Panalyser, Uma Ferramenta de Baixo Impacto para Medição de Utilização de Recursos do Sistema Operacional Linux

Martin A Kretschek[1] **, Roberto A Hexsel**[1] **, Aldri Luiz dos Santos**[1]

[1]Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19081 – 81531-990 Curitiba, PR

martin.alain@terra.com.br, roberto@inf.ufpr.br, aldri@inf.ufpr.br

Abstract. *This paper describes the implementation of a tool for measuring resource usage by processes in a GNU/Linux environment. The tool is called panalyser and it provides support (1) for application development and performance evaluation, (2) diagnosis of system bottlenecks, (3) workload characterization, (4) finding model parameters, and (5) to validate system models. Panalyser was designed to produce very little interference in the system being investigated. It operates in batch mode, and is an event and sample driven monitor. This paper also presents measurements taken on a heavily loaded Apache server.*

Resumo. *Este trabalho apresenta uma ferramenta de baixo impacto para medição de utilização de recursos do Sistema Operacional GNU/Linux pelos processos. A ferramenta é chamada panalyser e provê informações para (1) o desenvolvimento e avaliação de desempenho de aplicações, (2) diagnóstico de pontos de contenção no sistema, (3) caracterização de carga, (4) levantamento de parâmetros para modelos e (5) validação de modelos. A implementação do panalyser torna sua execução leve, minimizando distorções no resultado das medições. O panalyser é um monitor que opera em batelada e é controlado por eventos e amostragem. Este artigo descreve também medidas efetuadas em um servidor Apache sobrecarregado.*

1. Introdução

O papel de um Sistema Operacional (SO) é mediar e multiplexar o acesso de múltiplos processos aos recursos providos pelo *hardware*. Ferramentas para medição de utilização de recursos do SO são usadas para fornecer dados sobre aspectos específicos do desempenho de um sistema de computadores. Estes dados, por sua vez, necessitam ser relacionados e comparados, dentro de determinados critérios, para obter uma visão ao mesmo tempo geral e precisa do desempenho do sistema avaliado.

Os dados de utilização de recursos do SO em um sistema são de grande relevância para: (1) projeto e desenvolvimento de aplicações e sua avaliação de desempenho; (2) diagnóstico de pontos de contenção; (3) ajuste e melhoramento do desempenho; (4) caracterização de uma carga de trabalho; e (5) levantamento de parâmetros, sua validação, e de dados para modelos. Estes aspectos são critérios chave para o projeto, aquisição e uso de sistemas de computadores [Jain, 1991].

As ferramentas para medição de utilização de recursos do SO que são distribuídas correntemente têm a capacidade de medir a utilização de uma vasta gama de recursos, incluindo tempo de CPU, a memória (RAM e discos), e os dispositivos da rede. No entanto, estas ferramentas normalmente fornecem dados sobre o sistema como um todo, sem discriminar os dados de utilização de recursos para cada programa em execução. Outro problema destas ferramentas é seu alto custo de execução, que pode interferir significativamente nas medições efetuadas. Além disto, o relacionamento e comparação dos dados fornecidos por elas podem não ser triviais. Existem também ferramentas desenvolvidas para avaliação de programas específicos, como por exemplo o *WebMonitor* [de Almeida, 1997]. Estas ferramentas, apesar de apresentarem boa precisão nas medidas efetuadas, são de difícil portabilidade por exigirem modificações nos programas a serem medidos e até mesmo no SO.

Com a finalidade de resolver os problemas mencionados, foi desenvolvida a ferramenta *panalyser* para medição da utilização de recursos pelos processos no ambiente do Sistema Operacional GNU/Linux. O *panalyser* fornece dados sobre a utilização de CPU, memória primária e secundária, e classificação e totalização das chamadas de sistema para um dado processo. O *panalyser* impõe pouca sobrecarga no sistema, minimizando distorções no resultado das medições. As informações fornecidas permitem uma melhor comparação dos dados medidos devido a técnica de amostragem de dados empregada. Outro aspecto importante do *panalyser* é a possibilidade da medição de qualquer processo em execução, inclusive processos do SO.

Na implementação do *panalyser* foram consideradas a implementação e a demanda de recursos do SO das ferramentas distribuídas com os SOs atuais. De modo a não repetir as mesmas implementações e modos de operação daquelas, apresentando assim os mesmos custos de execução e as conseqüentes distorções nas medidas produzidas, foram usadas chamadas de sistema padrão do Linux. Isto proporciona portabilidade à ferramenta, além de não ser necessário modificar o SO. A implementação do *panalyser* também não necessita que sejam feitas modificações nos programas a serem medidos. A medição de processos e de seus processos filhos é outro ponto de destaque do *panalyser*, que permite a clara discriminação da utilização dos recursos do SO entre vários processos.

A seção 2. contém uma breve descrição das técnicas de avaliação de desempenho, bem como as ferramentas de avaliação de desempenho de maior destaque distribuídas com os SOs UNIX System V e Linux. A seção 3. descreve a ferramenta *panalyser*, seus parâmetros de configuração e sua implementação para atingir os desafios acima propostos. A seção 4. apresenta um exemplo do uso do *panalyser* para a medição dos processos de um servidor Web Apache. O objetivo do experimento é demonstrar a capacidade do *panalyser* em fornecer dados de utilização de recursos do SO pelo Apache, e assim mostrar porque o mesmo atingiu um estado saturado. A seção 5. apresenta as conclusões sobre o trabalho.

