CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

SISTEMA DISTRIBUÍDO PARA MONITORAR RECURSOS DE HARDWARE E SOFTWARE DE ESTAÇÕES DE TRABALHO - ESTUDO DE CASO NOS LABORATÓRIOS DE INFORMÁTICA DA UNIVATES

EDUARDO DIERSMANN

Lajeado
2010

EDUARDO DIERSMANN

SISTEMA DISTRIBUÍDO PARA MONITORAR RECURSOS DE HARDWARE E SOFTWARE DE ESTAÇÕES DE TRABALHO - ESTUDO DE CASO NOS LABORATÓRIOS DE INFORMÁTICA DA UNIVATES

Trabalho de Conclusão de Curso II apresentado ao Centro de Ciências Exatas e Tecnológicas, do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Engenharia da Computação.
Área de concentração: Sistemas Operacionais e Sistemas Distribuídos.

Orientador: Maglan Cristiano Diemer

Lajeado

2010

EDUARDO DIERSMANN

SISTEMA DISTRIBUÍDO PARA MONITORAR RECURSOS DE HARDWARE E SOFTWARE DE ESTAÇÕES DE TRABALHO - ESTUDO DE CASO NOS LABORATÓRIOS DE INFORMÁTICA DA UNIVATES

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Engenharia da Computação do CETEC e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador:_____

Prof. Maglan Cristiano Diemer, UNIVATES

Mestre pela Unisinos – São Leopoldo, Brasil

Banca Examinadora:

Prof. Luis Antonio Schneiders, UNIVATES.

Mestre pelo PPGC – UFRGS - Porto Alegre, Brasil.

Prof. Mouriac Halen Diemer, UNIVATES

Mestre pela UFRGS – Porto Alegre, Brasil

Coordenador do Curso de Engenharia da Computação :_____

Prof. Marcelo de Gomensoro Malheiros

Lajeado, dezembro de 2010.

Dedico este trabalho a minha família e amigos.

AGRADECIMENTOS

Em primeiro lugar a minha noiva Marcilene pelo carinho e amor incondicional, apoio e incentivo nos momentos difíceis, e por sempre acreditar na minha capacidade. Conseguimos juntos.

Ao meu orientador Maglan, pela amizade, e apoio que permitiu a conclusão deste trabalho.

Aos colegas de trabalho Fábio, Augusto, Tiago e demais pelo apoio e auxílio sempre que necessário.

À minha mãe Dulce, que sempre acreditou e se esforçou para que o estudo fizesse parte da minha vida. Aos meus irmãos Betina e William por entender que seria um período difícil.

Ao meu pai, que certamente está orgulhoso e me receberia com um abraço e uma boa risada.

Aos meus sogros, Alvori e Lenuir, meus cunhados e cunhadas, pela preocupação e desejo de sucesso. Ao meu sobrinho Pedro pela alegria proporcionada.

A todos os professores, colegas, amigos e familiares que de uma forma ou de outra contribuíram com auxílio, opiniões ou com uma conversa descontraída que serviu de combustível nos momentos difíceis.

RESUMO

A crescente adoção da tecnologia nas empresas ocasiona a utilização de uma grande quantidade de ativos computacionais e a diversificação das aplicações dadas à eles. Isto implica na importância do gerenciamento destes recursos, garantindo a sua melhor distribuição no atendimento das necessidades de cada aplicação. O presente trabalho deu continuidade a ferramenta SDMR, um Sistema Distribuído para Monitoração de Recursos, agregando novas funcionalidades e realizando sua validação num ambiente de produção real de larga escala. A utilização da ferramenta teve por objetivo obter informações da utilização dos recursos computacionais dos Laboratórios de Informática da UNIVATES, permitindo assim produzir relatórios para auxílio na tomada de decisão de aquisição e alocação de recursos. As funcionalidades implementadas e agregadas ao sistema possibilitam a obtenção de informações mais consistentes e relevantes. Da mesma forma, validou-se a ferramenta em questões de escalabilidade e robustez, visto que a ferramenta não havia sido aplicada num ambiente de grande porte.

Palavras-chave: monitoramento de recursos, gerência de redes, parque de computadores, sistemas distribuídos.

ABSTRACT

The increasing adoption of technology in enterprises leads to the use of a large amount of computing assets and diversification of applications given to them. This implies the importance of managing these resources, ensuring their better distribution in addressing the needs of each application. This work has continued SDMR tool, a System for Distributed Resource Monitoring, adding new functions and performing its validation in a real production environment of a large scale. The use of the tool aimed at obtaining information on the use of computational resources of the Computer Labs at UNIVATES, thus producing reports to aid in making purchasing decisions and resource allocation. The features implemented and aggregate enable the system to obtain information more relevant and consistent. Likewise, the tool was validated on issues of scalability and robustness, since the engine had not been implemented in a large environment.

