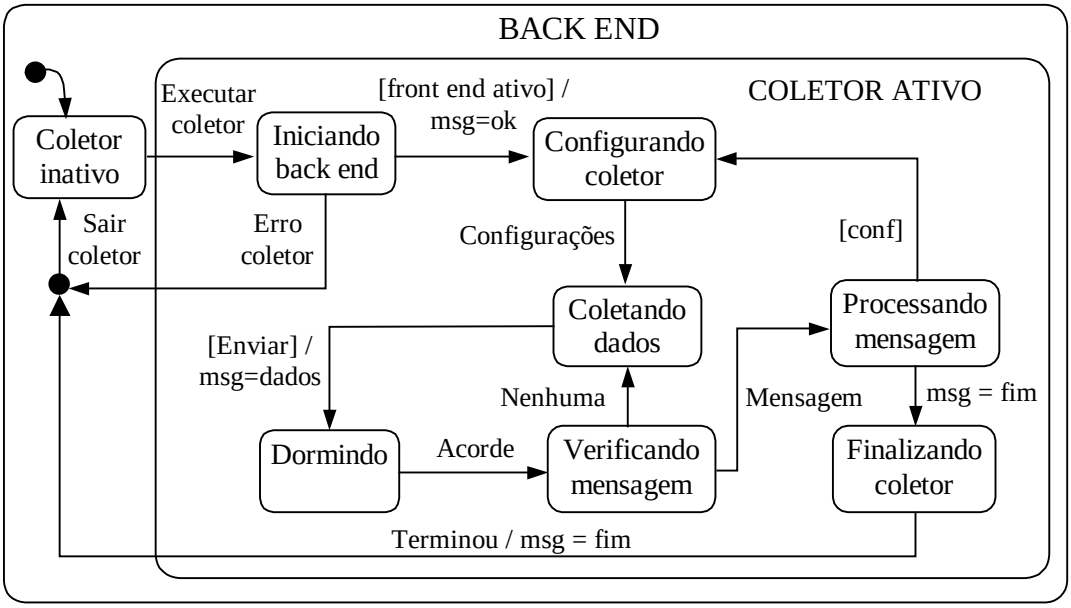

Figura 2. Camada inferior

3.2. Front End

A Figura 3 apresenta os possíveis estados em que o front end pode estar. Inicialmente ele se encontra no estado inativo, e quando colocado para executar passa para o estado "inicia front end". Neste estado são carregados os componentes visuais, verificadas as configurações, e criados os comandos conhecidos pela interface. A interface foi desenvolvida em C utilizando as bibliotecas do Tcl/Tk, por serem softwares gratuitos, portáteis, de fácil desenvolvimento e por apresentarem bom desempenho [Kikuti et al. 2002][Kikuti et al. 2003].

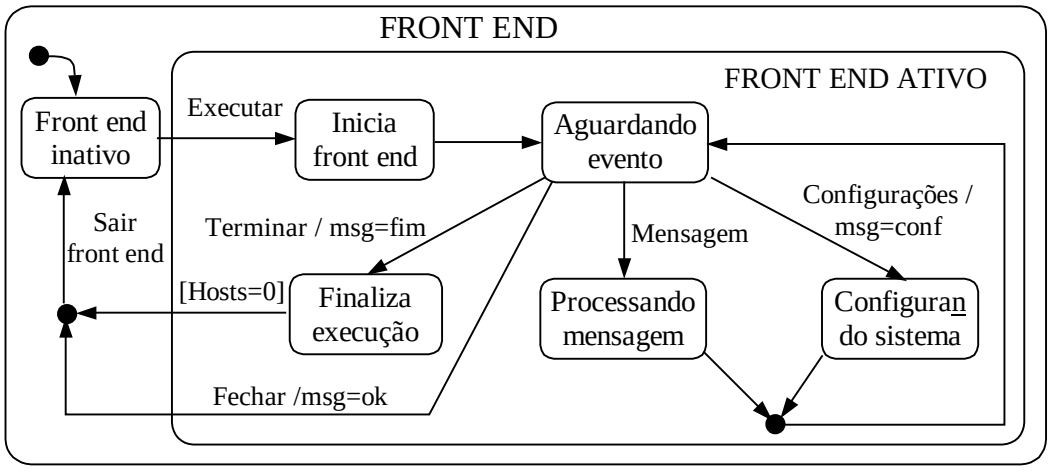

Figura 3. Camada superior

Passado este estado inicial, o front end entra no loop principal gerenciado pelo Tk, permanecendo neste estado até que algum evento reconhecido pelo interpretador seja ativado. Os estados reconhecidos pelo gerenciador de eventos podem ser solicitações do usuário ou mensagens recebidas da camada inferior.