2. Técnicas de Avaliação de Desempenho

As três principais técnicas para avaliação de desempenho são *modelagem analítica*, *simulação* e *medição* [Jain, 1991]. A medição de sistemas de computadores envolve a monitoração dos mesmos, enquanto submetidos a cargas de trabalho específicas. Para a obtenção de resultados razoáveis, as cargas de trabalho devem ser cuidadosamente escolhidas. A medição somente é possível com a existência de um sistema real a ser medido, bem como de ferramentas para realizar as medições.

Um monitor é uma ferramenta usada para observar a atividade de um sistema. Em geral, monitores observam o desempenho dos sistemas, coletam estatísticas sobre o desempenho, analisam os dados e mostram resultados. Alguns monitores também identificam áreas problemáticas e sugerem correções. Monitores são usados não somente por analistas de desempenho, mas também por programadores e gerentes de sistema para: (1) encontrar segmentos do programa freqüentemente usados e otimizar seu desempenho; (2) medir alocação de recursos e encontrar pontos de contenção no desempenho; (3) fazer o ajuste do sistema e melhorar o desempenho; (4) caracterizar uma carga de trabalho, para planejamento de capacidade e criação de cargas de trabalho de testes; e (5) encontrar os parâmetros de um modelo, validar modelos, e desenvolver dados de entrada para modelos [Jain, 1991].

2.1. Ferramentas de Avaliação de Desempenho

Dentre as ferramentas de avaliação de desempenho distribuídas com os SOs correntes destacam-se as ferramentas `strace`, `atsar`, `ps`, `top`, presentes nos UNIXes comerciais baseados no padrão System V, e no Linux, tais ferramentas são descritas abaixo, assim como a ferramenta *WebMonitor*, usada para avaliação de desempenho do programa servidor Apache no Linux.

Strace A ferramenta `strace` intercepta e registra as chamadas de sistema efetuadas pelo processo rastreado, bem como os sinais que são recebidos pelo processo. O nome de cada chamada de sistema e seus argumentos são impressos na saída de erro padrão ou em um arquivo. É possível gerar um sumário ao término da execução do programa rastreado, ou na interrupção do rastreamento, contendo o tempo decorrido, a quantidade e os erros para cada uma das classes de chamadas de sistema efetuadas [Akkerman, 1999].

Ferramentas Baseadas no /proc As ferramentas `atsar` [Langeveld, 2001], `free`, `ps` e `top`, são baseadas no pseudo sistema de arquivos `/proc` [Torvalds, 1996]. O `/proc` é usado como uma interface para as estruturas de dados do *kernel*, e para leitura e interpretação de `/dev/kmem`, que é o arquivo associado ao dispositivo que representa a memória virtual do *kernel* do Linux. É importante ressaltar que embora o `/proc` ofereça uma ampla variedade de informações, a sua utilização freqüente causa um alto impacto no SO. A ineficiência da utilização do `/proc` é causada pela necessidade de realizar várias leituras no sistema de arquivos ou ler grandes quantidades de dados. O alto custo de acesso ao sistema de arquivos se deve ao fato do processo em questão interromper sua execução e aguardar uma resposta indicando se a leitura obteve sucesso ou não. Enquanto o processo espera, o escalonador pode escolher um outro processo para executar. Se a freqüência com que o processo em questão efetua chamadas, ou se o tamanho da

requisição no *read* for grande, isto acarretará desperdício do tempo de CPU em atividade de E/S.

WebMonitor O *WebMonitor* [de Almeida, 1997] é uma ferramenta que emprega uma combinação de técnicas de monitoração por amostragem e técnicas de monitoração orientada por eventos para coletar diferentes níveis de informação sobre a operação de um servidor Web `Apache` versão 1.1.1 [Robinson and the Apache Group, 1995].

A coleta baseada em amostragem é usada para ler os valores de contadores mantidos pelo *kernel*. Foram necessárias modificações no *kernel* do Linux para leitura destes contadores. Para o monitoramento das requisições HTTP emprega-se a técnica de monitoração orientada por eventos. Para isto foi necessária a instrumentação do código do programa servidor Apache. As instrumentações do *kernel* e do Apache, usadas para viabilizar a implementação *WebMonitor* permitem uma grande precisão nas medições.

3. A Ferramenta Panalyser

As informações sobre a utilização de recursos do SO, discriminadas por processo, permitem quantificação precisa da utilização de recursos do SO pelos processos analisados. Estas informações permitem ampliar a gama de fatores para determinação da melhor configuração, otimização e ampliação de um sistema para o atendimento dos serviços para os quais foi concebido.

O *panalyser* é um monitor que opera em modo batelada, sendo controlado por eventos e amostragem, e que permite medir a utilização de recursos pelos processos no SO Linux. As informações apresentadas possibilitam desenvolver e avaliar o desempenho de aplicações, diagnosticar pontos de contenção no sistema, caracterizar cargas de trabalho, fornecer parâmetros e validar modelos.