Keywords: resource monitoring, network management, computer park, distributed systems.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CETIC - Centro de Estudos sobre as Tecnologias da Informação e da Comunicação
CTTI - Centro de Treinamento de Tecnologia da Informação
GPL - General Public License
JSP - Java Server Pages
LAN - Local Area Network
MD5 - Message-Digest algorithm 5
MIB - Management Information Base
PID - Process Identifier
RRDTool - Round-robin Database Tool
SDMR - Sistema Distribuído para Monitoração de Recursos
SNMP - Simple Network Management Protocol
SGBD - Sistema Gerenciador de Banco de Dados
TI - Tecnologia da Informação
WEB – World Wide Web

1 INTRODUÇÃO

A utilização de recursos de Tecnologia da Informação nas empresas brasileiras é uma realidade evidente que independe do porte e do número de funcionários. A tecnologia tornou-se um requisito de competitividade e capacitação das ferramentas de trabalho, proporcionando o acesso a sistemas empresariais que gerenciam as atividades das instituições.

Este cenário pode ser evidenciado com dados do Centro de Estudos sobre as Tecnologias da Informação e da Comunicação (CETIC), em pesquisa realizada entre agosto e outubro de 2009, que aponta em 97% o índice médio de empresas brasileiras que utilizam computadores em seus ambientes de trabalho, sendo que o índice chega a 100% para empresas com mais de 50 funcionários. A pesquisa aponta ainda que no período de 2008 a 2009, o percentual médio de funcionários que utilizaram computadores no local de trabalho foi de 45%. Outro dado está relacionado ao índice de utilização de uma rede com fio LAN (Local Area Network) para conexão dos computadores, que nas empresas com mais de 250 funcionários é de 97% (CETIC, 2009).

Estas informações consolidam a afirmação de Tanenbaum (1997), de que empresas de grande porte possuem um número significativo de computadores em seu ambiente de trabalho, e que estes computadores estão conectados à uma LAN, uma rede de computadores com o objetivo de permitir que informações possam ser compartilhadas, mesmo estando localizados fisicamente distantes. Os computadores são utilizados para atender às finalidades específicas inerentes às tarefas de cada setor ou departamento em diferentes locais dentro da empresa ou até mesmo localizados mais distantes geograficamente, como em filiais.

No contexto desta grande quantidade de ativos computacionais e da diversidade de aplicações dadas a eles, Spezia (2007) afirma ser necessário que a empresa faça o gerenciamento da utilização destes recursos, para poder assim garantir a sua melhor distribuição, e atender às necessidades de cada aplicação. Algumas informações de utilização de recursos computacionais são importantes para este gerenciamento, como métricas de uso de hardware, softwares disponíveis e tempo de uso de cada recurso. Através destas informações se torna possível determinar bases para tomada de decisão de realocação de recursos e aquisição de novos equipamentos, satisfazendo mais precisamente as necessidades de cada situação.

Spezia (2007) destaca ainda que soluções em software livre de monitoramento focam suas funcionalidades na análise e gerenciamento de rede, no inventário dos computadores ou no controle de configurações, não se preocupando com a análise mais detalhada da utilização de recursos de softwares pelos usuários.

Buscando alternativa para esta deficiência, Spezia (2007) implementa um Sistema Distribuído para Monitoração de Recursos – SDMR, uma ferramenta para monitorar recursos de hardware e software de ativos computacionais com o SO (Sistema Operacional) GNU/Linux. A ferramenta SDMR implementa um agente responsável por capturar informações dos processos ativos dos usuários e do sistema, como de uso de processador e memória, disco rígido e temperatura. Implementa também um coletor, que obtém as informações dos agentes através da rede de computadores utilizando o protocolo de comunicação SNMP (Simple Network Management Protocol) e centraliza o armazenamento em um SGBD (Sistema Gerenciador de Banco de Dados).

Como exposto acima, a ferramenta SDMR implementada por Spezia (2007) foi desenvolvida somente para o sistema operacional GNU/Linux, porém em trabalho posterior, Stoll (2008) afirma que o mercado tem característica de ser heterogêneo, e mesmo que o SO GNU/Linux apresente uma crescente adoção pelas empresas, o SO Microsoft Windows possui grande utilização no mercado. Afirma também que as empresas possuem em sua infra-estrutura um parque de máquinas com diferentes sistemas operacionais, sendo necessárias aplicações multiplataforma para suportar estas diferenças. Desta forma, Stoll (2008) justifica a complementação da aplicação SDMR, e em seu trabalho desenvolve um agente para capturar as informações do SO Microsoft Windows, mantendo-se compatível com a estrutura inicial da ferramenta, e cria uma aplicação chamada “Console” para visualização de forma amigável das informações armazenadas no banco de dados.