As ações que o usuário pode efetuar são consideradas como “configurações”. Solicitação de visualização de uma métrica, alteração no tempo de coleta de carga, finalização de execução e adição ou remoção de coletor, são alguns exemplos de configuração que exigem comunicação entre o front end e o back end.

Mensagens da camada inferior podem ser classificadas de duas formas: dados ou confirmações. Dados são transformados em gráficos e apresentados ao usuário e confirmações constituem um processo para aumentar confiabilidade da comunicação entre as duas camadas.

O estado “finaliza execução” só é alcançado quando o usuário solicita terminar a execução da ferramenta (implicando na finalização dos coletores também) e, após todos os coletores confirmarem com mensagem de finalização ou que ocorreu timeout, a ferramenta passa para o estado inativo. Outra forma de colocar o front end em estado inativo é simplesmente sair. Neste caso os coletores serão notificados e permanecerão coletando carga do *cluster*.

A camada inferior foi desenvolvida em C e a comunicação entre as duas camadas utilizou a interface sockets, esta acessando a pilha TCP/IP. Para a comunicação entre os módulos optou-se pelo protocolo UDP, por questões de capacidade escalar da ferramenta.

Devido à sua organmização e construção, pode-se afirmar que a ferramenta é independente das aplicações em execução no *cluster*, assim como também é independente de qualquer ambiente de passagem de mensagem, tais como PVM (*Parallel Virtual Machine*) ou MPI (*Message Passing Interface*).

4. Métricas para Análise de Desempenho

As métricas para análise de desempenho disponíveis na ferramenta estão distribuídas em quatro categorias: poder de processamento, memória, disco e rede. Estas informações são coletadas diretamente do sistema operacional através do sistema de arquivos */proc* e refletem a utilização de recursos da máquina local.

4.1 Métricas Sobre Poder de Processamento

Informações referentes ao poder de processamento incluem o percentual de utilização do processador, troca de contexto, número de processos em execução e interrupções.

O percentual de utilização de processador (cpu) fornece informações sobre a distribuição do tempo de cpu entre: tempo de sistema, tempo do usuário e tempo ocioso. Esta informação permite verificar diretamente a carga de processamento incidente em cada máquina, sendo desejável minimizar o tempo ocioso da cpu, aumentar o tempo disponível para processamento de aplicações em nível do usuário e reduzir o tempo gasto no processamento em nível de sistema, otimizando assim o tempo necessário à execução das aplicações.

Uma troca de contexto ocorre quando um processador pára de rodar um processo e começa a executar outro. Como cada contexto exige que o sistema operacional tenha controle da cpu, excessivas trocas de contexto aumentam o consumo de tempo de cpu em nível de sistema. Então a monitoração do número de trocas de contexto pode ser uma métrica eficiente na distribuição de carga.

Informação sobre o número de processos pode ser outro indicativo da utilização do poder de processamento de uma máquina. Uma métrica interessante é a análise de quantos processos estão aptos a rodar em um determinado instante. Um processo apto a rodar é aquele que pode fazer seu trabalho assim que for escalonado a receber tempo de cpu [Tanenbaum, 1995]. Então quanto mais processos aptos a rodar, maior será a demanda no *host* por processamento.

As interrupções também geram degradação de desempenho devido ao consumo de cpu em nível de sistema. Um grande número de interrupções pode indicar uma alta atividade no *host* principalmente relacionadas com o sistema operacional e/ou I/O.

4.2 Métricas Sobre Memória

Através da análise do poder de processamento do *cluster* podem ser elaboradas muitas políticas de escalonamento, porém muitas vezes só estas informações não bastam, pois uma política de escalonamento varia de acordo com o propósito para qual foi desenvolvida. O balanceamento de carga não está necessariamente vinculado ao uso de processador e um exemplo é a distribuição de carga de acordo com utilização de memória [Souza et al. 1999]. Para atender as políticas de escalonamento que necessitem de informações sobre memória ou os analistas que gerenciam sua utilização, a ferramenta disponibiliza métricas sobre utilização, paginação, páginas ativas e swap.

Métricas relacionadas à utilização de memória apresentam dados estatísticos sobre a quantidade de memória usada, quantidade de memória livre, compartilhada, usada em buffers e em cache. Esta é uma forma simples e direta de verificar como a memória disponível está sendo utilizada.