3.1. Informações Fornecidas pelo Panalyser

As informações de utilização de recursos do SO, fornecidas pelo *panalyser*, são baseadas na estrutura de dados `rusage`, a qual é preenchida pela chamada de sistema `getrusage` [Torvalds, 2001]. Um dos fatores que contribuem para baixo custo de execução do *panalyser* é o uso da chamada de sistema `getrusage` ao invés da leitura constante do `/proc` para obtenção de dados.

Os dados de utilização de tempo da CPU no domínio de usuário, `ru_utime`, e no domínio do sistema operacional, `ru_stime`, juntamente com a informação de tempo real decorrido, permitem ao *panalyser* fornecer a utilização da CPU. O dado de utilização de tempo da CPU no domínio do SO não inclui o tempo de tratamento de interrupções do processador. O `ru_minflt` fornece o número de páginas de memória requisitadas pelo processo que não foram lidas do disco. O `ru_majlt` fornece o número de páginas de memória requisitadas pelo processo que foram lidas do disco. O `ru_nswap` fornece o número de operações de *swap* realizadas. Estes dados demonstram a utilização de memória primária e secundária pelo processo analisado.

Por uma limitação de implementação do *kernel* 2.4.16 do Linux apenas os campos mencionados acima são preenchidos. Os demais campos da estrutura `rusage` conteriam

dados muito importantes sobre a utilização de recursos do SO, tais como o uso da memória primária, leitura e escrita de blocos, troca de mensagens, número de sinais recebidos e trocas de contexto. A implementação do *panalyser* apresentará automaticamente estas informações quando seu preenchimento for implementado no *kernel*.

As chamadas de sistema efetuadas pelo processo analisado são classificadas e contabilizadas por classe ou tipo, permitindo identificar quais chamadas de sistema de alto custo estão sendo invocadas, e quão freqüentemente. Em conjunto com os dados mencionados são apresentados ainda a média, o desvio padrão, o coeficiente de variação e a mediana de séries temporais de medidas. Também são fornecidos, no início e no final da análise, dados sobre a utilização de memória virtual e o conjunto de páginas residentes do processo analisado.

3.2. Considerações Sobre a Implementação

A implementação do *panalyser* se baseia nas chamadas de sistema `ptrace`, `wait4` e `getrusage`. A decisão de basear a implementação do *panalyser* em chamadas de sistema, já implementadas no SO, foi tomada para facilitar a portabilidade do programa. A implementação baseada nessas chamadas de sistema causa baixo impacto no SO minimizando distorções no resultado das medições. O *panalyser* foi desenvolvido para Linux, sendo portável para todas as plataformas de *hardware* suportadas por este SO.

A chamada de sistema `ptrace` provê meios para que um processo possa observar e controlar a execução, e examinar e mudar a imagem e registradores de outro processo. Esta chamada é usada principalmente para implementar pontos de verificação e acompanhamento de chamadas de sistema em um processo rastreado por outro. A chamada de sistema `wait4` suspende a execução do processo corrente até que um processo filho, especificado nos argumentos de chamada da `wait4`, tenha terminado ou até que um sinal tenha sido entregue ao último. A chamada de sistema `getrusage` retorna a utilização corrente de recursos de um processo, segundo a estrutura `rusage`, descrita em [Torvalds, 2001].

O *panalyser* foi projetado de forma a possibilitar mudanças na sua implementação para permitir o rastreamento de mais de uma geração de processos sem um limite determinado. A atual implementação foi limitada a uma geração de filhos do processo rastreado, pois a arquitetura da maioria das aplicações que geram filhos é baseada em um processo principal que recebe as requisições e gera filhos para atendê-las. Uma versão esquemática do algoritmo de funcionamento do *panalyser* e as etapas da execução do algoritmo de monitoração e análise são descritas em [Kretschek, 2002].

3.3. Comparação com Outros Monitores

Os experimentos foram realizados em uma computador com processador Intel Pentium III a 800MHz com *cache* de 256KB integrada, 192MB de memória SDRAM a 100MHz, executando Debian Linux 3.0, com *kernel* Linux 2.4.16 i686. Este programa de teste foi desenvolvido em C padrão ANSI e compilado pelo compilador `gcc` versão 2:2.95.4-14, sem nenhuma opção explícita. O programa teste executa um número arbitrário de operações matemáticas simples, operações de controle de laço e operações de escrita e leitura em disco. Uma descrição detalhada, e o código fonte do programa de teste, podem ser encontrado em [Kretschek, 2002].

A Tabela 1 apresenta uma comparação do tempo de execução, nos domínios do usuário e sistema, de um programa de teste de comportamento determinístico, sem ser rastreado e sendo rastreado pelo `atsar`, `strace` e *panalyser*. O objetivo desta medição é comparar o grau de intrusão do *panalyser* com o das outras ferramentas.