Através da análise dos trabalhos realizados verifica-se que a ferramenta SDMR se mostra como uma alternativa para o monitoramento de recursos computacionais, no momento em que não analisa somente o valor total de utilização de recursos, mas oferece informações sobre cada processo, o que permite a criação de filtros e consultas dos dados e obtenção de resultados específicos, como por exemplo, determinar perfis de utilização de software e aplicações em um determinado equipamento.

Neste contexto, este trabalho teve por objetivo a utilização e validação do sistema SDMR, com sua implantação nos Laboratórios de Informática da UNIVATES em computadores com o sistema operacional Linux e Windows, buscando assim traçar o perfil de uso destes Sistemas Operacionais dentro da instituição. Para alcançar um ambiente capaz de fornecer informações ao gestor de TI (Tecnologia da Informação), que auxiliem sua tomada de decisão em questões de realocação e aquisição de recursos computacionais, novas funcionalidades relacionadas com o tempo de atividade da estação e o tempo de utilização pelos usuários foram implementadas, buscando desta forma obter informações administrativas para complementação dos dados já oferecidos pelo sistema.

As funcionalidades que foram agregadas ao sistema são:

a) Informação do tempo de utilização por usuário.
   - Acrescenta a possibilidade de consultar dados de utilização por usuário.

b) Informação da quantidade de tempo que o computador ficou ativo/ligado.
   - A relação entre o tempo ativo e o tempo de utilização por usuário permite obter a ociosidade do computador.

c) Distinguir os processos pelo sistema operacional onde foram executados, ao invés de pré-definir este sistema.
   - Na estrutura original do SDMR o SO era definido de modo fixo em cada agente, no entanto, existe a possibilidade de dois ou mais sistemas operacionais estarem instalados no mesmo computador, e serem usados alternadamente. Definir a qual sistema os processos coletados pertencem torna possível a distinção e análise mais coerente.

d) Oferecer mais funcionalidades no "Console". Oferecer ao gestor de TI um ambiente que possibilite a obtenção de informações adequadas à cada necessidade.
   - Relatórios: oferecer novos relatórios para visualizar informações do uso de recursos do sistema, como por exemplo:
     1) Tempo de uso por laboratório;
     2) Tempo de uso por curso;
     3) Tempo de ociosidade;

Além disso, as experimentações em ambientes de produção reais realizadas nos trabalhos de Spezia (2007) e Stoll (2008) restringiram-se à um número reduzido de computadores. O setor de Laboratórios de Informática da UNIVATES é um ambiente de produção real que possui um grande número de ativos computacionais (UNIVATES, 2010). Desta forma, para a implantação do sistema SDMR, avaliou-se se o mesmo possuiu comportamento escalável e robusto o suficiente para que uma execução que não prejudique a análise dos dados.

O presente trabalho está dividido da seguinte maneira: o Capítulo 2 deste documento apresenta a revisão da literatura dos assuntos envolvidos e trabalhos relacionados; o Capítulo 3 apresenta as implementações realizadas; o Capítulo 4 apresenta as experimentações e os resultados obtidos; e o Capitulo 5 apresenta as considerações finais sobre a realização deste trabalho.

2 REVISÃO DE LITERATURA

Este capítulo apresenta a revisão de literatura dos assuntos que envolvem o desenvolvimento deste trabalho. Primeiramente é abordado o sistema SDMR, distinguindo seus componentes, para formar a base de conhecimento sobre o seu funcionamento. Para complementação das funcionalidades é apresentado o protocolo de comunicação SNMP e os mecanismos para obtenção e filtragem das informações do sistema operacional. Ao fim são abordados os temas escalabilidade e robustez, que têm papel fundamental em garantir que a ferramenta funcione de forma adequada ao cenário proposto.

2.1 SDMR

O Sistema Distribuído para Monitoração de Recursos – SDMR tem a finalidade de extrair informações sobre o uso de recursos de hardware e software de estações de trabalho de uma determinada rede de computadores e armazenar estas informações numa base de dados para que possam ser processadas e analisadas através de um console pelo administrador do sistema. A versão implementada por Spezia (2007) e posteriormente complementada por Stoll (2008) suporta a coleta de informações de estações com os sistemas operacionais GNU/Linux e Microsoft Windows. Em Windows o agente foi desenvolvido e executado sobre o ambiente CygWin, que é um ambiente que agrega uma coleção de ferramentas em software livre para portar a execução de softwares que rodam em sistema POSIX, como o Linux, sobre o sistema operacional Windows (CYGWIN, 2010).

O sistema SDMR é composto basicamente por: um agente, que está localizado nas estações clientes monitoradas, um coletor que obtém as informações dos agentes e faz seu armazenamento em uma base de dados, e um console para visualização e análise de forma amigável das informações coletadas. Podemos verificar a representação dos componentes da ferramenta SDMR na Figura 1.