A paginação, quantidade de blocos paginados do disco para a memória (page ins) e quantidade de blocos paginados da memória para o disco (page outs), fornece estatísticas que permitem observar quão ativo está o uso de memória virtual. O uso constante de memória virtual provoca a degradação de performance e pode ser um indício de pouca memória física para as aplicações em execução no *cluster*.

Informações mais refinadas sobre o uso de memória podem ser obtidas através da análise das páginas ativas, inativas limpas e inativas sujas. Estas estatísticas mostram como as páginas residentes em memória estão sendo usadas. Páginas ativas indicam a quantidade de memória que está sendo utilizada de alguma forma, páginas inativas sujas informam a quantidade de memória que foi modificada e precisa ser escrita no disco e, páginas limpas, ao contrário, informam a quantidade de memória que não foi modificada e portanto não precisam ser escritas no disco.

Quando a memória é insuficiente para armazenar todos os processos, o excedente é mantido em disco. Para rodar os processos excedentes, estes devem ser trazidos para a memória. Esta movimentação de processos entre disco e a memória (e vice-versa) é denominado *swapping* [Tanenbaum, 1995]. Assim como na paginação, o swap também degrada o desempenho por fazer acesso a um dispositivo lento (disco). As métricas sobre swap disponíveis na ferramenta fornecem informações da quantidade de processos trazidos do espaço de swap para a memória (swap in), quantidade de processos enviados da memória para o espaço de swap (swap out), quantidade de espaço livre e utilização do espaço de swap.

4.3 Métricas Sobre Armazenamento

A monitoração de dispositivos de armazenamento deve ser efetuada porque estes dispositivos apresentam grande diferença de velocidade (devido às limitações elétricas e mecânicas) quando comparados à memória primária e, acessos excessivos podem resultar em perdas de desempenho. As métricas cobertas pela ferramenta para monitoração de dispositivos de armazenamento são: transferência, leituras e escritas, fila de pedidos e algumas médias de tempo. O cálculo de transferência por segundo é feito através da soma de todas as escritas e leituras efetuadas no disco dividido pelo tempo total de cpu. É uma métrica que mostra como está o tráfego no barramento.

Para obter informações sobre a quantidade de leitura e escrita efetuada, a ferramenta disponibiliza o número de leituras/escritas e a quantidade em KBytes lidos e escritos.

A análise da fila dos pedidos de I/O atendidos informa a quantidade de escrita/leitura atendidas por segundo.

Outras métricas interessantes para análise do uso dos dispositivos de armazenamento disponíveis na ferramenta são as médias. São dispostas três médias para o usuário: tempo médio de espera (em mili-segundos) por pedido de I/O, tempo médio (em mili-segundos) efetuando I/O e média percentual de CPU usado para atender pedido. Estas três médias são interessantes porque possuem relação simples com o índice de performance, isto é, pode-se avaliar a carga do dispositivo olhando para estas médias e ter uma estimativa de qual o tempo necessário para efetuar uma operação de I/O.

4.4 Métricas Sobre Rede

As métricas para monitoração da carga da rede referem-se ao dispositivo de rede local, estando limitado aos dados sobre bytes enviados/recebidos e estatísticas de erros e colisões. Embora a ferramenta ainda não forneça informações sobre rede de forma mais abrangente, estes dados podem servir para um administrador no diagnóstico de problemas em determinado *host*.

Embora neste artigo sejam apresentadas as diferentes métricas disponíveis juntamente com seu significado, não é função da ferramenta tomar decisões baseado nestes dados. Esta atividade fica a cargo das políticas de escalonamento ou do administrador do sistema.

5. Resultados Obtidos

A Figura 4 mostra a interface gráfica disponível ao usuário. Há um menu principal, onde o usuário pode acessar todas as funcionalidades da ferramenta, e um diagrama representando todas as máquinas que compõem o cluster (cada retângulo contém um nome e representa uma máquina).

No menu File o usuário pode visualizar os arquivos de rastro gerados (trabalho futuro), sair ou terminar a aplicação. No menu View estão dispostas todas as métricas descritas na seção anterior. No menu edit pode-se configurar o tempo para coleta de informações e adicionar ou remover coletores que compõem o back end. E o menu Help contém informações sobre utilização da ferramenta e significado de cada métrica disponível.

