Utilização de Scripts para Monitoramento de Sistemas Linux: Abordagem para Criação de Relatórios e Gráficos com Ferramentas *open-source*

Matheus Torquato[1], Harley Mello[1], Lucas Torquato[2], Jean Araujo[3], Erico Guedes[4]

[1]Instituto Federal do Sertão Pernambucano - Campus Salgueiro
Salgueiro/PE - Brasil

[2]Instituto Federal de Alagoas - Reitoria
Maceió/AL - Brasil

[3]Universidade Federal Rural de Pernambuco - Unidade Acadêmica de Garanhuns
Garanhuns/PE - Brasil

[4]Instituto Federal de Alagoas - Campus Palmeira dos Índios
Palmeira dos Índios/AL - Brasil

{matheus.torquato, harley.macedo}@ifsertao-pe.edu.br

lucas.torquato@ifal.edu.br, jean@uag.ufrpe.br, erico@ifalpalmeira.edu.br

Abstract. *Systems monitoring is a real important task for managers of these environments. Managers can take decision more properly with system monitoring reports. However, the use of proprietary tools (or open-source) do not bring flexibility to show data results, as managers want. This paper proposes a approach to generate Shell Scripts codes for build customized reports of data collected, and also presents a way to plot data with open-source tools. Two case studies are presented for approach validation. The first aims to monitor a process on a operating system, and second for battery laptop discharging analysis. At the end it is possible to notice that using flexible scripts to generate data and open-source tools for data plots are valuable mechanism for system administrators.*

Resumo. *Monitoramento em sistemas computacionais torna-se uma tarefa muito importante ao passo que administradores e gerentes desses ambientes precisam de subsídios para tomada de decisão. Porém, nem sempre a utilização de ferramentas comerciais (proprietárias) e até mesmo open-source, fornece a flexibilidade desejada pelos administradores de sistemas. Este artigo propõe uma abordagem para geração de scripts Shell para criação de relatório personalizados de monitoramento das métricas pretendidas e geração de gráficos para apresentação dos dados. São apresentados dois estudos de caso para validar a abordagem proposta, o primeiro para observação de consumo de recursos por processo e o segundo para análise de descarga de bateria em laptops. Observa-se ao final que a utilização de métodos para flexibilizar o monitoramento constituem uma ferramenta de grande valia para a administração de sistemas.*

1. Introdução

Por diversas vezes, administradores de sistemas devem checar os *status* de suas aplicações e recursos computacionais para tomar decisões administrativas. Observar e avaliar o de-

sempenho das aplicações e a utilização dos recursos computacionais torna-se essencial para planejar tarefas, agendar procedimentos ou atualizar rotinas padrão nos ambientes computacionais. Dentre tais tarefas podem-se destacar: escolher os melhores horários para atualizar o sistema; checar se os recursos computacionais são suficientes para atender às demandas solicitadas; agendar desligamentos e reinicializações do sistema. O primeiro passo para realizar as tarefas com maior eficiência é monitorar o sistema em questão [Sottile and Minnich 2002].

Existem diversas ferramentas de monitoramento disponíveis [Sottile and Minnich 2002, Eranian 2006], além de softwares nativos em sistemas operacionais conhecidos como *Windows* e *Linux*, que podem ser utilizadas para monitorar recursos como memória RAM, utilização de CPU e *status* da rede. Algumas das ferramentas disponíveis para monitoramento de sistemas pode ser utilizada via linha de comando [Andresen 2004]. Com isso é possível flexibilizar a coleta de resultados utilizando-se de *scripts* ou trabalhos em lotes (*batchs*) para gerar relatórios personalizados de monitoramento.