Condições da Medição	Tempo de Execução (s)	
	dom. usuário	dom. sistema
sem rastr.	4,4096	13,4436
`atsar`	4,4160	13,7330
var. %	0,15	2,11
`strace`	7,5594	22,3054
var. %	41,67	39,73
`panalyser`	7,1756	21,3514
var. %	38,55	37,04

Tabela 1: Tempo de execução do programa de teste sem rastreamento e rastreado.

O *panalyser* expande o tempo de execução do programa de teste em 38,55 % para o domínio do usuário e 37,04 % no domínio do sistema. Isto pode ser explicado pelo grande número de chamadas de sistema efetuadas pelo programa de teste. Como o *panalyser* executa a amostragem de chamadas de sistema por evento, o grande número de eventos faz com que o algoritmo do *panalyser* seja repetido para cada evento, interferindo significativamente na execução do programa de teste. Como o `atsar` não faz amostragens das chamadas de sistema por processo ele apresenta uma menor intrusão uma vez que efetua amostragens apenas em intervalos determinados sem atender a eventos como no caso do *panalyser* e `strace`. O `strace` apresenta uma intrusão semelhante à apresentada pelo *panalyser*, pois também faz o rastreamento das chamadas de sistema.

Comparação entre Panalyser e Atsar O `atsar` fornece uma ampla gama de informações sobre a utilização total dos recursos do SO por todos os processos ativos no momento da medição. Contudo, com este programa não é possível determinar que processo está fazendo maior utilização de qual recurso.

A Tabela 2 compara a utilização de CPU pelo programa de teste nos domínios do sistema e do usuário medidas pelo *panalyser*, com a medida feita pelo `atsar` de todos os processos do sistema, inclusive do programa de teste. A grande discrepância entre as medidas de utilização de CPU no domínio do sistema para as duas ferramentas se deve ao fato do *panalyser* medir exclusivamente a utilização de recursos do programa de teste, enquanto o `atsar` inclui o tempo de tratamento de interrupções do processador em suas medições, além das atividades dos demais processos, não relacionados ao programa de teste, executando no sistema [Langeveld, 2001].

Ferramentas	Ut usu [%]	Ut sist [%]
`panalyser`	10,59	33,03
`atsar`	22,72	51,74

Tabela 2: Comparação entre a utilização de CPU nos domínios de usuário e sistema medidos pelo panalyser e atsar.

Comparação entre o Panalyser e Strace O `strace` registra todas as chamadas de sistema efetuadas por um processo, com os parâmetros e valor de retorno, bem como os sinais recebidos por um processo. Tais informações são indispensáveis para depuração de programas, porém, torna-se bastante complexo mensurar a utilização de recursos e fazer a otimização do sistema considerando apenas estas informações, pois não é possível quantificar a utilização de recursos, como CPU, memória e disco.

A Tabela 3 contém uma comparação entre os dados fornecidos pelo *panalyser* e `strace`, com a contabilização dos tipos de chamadas de sistema efetuadas pelo programa de teste. Em decorrência do comportamento determinístico do programa de teste, o número de chamadas de sistema foi igual em todas as repetições do experimento tanto medido pelo *panalyser* como pelo `strace`.

Chamadas de Sistema	Número Total	
	`panalyser`	`strace`
`read`	1082489	1082489
`_llseek`	2042233	2042233
`write`	1000248	1000248
`open`	6	6
`getrusage`	42	42
`_mmap`	10	10
`munmap`	4	4
`close`	5	5
`fstat64`	6	6
`mprotect`	2	2
`brk`	4	4
`uname`	1	1

Tabela 3: Número de chamadas de sistema efetuadas pelo programa de teste contabilizadas e classificadas pelo panalyser e strace.

Vantagens do Panalyser O *panalyser* oferece tanto as informações fornecidas pelo `atsar` quanto pelo `strace`. As informações da estrutura `getrusage` fornecem a utilização de recursos como CPU, memória primária e memória secundária, enquanto a chamada de sistema `ptrace` permite o rastreamento, classificação e totalização das chamadas de sistema para um dado processo e seus processos filhos. A integração de ambos os tipos de informações na mesma medição facilita o entendimento das informações sobre a utilização de recursos pelos processos. Como exemplo, é possível determinar rapidamente quais chamadas de sistema efetuadas por um processo estão causando o maior tempo de utilização do processador no domínio de sistema, ou ainda como o número de chamadas de sistema, efetuadas por um conjunto de processos do mesmo tipo, pode indicar a saturação de um recurso por estes processos.

4. Estudo de Caso: Apache

Esta seção contém um exemplo detalhado de uso do *panalyser* para a medição do servidor Web Apache, em condições extremas de funcionamento num ambiente controlado. As medições permitem demonstrar a utilidade, precisão e integração das informações de utilização de recursos, pelo processo principal do Apache e seus filhos, fornecidas pelo

panalyser. Primeiramente é descrito o gerador de carga de trabalho utilizado para exercitar o Apache. A seguir é descrito o ambiente experimental e as medições efetuadas são então apresentadas e discutidas.

O servidor Apache foi escolhido por ser uma aplicação que provê um serviço implementado sobre a camada de aplicação de redes TCP/IP, gerar processos filhos para atender conexões distintas, possuir *benchmarks*, ser altamente configurável e estar disponível para diversos SOs [Barford, 2001, Robinson and the Apache Group, 1995].