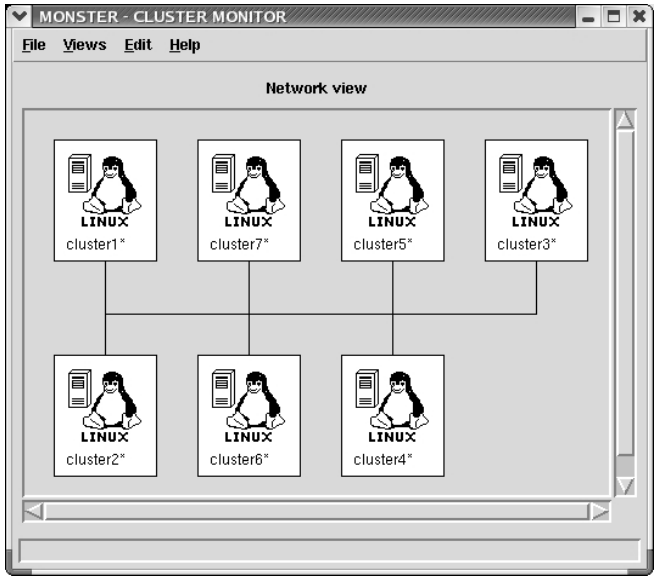

Figura 4. Interface Gráfica

Todas métricas disponíveis no Monster atualmente são representadas através de gráficos de linha. A Figura 5 mostra como exemplo a métrica sobre percentual de utilização em modo usuário. Cada linha possui uma cor e representa uma máquina. O gráfico apresenta uma barra horizontal, indicando o valor (neste caso percentual), e uma barra vertical, representando o tempo naquela posição.

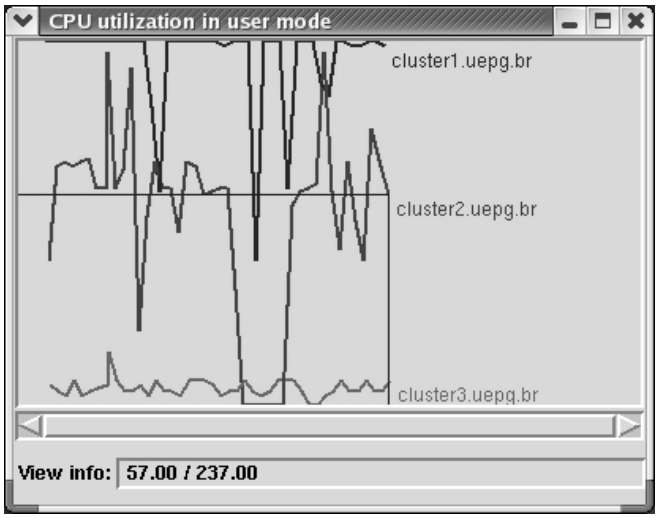

Figura 5. Exemplo gráfico (percentual de utilização de cpu)

Neste exemplo estão sendo monitoradas três máquinas. Pode-se perceber através deste gráfico que as cargas não estão corretamente balanceadas, estando o cluster 1 quase sempre com 100% de utilização em modo usuário, o cluster 2 com um nível intermediário de carga e o cluster 3 sempre ocioso.

6. Considerações Finais e Trabalhos Futuros

Este artigo apresentou o Monster, uma ferramenta de monitoração voltada ao escalonamento de processos. Este trabalho apresenta resultados satisfatórios em termos de avaliação de desempenho e auxílio no desenvolvimento de políticas de escalonamento de processos através da avaliação de utilização de recursos.

Como trabalhos futuros, deseja-se implementar um módulo analisador de arquivos de rastro capaz fazer previsões da carga futura em um curto período de tempo e desenvolver políticas que utilizem estes arquivos gerados pela ferramenta.

Referências Bibliográficas

Jain R., "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", Wiley-Interscience, New York, NY, April 1991, Chapter 7, pages 93-109.

Kikuti, D., Souza, P. S. L. and Souza, S. R. S. "XPVM-W95 - A Performance Monitoring Tool for PVM Clusters on Windows Operating Systems", XXII International Conference of the Chilean Computer Science Society - VI Workshop on Distributed Systems and Parallelism, v. 1 Copiapo - Atacama, Chile, 2002.

Kikuti, D., Souza, P. S. L. and Souza, S. R. S. "Using Portable Monitoring for Heterogeneous Clusters on Windows and Linux Operating Systems", Journal of Computer Sciense and Tecnology, 2003.

Souza, P. S.; Santana, M. J. and Santana, R. H. C**.** "AMIGO – A Dynamical Flexible Scheduling Environment", Proceedings of the 5th International Conference on Information Systems, Analysis and Synthesis: ISAS'99, junho, 1999**.**

Tanembaum, A. S. "Sistemas Operacionais Modernos", Ed. Prentice-Hall do Brasil Ltda. Rio de Janeiro, 1995.

Johnson M. K. TOP(1) – "Linux User's Manual" (2003), http://die.net/doc/linux/man/man1/top.1.html.