Haja vista que os Sistemas Linux estão em crescente adoção em ambientes de servidores de rede, que geralmente hospedam aplicações da Internet ou locais, além de estarem presentes em diversos sistemas *desktop* mundo afora, o processo de monitoramento de recursos e planejamento de capacidade se torna essencial para o bom gerenciamento de aplicações e serviços que rodam nesses sistemas [tec 2010]. Esse artigo propõe uma abordagem para desenvolvimento de scripts de monitoramento flexíveis utilizando a linguagem *Shell Script* focados na geração de relatórios personalizados. Com isso, espera-se que a utilização da abordagem propicie a criação de ferramentas de monitoramento capazes de atender à diversas demandas diferentes. Ao final do trabalho são apresentados dois estudos de caso baseados na abordagem desenvolvida com intuito de corroborar sua efetividade.

É importante salientar que a metodologia proposta visa trazer flexibilidade necessária para os administradores de sistemas. O intuito é permitir que, a partir da utilização de ferramentas nativas do sistema operacional, seja possível gerar relatórios e gráficos personalizados para cada administrador.

O restante deste artigo está dividido da seguinte maneira, a próxima seção apresenta algumas ferramentas voltadas para monitoramento em sistemas Linux. A Seção 3 apresenta a metodologia para geração de relatórios personalizados de monitoramento, além de exibir os métodos utilizados para geração dos gráficos. A Seção 4 apresenta dois estudos de caso para validar a abordagem proposta. Por fim, a Seção 5 apresenta as conclusões retiradas do trabalho apresentado.

2. Ferramentas para monitoramento

Em sistemas Linux existem quatro classes principais de ferramentas de monitoramento [Andresen 2004]:

- **Monitoramento em tempo real** - Ferramentas que atualizam automaticamente as métricas observadas.
- **Comandos estáticos** - Exibem um *snapshot* do estado atual das métricas
- **Sistema de arquivos */proc*** - Um pseudo-sistema de arquivos com arquivos com métricas para observação.

- **sysstat** - Projeto Linux para observação de métricas.

Algumas das ferramentas de monitoramento são apresentadas na Tabela 1 [Andresen 2004, sys 2015].

Tabela 1. Tabela com comandos de monitoramento do sistema Linux.

Ferramenta	Classe	Objetivo
`top`	Tempo Real	Listar em tempo real os processos em andamento. Geralmente são ordenados de modo decrescente em relação à utilização do processador.
`vmstat`	Tempo Real	Fornece informação sobre processos, memória, paginação, *IO* de disco, *traps* e Atividade de CPU.
`uptime`	Comando estático	A quanto tempo o sistema está ligado e a carga de trabalho média
`free`	Comando estático	Informações sobre o estado da memória RAM e *swap*
`ps`	Comando estático	Mostra o status dos processos dependendo dos parâmetros utilizados e do privilégio do usuário solicitante.
`iostat`	*sysstat*	Mostra estatísticas de CPU, e estatísticas de entrada e saída para dispositivos, partições e sistemas de arquivo em rede.
`mpstat`	*sysstat*	Mostra estatísticas individuais ou combinadas dos processadores.
`pidstat`	*sysstat*	Estatísticas para processos Linux (IO, CPU, memória, etc)
`/proc/cpuinfo`	Sistema de arquivos */proc*	Mostra informações da CPU (Fabricante, Nº de núcleos, etc)
`/proc/meminfo`	Sistema de arquivos */proc*	Mostra informações da Memória RAM(Memória Total, Livre, Buffers, Cache, etc)
`/proc/acpi/battery/BAT1/state`	Sistema de arquivos */proc*	Mostra informações do status da bateria de um *laptop*

Outras ferramentas para filtragem de texto como **`awk`**, **`grep`** e **`sed`** podem ser bastante úteis para a geração de relatórios. Por algumas vezes é necessário filtrar a saída

de alguns comandos estáticos em busca de valores específicos, essas ferramentas podem ser parametrizadas para obter os resultados esperados [Robbins 2002].

3. Abordagem proposta

A abordagem proposta é baseada em três etapas principais:

1. Seleção dos recursos a serem monitorados;
2. Criação de um script para geração dos relatórios do monitoramento;
3. Utilização de ferramenta para geração de gráficos.