4.1. O Gerador de Carga SURGE

O gerador de carga de trabalho SURGE foi empregado para produzir as cargas de trabalho para o Apache no ambiente de testes que é descrito na Seção 4.2. O SURGE usa a abordagem analítica, que depende de modelagem matemática para as várias características da carga de trabalho, para gerar a carga de requisições HTTP [Barford, 2001]. O objetivo do SURGE é imitar o mais fielmente possível um fluxo de requisições HTTP originadas por uma população fixa de usuários Web.

O SURGE incorpora a idéia de número de equivalentes de usuário (EU) como uma medida da intensidade da carga de trabalho, em adição a diferentes tipos de distribuição para as seis características principais da carga de trabalho: tamanho dos arquivos, popularidade dos arquivos, localidade temporal, tamanho das requisições, intervalo entre requisições e referências a arquivos que compõem objetos em páginas HTTP. A carga gerada pelo SURGE mantém um número maior de conexões abertas com o servidor Web, quando comparado ao SPECweb96 [Barford, 2001], o que resulta numa alta utilização de CPU. Em cargas altas, o SURGE gera tráfego de rede de alta variabilidade.

4.2. Ambiente Experimental

Os experimentos foram conduzidos em um ambiente composto de três PCs conectados através de uma rede local Ethernet de 100Mbps isolada de tráfego externo. O sistema servidor estava configurado com CPU AMD Athlon(tm) de 1311MHz com 512MB de RAM. Os sistemas cliente estavam configurados com CPU Intel Pentium III de 800MHz com 192MB de RAM. Os sistemas estavam executando Debian/GNU Linux Woody com *kernel* versão 2.4.16 otimizado para os respectivos processadores. O sistema servidor executava o servidor Apache 1.3.22 Debian/GNU. Os sistemas cliente executavam 6 processos cliente SURGE cada um.

Dentre os parâmetros de configuração do Apache utilizados no experimento destacam-se o número máximo de clientes que podem se conectar simultaneamente (`MaxClients = 4800`) e o número máximo de requisições que cada processo filho pode atender (`MaxRequestsPerChild = 10000000`), que permitiram a saturação do sistema. Maiores detalhes sobre os parâmetros do Apache podem ser encontrados em [Robinson and the Apache Group, 1995].

A versão do protocolo HTTP usada pelo SURGE no experimento é a 1.1, que possui como principais características as conexões persistentes, *pipelining*, compressão de documento no nível de *link*, e *caching* de documentos. Conexões persistentes reduzem o tráfego de pacotes, uma vez que apenas uma conexão TCP é estabelecida e utilizada durante uma sessão cliente de navegação em um servidor e isto significa que um maior número de conexões podem ser abertas no servidor a qualquer momento.

4.3. Resultados dos Experimentos

Vazão Máxima e Saturação O primeiro conjunto de experimentos efetuados mediu a vazão máxima do servidor Apache no ambiente experimental descrito na Seção 4.2., e serviu para demonstrar a saturação além da vazão máxima utilizável, pelo grande aumento da latência no atendimento das conexões e da diminuição da vazão. O número de EUs foi variado de 120 até 2400 em incrementos de 120 EU, divididos em dois sistemas-cliente e as cargas mais altas levaram o servidor a um estado de sobrecarga. Cada experimento foi executado durante 10 minutos, tempo necessário para estabilizar medidas subseqüentes, conforme [Barford, 2001]. O SURGE foi configurado com 10000 arquivos únicos, resultando em aproximadamente 196 MBytes de dados de páginas HTTP no servidor. O experimento foi repetido 3 vezes e o conjunto de dados escolhido foi o que apresentou menor desvio padrão com relação a média dos dados das 3 repetições. Acima de 2040 equivalentes de usuário o servidor chega a um estado saturado, no qual a vazão média de requisições por segundo no servidor começa a declinar e a latência média para a finalização das requisições nos clientes tem seus valores sensivelmente aumentados, quando comparada aos valores da latência em operação não saturada.

Vazão x Latência O segundo conjunto de experimentos repete as medições do primeiro conjunto de experimento em 5 pontos das curvas de vazão e latência. Para isto foram escolhidos os pontos com 840 e 1560 equivalentes de usuário, abaixo do ponto de inflexão da curva de vazão; 1800 e 2040 equivalentes de usuário, onde ocorre vazão máxima; e 2280 equivalentes de usuário, onde ocorre a saturação. Estes pontos foram escolhidos de forma a reproduzir os pontos críticos nas curvas do experimento anterior, desde antes do ponto de inflexão da curva de vazão até a saturação. As curvas de vazão e latência para os pontos escolhidos são mostradas na Figura 1. Para que o servidor estivesse nas mesmas condições do experimento anterior, ou seja, com os *caches* do sistema de arquivos pré-aquecidos, para cada um dos pontos de medição foi feito o exercício do servidor com número de equivalentes de usuário imediatamente inferior ao do ponto medido (como empregado no primeiro conjunto de experimentos) seguido pela medição com o número de usuários escolhido. O experimento foi repetido 5 vezes para cada ponto escolhido e o conjunto de dados apresentado foi o que apresentou menor desvio padrão em relação a média dos dados das 5 repetições, em cada ponto.