Na primeira etapa é necessário definir quais serão os recursos/processos a serem monitorados. A partir desta seleção deve-se definir qual ferramenta será utilizada para monitorar o recurso pretendido. É importante salientar que a saída produzida pela ferramenta de monitoramento deve ser apresentada em uma única linha estática. Essa linha de resultado será transferida para os arquivos de relatório. É necessário testar previamente os comandos selecionados para checar se os resultados apresentados são os esperados. Com isso será possível gerar relatórios do sistema mais coesos, além de fornecer subsídio direto para geração de gráficos.

Após a seleção dos recursos a serem monitorados e escolha das ferramentas necessárias para realizar o monitoramento, deve-se criar um *script* capaz de coletar o monitoramento e gerar os relatórios necessários. A abordagem de criação do *script* consiste em redirecionar as saídas geradas pela ferramenta de monitoramento para arquivos de relatório que são organizados de modo personalizado. O Algoritmo 1 possui uma representação genérica do script de geração de relatório.

Algoritmo 1 Script Shell genérico para geração de relatórios de monitoramento

```
1: # /bin/bash
2: echo "Contador Métrica1 Métrica2" »"Relatorio".txt
3: CONTADOR=0
4: while True do
5:     echo $CONTADOR $(Comando-para-monitorar-metrica1) $(Comando-para-
   monitorar-metrica2) »"Relatorio".txt
6:     sleep 10
7:     CONTADOR=$($CONTADOR+1)
8: end while
```

A primeira instrução dos scripts *Shell* deve conter o interpretador de comandos que irá ser utilizado. Para os propósitos apresentados nesse artigo, o interpretador selecionado é o `sh`, que é acessado através do link **`#!/bin/bash`** [Matthew and Stones 2011]. A segunda instrução serve para adicionar o cabeçalho do relatório ao arquivo. O comando **`echo`** serve para exibir *strings* na saída padrão. Com a utilização dos redirecionadores "»" é possível adicionar uma linha ao arquivo de texto **`Relatorio.txt`**. Após isso inicia-se uma variável para contar as medições realizadas. No caso, a variável iniciada é denominada de **`CONTADOR`**.

A instrução na linha 4 inicia um *loop* infinito das instruções 5, 6 e 7. O intuito é que o script seja executado pelo tempo que o administrador desejar. Para cancelar o

monitoramento basta encerrar a execução do script. A linha 5 exibe a instrução mais importante do script. Esta instrução é a responsável por exibir, em uma única linha estática, os valores das métricas selecionadas para observação. De maneira simples, ela recolhe os valores de saída dos comandos realizados para o monitoramento, organiza-os de modo a obedecerem o cabeçalho especificado, depois redireciona o conjunto de informações para o final do arquivo de relatório. Após isso, para controlar o monitoramento em andamento, espera-se um tempo determinado (no Algoritmo 1 o tempo é de 10 segundos) para refazer o monitoramento. Incrementa-se o contador e reinicia-se o monitoramento.

Ao final, para melhor observação dos resultados obtidos, deseja-se gerar alguns gráficos do monitoramento. Para realizar esta tarefa pode-se utilizar diversas ferramentas como (*MS Office Excel, Matlab, LibreOffice Calc, gnuplot*). Dada a flexibilidade da ferramenta, e o respeito às políticas de software livre, para as demonstrações mostradas neste trabalho é utilizada a ferramenta **`gnuplot`** [Williams et al. 2010]. É importante salientar que a ferramenta selecionada possui diversas possibilidades para a criação de gráficos para a visualização de dados, e que neste trabalho será apresentado apenas o gráfico XY (dispersão) com somente linhas.

É possível realizar comandos no gnuplot a partir da linha de comando. Uma das vantagens na sua utilização é a possibilidade de utilizar arquivos de template como parâmetro de entrada. Assim, é possível estabelecer modelos para reutilização posterior. Para gerar um gráfico do monitoramento a partir do resultado do script anterior é possível utilizar o *template* exibido no Arquivo 1.

Arquivo 1. Template Gnuplot

```
set terminal pdfcairo dashed
set output "graficoRelatorio.pdf"
set title "Relatório do Monitoramento"
set xlabel "Tempo (seg * 10)"
set ylabel "Valor"
plot "Relatorio.txt" using 1:2 with lines lt 1 lw 4 linecolor rgb "
   black" title "Métrica 1", "Relatorio.txt" using 1:3 with lines lt 2
    lw 4 linecolor rgb "black" title "Métrica 2".
```

Nesse exemplo, o arquivo de saída gerado é do formato `PDF`, nomeado de `graficoRelatorio.pdf`. O título do gráfico no arquivo é `Relatório do Monitoramento`, o eixo X possui o rótulo `Tempo (seg*10)` e o eixo Y, `Valor`. O comando **`plot`** indica quais são os arquivos dos quais os dados devem ser obtidos para gerar os gráficos. No caso apresentado no Arquivo 1, a fonte dados é o arquivo `Relatorio.txt`. Os parâmetros `1:2` indica que a linha a ser *plotada* obedece à coluna 1 para o eixo X e coluna 2 (o mesmo serve para a instrução `1:3`).

Os estudos de caso da próxima seção apresentam de modo mais detalhado a aplicação da abordagem para situações diversas.

4. Estudos de caso

Nesta seção serão apresentados dois estudos de caso para validar a proposta apresentada. O primeiro consiste no monitoramento de consumo de recursos por processos ativos na máquina e o segundo apresenta o monitoramento do consumo de bateria de um *laptop* dentro de um espaço de tempo determinado. É importante ressaltar que os estudos de caso

apresentados não são voltados à avaliação de desempenho dos itens observados, apenas à validação da abordagem proposta nesse trabalho.

Para cada um dos estudos de caso são apresentados os scripts e templates utilizados, bem como uma breve motivação e descrição do monitoramento realizado.

4.1. Monitoramento de consumo de recursos de um *web browser*

Monitorar processos é uma maneira de verificar se há consumo excessivo de recursos, para tomar as providências necessárias. Neste estudo de caso, verificou-se o consumo de recursos do *web browser* ***Google Chrome***[1] em sua utilização normal, sem utilização de ferramentas de *benchmark*[2].

O monitoramento foi realizado no espaço de tempo de aproximadamente duas horas. Para a medição dos recursos foram utilizados os seguintes componentes:

- **Aplicação** - *Google Chrome* na versão 38.0.2125.122 (64-bit);
- **SO** - Sistema Operacional Ubuntu Desktop 14.04.1 LTS 64Bits;
- **Hardware** - Lenovo Z470, Intel Core i5 -2410M 2.30GHz, 4GB de RAM, 750 GB de HD.

Para realizar o monitoramento do Google Chrome é necessário recuperar o **`PID`**[3] do processo. Para realizar isso utilizou-se o seguinte comando **`$ ps aux | grep chrome`**. O intuito do monitoramento é observar o percentual de recursos consumidos pelo processo em relação a todo montante disponível. Obedecendo o algoritmo genérico apresentado anteriormente, o Algoritmo 2 mostra o script desenvolvido para o monitoramento do processo em questão. No algoritmo considera-se que o **`PID`** do processo é **`5382`**.

Algoritmo 2 Monitor de processos

```
1: # /bin/bash
2: echo "Contador %CPU %MEM" »"ChromeMonitor".txt
3: CONTADOR=0
4: while True do
5:     echo $CONTADOR $(ps -p 5382 -o %cpu,%mem | grep [0-9]) »"ChromeMonitor".txt
6:     sleep 30
7:     CONTADOR=$($CONTADOR+1)
8: end while
```

O arquivo de relatório gerado obedece o padrão exibido no Arquivo 3. A organização do arquivo facilita a geração e análise de dados, pois obedece ao padrão *TSV*[4].

Arquivo 2. ChromeMonitor.txt

```
Contador %CPU %MEM
0 3.2 2.3
```

[1] http://www.google.com.br/chrome/

[2] Ferramentas utilizadas para avaliação de desempenho.