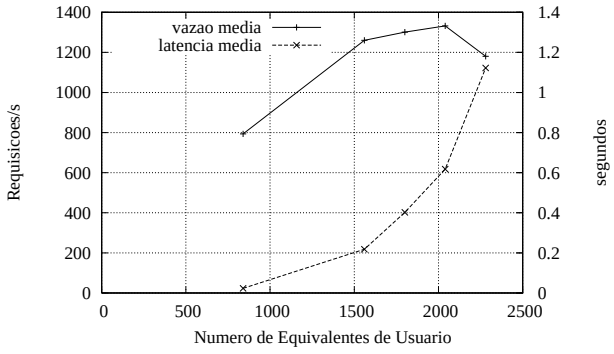

Figura 1: Vazão dos processos Apache no servidor medida pelo panalyser e latência média para a finalização das requisições nos clientes.

O intervalo de amostragem usado nestes experimentos foi de 5 segundos, para que fosse possível comparar alguns dos resultados obtidos com dados do `atsar`. O intervalo de amostragem de processos filhos foi de um rastreado a cada 20 processos criados pelo Apache. A escolha deste intervalo de amostragem foi motivada pelas conexões persistentes mantidas pelo SURGE, somada ao alto número de requisições que cada processo filho pode atender. Isto resultou num tempo de duração suficientemente longo dos processos atendendo às requisições, provocando a existência de um grande número de processos do Apache simultaneamente em memória. Caso todos os processos filhos fossem rastreados, o próprio *panalyser* contribuiria para a sobrecarga do sistema.

A Tabela 4 fornece a variação percentual na vazão e latência do Apache para a média dos dados de vazão e latência das 5 repetições do experimento nos 5 pontos escolhidos, informadas pelo SURGE, com e sem a monitoração do *panalyser.* As variações percentuais para a vazão e latência foram de menos de 2%, demonstrando a baixa distorção nas medições causada pelo *panalyser.*

EU	Com panalyser		Sem panalyser		Variação	
	vazão méd(r/s)	lat. méd(s)	vazão méd(r/s)	lat. méd(s)	vazão (%)	lat. (%)
840	793,2	0,024	798,2	0,024	0,63	0,00
1560	1256,1	0,222	1265,9	0,221	0,77	0,46
1800	1292,3	0,403	1306,8	0,399	1,11	0,99
2040	1298,7	0,648	1321,3	0,644	1,71	0,62
2280	1181,6	1,078	1189,8	1,065	0,69	1,21

Tabela 4: Variação percentual gerada pelo panalyser na vazão e latência médias do Apache.

Utilização do Processador Nos gráficos da Figura 2 e na Tabela 5 são apresentadas as médias de utilização de CPU nos domínios de usuário e sistema. Estes valores foram obtidos através da média dos tempos de utilização de CPU nos respectivos domínios em intervalos de 5 segundos. Os dados apontam uma maior utilização da CPU no domínio do sistema nos cinco pontos escolhidos nas curvas de vazão e latência, mostradas nos gráficos da Figura 1. Comportamento semelhante foi detectado nos experimentos descritos em [de Almeida, 1997, Hu et al., 1999], indicando que o tempo despendido no processamento de chamadas de sistema é maior que o tempo despendido no processamento de URLs e logs. Observa-se que na saturação, a utilização de CPU no domínio de usuário é menor do que nos outros quatro pontos anteriores à saturação. No domínio de sistema ocorre o oposto, devido ao tempo de espera dos processos pelas respostas das chamadas de sistema efetuadas, e ao grande número de processos e o conseqüente alongamento da fila de processos prontos para executar.

EU	CPU Usr.	CPU Sist.	*Page faults*/s	Cham. de Sistema	
				reads/s	*writes*/s
840	3,35	6,13	467,67	176,20	246,03
1560	4,22	8,61	500,36	152,63	200,31
1800	4,94	12,64	662,79	201,54	265,20
2040	3,34	8,36	449,12	149,92	187,24
2280	2,16	19,20	275,77	92,44	116,70

Tabela 5: Médias das medições dos recursos utilizados pelo Apache.

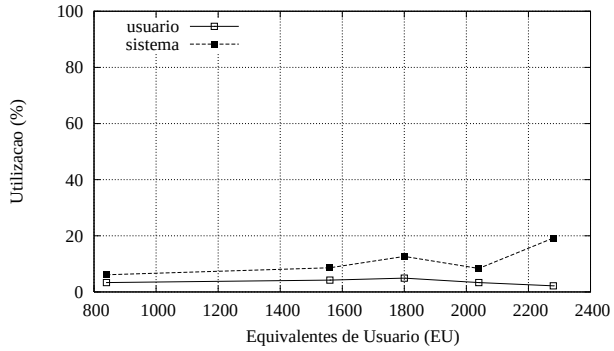

Figura 2: Média de utilização de CPU para 840 até 2280 EU, nos domínios do usuário e sistema.

Page Faults Nos gráficos da Figura 3 é apresentado o número de *page faults* que os processos incorreram nos três pontos medidos onde ocorrem a vazão máxima e a saturação, ou seja, o número de faltas que exigiram a carga de páginas do disco para a memória. A média de *page faults* que os processos incorreram é apresentada na Tabela 5. O menor número de *page faults* ocorrido no ponto de saturação é explicado pelo maior número de processos do mesmo tipo em memória e conseqüentemente a maior probabilidade de a página requerida estar em memória, um efeito similar a uma *cache* com o código dos programas. Como cada processo ficará aguardando até que as páginas requisitadas sejam carregadas na memória principal a partir da memória secundária, esta operação é de alto custo, e portanto de relevância para avaliação de desempenho de um processo [Bach, 1987].

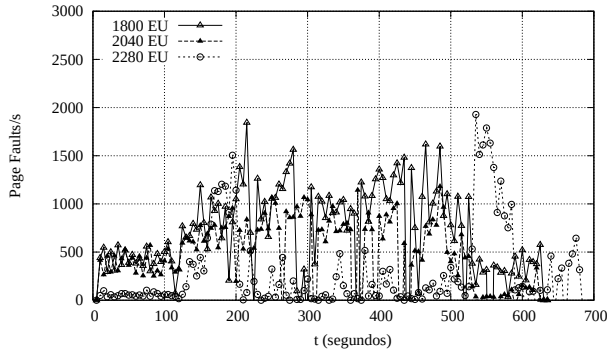

Figura 3: Número de page faults/s para 1800, 2040 e 2280 EU.

Chamadas de Sistema Nos gráficos da Figura 4 são mostradas as chamadas de sistema `read` e `write`, efetuadas pelos processos do Apache durante execução nos três pontos medidos onde ocorrem a vazão máxima e a saturação. A média das chamadas de sistema de cada tipo é apresentada na Tabela 5. É notável a correspondência entre o número de ocorrências de cada tipo de chamada de sistema em cada um dos pontos. Na saturação observa-se um menor número de chamadas de sistemas de cada tipo, durante a maior parte da medição. Isto decorre do grande número de processos aguardando a sua vez de ser executado, reduzindo significativamente a quantidade de tempo do processador dedicada a cada processo. Como a capacidade do processador, vazão da memória e disco para uma dada latência são limitados fisicamente, quanto menor o tempo de processador disponível a cada processo, menor a quantidade de tarefas executadas pelos processos. O número de

trocas de contexto necessárias, proporcional ao número de processos, contribui para o estado de saturação, pois diminui ainda mais o tempo de processamento, acessos à memória e ao disco possíveis a cada processo. Este conjunto de fatores levam a diminuição da vazão, aumento da latência e redução do número de chamadas de sistemas dos processos.

No gráfico de `writes`/s da Figura 4, aproximadamente aos 530 segundos, observa-se pontos com valores muito superiores ao restante das medições. A ocorrência de tais valores é explicada pelo fato de o intervalo entre amostragens não ser exatamente aquele declarado no parâmetro de chamada do *panalyser.* Estas diferenças na duração do intervalo decorrem do tempo de processamento do *panalyser.* Este tempo normalmente é muito pequeno e pode ser desconsiderado, mas uma troca de contexto efetuada no momento em que a amostragem está sendo processada pode alongar este tempo.

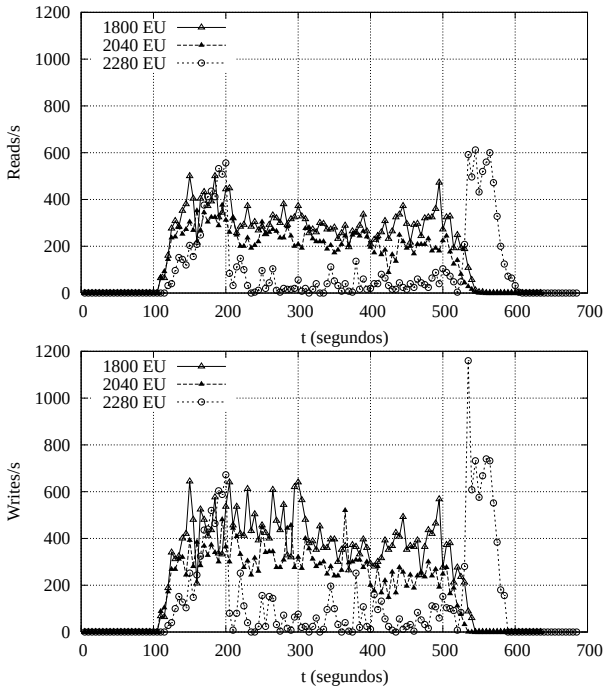

Figura 4: Número de reads/s (topo) e writes/s (base) para 1800, 2040 e 2280 EU.

É importante observar que o número de `writes`/s e `reads`/s não implicam necessariamente em acesso a disco, uma vez que no primeiro caso o SO se utiliza de escrita preguiçosa, ou seja, acumula os dados em memória antes de serem escritos no disco, só o fazendo quando estes dados são requisitados, ou quando o espaço em memória destinado a esta função é esgotado. No caso da chamada de sistema `read` o que ocorre é a leitura a partir do disco apenas quando os dados requisitados não estão em memória.