[3] ***P****rocess* ***Id****entification* - Identificador do processo

[4] *Tab Separated Values* - Valores separados por tabulação.

```
1 5.6 2.5
2 5.7 2.5
3 4.9 2.5
4 4.7 2.6
...
```

Deve-se adaptar os templates apresentado no Arquivo 1, para que o gráfico do relatório obedece o padrão desejado.

Arquivo 3. templateChrome.tpl

```
set terminal pdfcairo dashed
set output "ChromeOutput.pdf"
set title "Monitoramento do Google Chrome Web Browser"
set key right
set xrange[0:255]
set xlabel "Tempo (seg * 30)"
set ylabel "% Utilização"
plot "ChromeMonitor.txt" using 1:2 with lines lt 1 lw 4 linecolor rgb "
    black" title "CPU", "ChromeMonitor.txt" using 1:3 with lines lt 2
    lw 4 linecolor rgb "black" title "RAM"
```

Utilizou-se o **gnuplot** para a geração do gráfico exibido na Figura 1. O comando a ser realizado é o seguinte **$ gnuplot templateChrome.tpl**.

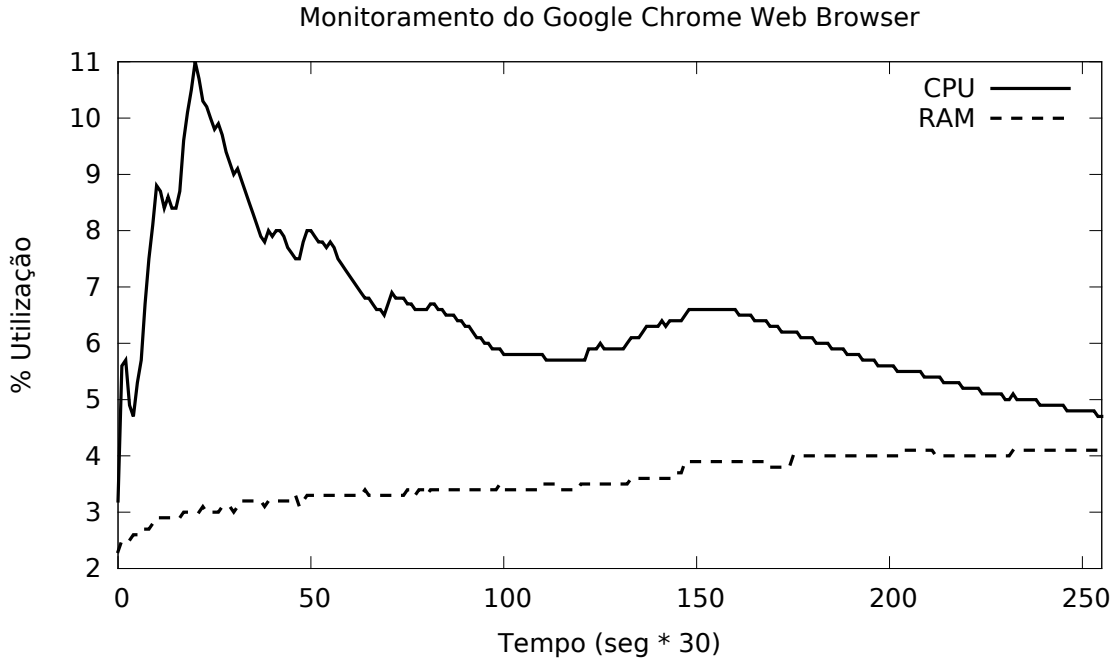

Figura 1. Gráfico gerado a partir do relatório de monitoramento do Google Chrome.

4.2. Monitoramento do consumo de bateria de um laptop

O consumo de bateria dos dispositivos móveis é uma das principais preocupações de seus usuários. Manter o computador ativo, mesmo longe de tomadas, é uma das principais

características dos *laptops*. A mobilidade do usuário está intimamente ligada à disponibilidade de carga na bateria do dispositivo. Observar o consumo de energia do dispositivo permite gerenciar melhor o seu uso, além de permitir o estudo dos gastos oriundos de sua utilização [Ayres de Souza et al. 2011].

Nesse estudo de caso é realizada uma observação do consumo de bateria de um *laptop* em seu uso normal (sem a utilização de *benchmarks*) durante aproximadamente duas horas. É importante salientar que, durante toda a observação, a única fonte de energia para o *laptop* era a bateria (ou seja, o computador estava desconectado da rede elétrica). Durante a observação a conexão de rede sem fio ficou ativa.

Para a medição do consumo de bateria foram considerados os seguintes componentes:

- **SO** - Sistema Operacional Ubuntu Desktop 14.04.1 LTS 64Bits;
- **Hardware** - Lenovo Z470, Intel Core i5 -2410M 2.30GHz, 4GB de RAM, 750 GB de HD.
- **Bateria** - Sanyo, Li-On, recarregável, Capacidade máxima: 3888 mAh, Voltagem 10800 mV, modelo L09S6Y02.

Do mesmo modo que o Estudo de Caso anterior, é necessário utilizar comandos para recuperar o status da bateria. Para tanto utilizou-se o seguinte comando: **`cat /proc/acpi/battery/BAT1/state | grep remaining | awk '{print $3}'`**. Além disso, decidiu-se recuperar os valores de: capacidade total da bateria e capacidade de alerta. A capacidade de alerta é aquela na qual o aparelho começa a acusar a necessidade de recarregar a bateria. O intuito é poder observar o consumo da bateria ao longo do tempo tendo base nos limites superior e inferior de carga.

O Algoritmo 3 apresenta o script utilizado para recolher as informações desejadas do ambiente. O cabeçalho contém três valores de interesse, o `CapacitymAh` - capacidade total da bateria, `WarnmAh` - faixa de alerta de pouca carga da bateria e `RmAh` - Carga restante.

Algoritmo 3 Monitor de descarga de bateria

```
1: # /bin/bash
2: echo "Contador CapacitymAh WarnmAh RmAh" »"BatteryMonitor".txt
3: CONTADOR=0
4: while True do
5:     echo $CONTADOR $(cat /proc/acpi/battery/BAT1/info | grep "design capacity:" | awk '{print $3}') $(cat /proc/acpi/battery/BAT1/info | grep "warn" | awk '{print $4}') $(cat /proc/acpi/battery/BAT1/state | grep remaining | awk '{print $3}') » "BatteryMonitor".txt
6:     sleep 30
7:     CONTADOR=$($CONTADOR+1)
8: end while
```

O arquivo gerado de relatório possui o mesmo formato do Estudo de Caso anterior. Para realizar a plotagem dos dados, utilizou-se o template apresentado no Estudo de Caso com algumas adaptações. O resultado da plotagem pode ser visto na Figura 2.