5. Conclusão

O Sistema Operacional torna disponíveis e gerencia os recursos de *hardware* de um sistema de computadores para os processos. Os dados de utilização destes recursos pelos processos são de grande relevância para o desenvolvimento e avaliação de desempenho de aplicações, o diagnóstico de pontos de contenção, a caracterização de carga, e o levantamento de parâmetros e validação de modelos de um sistema.

Este trabalho descreve a ferramenta *panalyser* que foi projetada e implementada para medição de utilização de recursos pelos processos no SO GNU/Linux. O *panalyser* é um monitor que opera em batelada sendo controlado por eventos e amostragem e fornece dados sobre a utilização de CPU, memória primária e secundária, e classificação e totalização das chamadas de sistema dos processos monitorados. Os processos monitorados podem ser quaisquer processos em execução inclusive processos filhos de processos sob exame. Além disso o *panalyser* é portável para todas as plataformas de *hardware* suportadas pelo Linux.

O preenchimento incompleto da estrutura `rusage` pelo *kernel* do Linux reduz a gama de informações fornecidas pelo *panalyser*, mas não impede sua utilização para atingir os objetivos de uso propostos. A implementação do *panalyser* contempla os campos não preenchidos da estrutura, de forma tal que quando seu preenchimento for implementado no *kernel*, estas informações serão facilmente disponibilizadas pelo *panalyser*.

A comparação dos dados fornecidos pelo *panalyser*, `atsar` e `strace` na medição de um programa de teste de comportamento determinístico atestaram a correção dos dados fornecidos pelo *panalyser*. Um estudo de caso demonstra o uso do *panalyser* no monitoramento dos recursos utilizados por um servidor Apache, quando este é submetido a diferentes intensidades de cargas, até a saturação do sistema como um todo. As medições mostraram que o percentual de utilização de CPU no domínio do sistema é superior a do domínio do usuário, em concordância com o descrito na literatura [de Almeida, 1997, Hu et al., 1999]. A maior utilização de CPU no domínio do sistema durante a saturação juntamente com uma diminuição do número de chamadas de sistema executadas pelos processos do Apache, de onde conclui-se que a utilização maior de CPU no domínio do sistema durante a saturação se deve ao escalonador de processos e não à execução das chamadas de sistema pelos processos do Apache.

A configuração do número máximo de clientes que podem se conectar simultaneamente no Apache para 2040, ponto onde o Apache atingiu sua vazão máxima no experimento, tornaria a atuação do escalonador de processos do SO mais eficiente, pelo menor número de processos em memória. A redução do número máximo de requisições que cada processo filho do Apache pode atender diminuiria o número de processos simultaneamente aguardando sua vez de executar na fila do escalonador de processos, posto que atendendo um menor número de requisições o processo terminaria mais rapidamente sua execução. Ambas as medidas contribuiriam para a não saturação do sistema.

O baixo impacto de execução, a precisão e a variedade de informações sobre utilização de recursos do SO discriminadas por processo fazem do *panalyser* uma ferramenta ideal para o ajuste fino de sistemas servidores, bem como para o desenvolvimentos e avaliação de desempenho de aplicações servidoras complexas tais como NFS, HTTP e sistemas distribuídos.

Referências

Akkerman, W. (1999). *LINUX MAN - STRACE(1)*. Debian.

Bach, M. J. (1987). *THE DESIGN OF THE UNIX OPERATING SYSTEM*. Prentice Hall, Inc.

Barford, P. R. (2001). *MODELING, MEASUREMENT AND PERFORMANCE OF WORLD WIDE WEB TRANSACTIONS*. PhD thesis, Boston University - Graduate School of Arts and Sciences.

de Almeida, J. M. (1997). Investigação sobre a interacao entre um sistema operacional e um servidor web. Master's thesis, DCC - Universidade Federal de Minas Gerais.

Hu, Y., Nanda, A., and Yang, Q. (1999). Measurement, analysis and performance improvement of the apache web server. In *18th IEEE International Performance Computing and Communications Conference*.

Jain, R. (1991). *THE ART OF COMPUTER SYSTEMS PERFORMANCE ANALYSIS: TECHNIQUES FOR EXPERIMENTAL DESIGN, MEASUREMENT, SIMULATION AND MODELING*. John Wiley & Sons, Inc.

Kretschek, M. A. (2002). Panalyser, uma ferramenta de baixo impacto para medição de utilização de recursos do sistema operacional linux. Master's thesis, Departamento de Informática - Universidade Federal do Paraná.

Langeveld, G. (2001). *LINUX MAN - ATSAR(1)*. AT Computing.

Robinson, D. and the Apache Group (1995). *APACHE - AN HTTP SERVER, REFERENCE MANUAL*.

Torvalds, L. (1996). *LINUX MAN - PROC(5)*. Transmeta.

Torvalds, L. (2001). *LINUX MAN - GETRLIMIT(2)*. Transmeta.