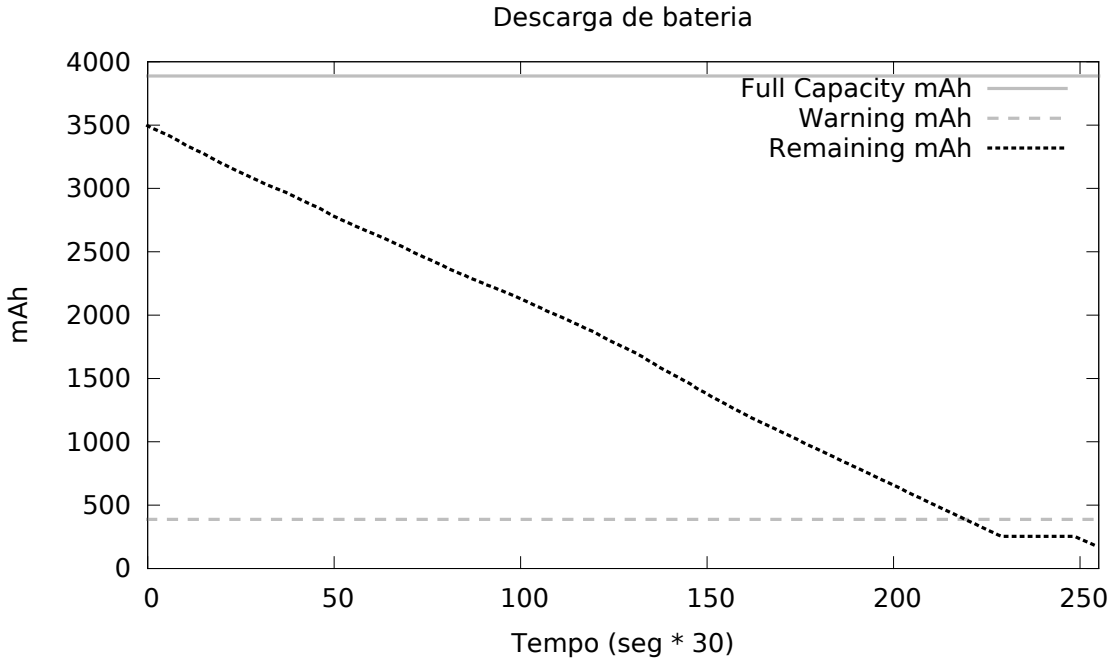

Figura 2. Gráfico gerado a partir do relatório de monitoramento da descarga de bateria.

5. Considerações Finais

A flexibilização dos relatórios de monitoramento de sistema constitui uma importante ferramenta para gerenciamento de sistemas computacionais. Nem sempre a simples utilização de ferramentas de monitoramento é capaz de fornecer dados de monitoramento específicos para administradores. Por várias vezes é necessário retratar os dados obtidos a fim de obter as informações desejadas.

Este trabalho apresentou uma metodologia para geração de *scripts* de monitoramento personalizados, além de apresentar os meios para plotagem de dados a partir de ferramentas *open-source*. É importante salientar que todo o processo apresentado é desenvolvido a partir de software livre, além de reduzir o custo com aquisição de ferramentas, ainda permite que desenvolvedores possam modificar as ferramentas quando necessário. Todo o trabalho foi desenvolvido de modo a ser aproveitado por administradores de sistemas e usuários em geral, sem grandes dificuldades de adaptação.

Como trabalhos futuros, pretende-se abranger os métodos a fim de englobar ferramentas livres para *benchmarks*. Além disso, deseja-se realizar estudos de ferramentas de monitoramento *open-source* para sistemas operacionais proprietários (e.g. Windows).

References

(2010). Linux adoption trends: A survey of enterprise end users. Technical report, The Linux Foundation.

(2015). Sysstat documentation. Technical report, Systat Home page - "http://sebastien.godard.pagesperso-orange.fr/documentation.html".

Andresen, R. (2004). Monitoring linux with native tools. In *Int. CMG Conference*, pages 345–354.

Ayres de Souza, A., Furlani Schembeck, L., and Porto de Andrade, M. A. (2011). Estudo sobre consumo de energia em notebook e gastos decorrentes. *Revista Ciências do Ambiente On-Line*, 7(1).

Eranian, S. (2006). Perfmon2: a flexible performance monitoring interface for linux. In *Proc. of the 2006 Ottawa Linux Symposium*, pages 269–288. Citeseer.

Matthew, N. and Stones, R. (2011). *Beginning Linux Programming*. John Wiley & Sons.

Robbins, A. (2002). *sed and awk Pocket Reference*. " O'Reilly Media, Inc.".

Sottile, M. J. and Minnich, R. G. (2002). Supermon: A high-speed cluster monitoring system. In *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on*, pages 39–46. IEEE.

Williams, T., Kelley, C., et al. (2010). Gnuplot 4.4: an interactive plotting program. *Official gnuplot documentation, http://sourceforge. net/projects/gnuplot*.