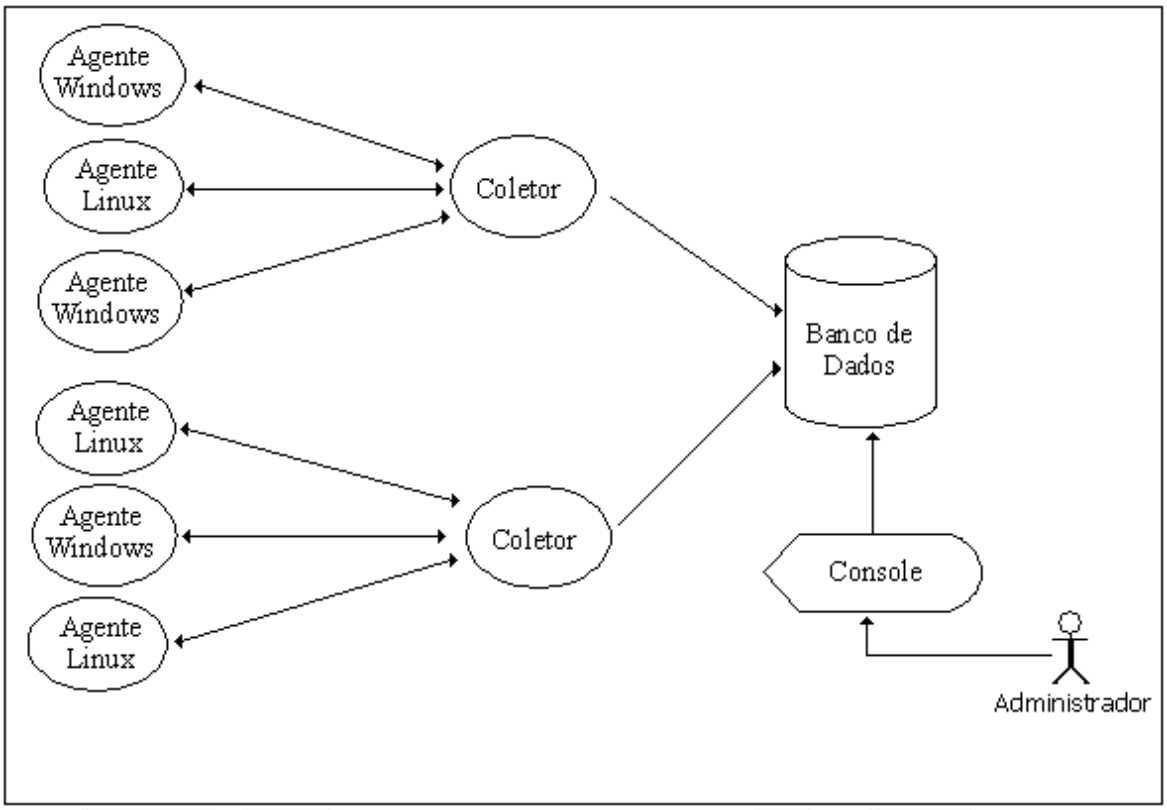


Figura 1 Estrutura do sistema distribuído para monitoramento de recursos (STOLL, 2008).

2.1.1 Agente

O agente é responsável por capturar as informações de utilização de recursos dos computadores monitorados, para isso o processo agente deve estar rodando na estação, suportando os sistemas operacionais GNU/Linux e Microsoft Windows. As informações capturadas pelo agente são consumo de CPU e memória pelos processos ativos nos sistema, possuindo a opção de capturar somente processos que utilizaram algum recurso de CPU ou todos os processos do sistema. Captura também as informações de temperatura do processador e ocupação das partições de disco rígido (STOLL, 2008).

2.1.2 Coletor

O coletor é responsável por obter as informações dos agentes através de uma requisição, processar estas informações e armazená-las numa base de dados. A obtenção dos dados é feita através do protocolo SNMP, onde cada agente publica as informações capturadas numa estrutura específica do protocolo. As requisições feitas pelo coletor obtém os dados desta estrutura. O coletor possui uma lista dos agentes que podem estar ativos na rede e

sequencialmente realiza o processo de obtenção de dados. Posteriormente os dados são armazenados numa base de dados e podem ser visualizados pelo console administrativo (STOLL, 2008).

2.1.3 Base de Dados

A base de dados possui a função de armazenar as informações obtidas pelo coletor, que podem ser posteriormente consultadas e modificadas pelo console. A estrutura modelada por Stoll (2008) possui 5 tabelas, nomeadas como *processos, particoes, temperaturas, computadores e usuarios*, conforme ilustrado na Figura 2 (STOLL, 2008).

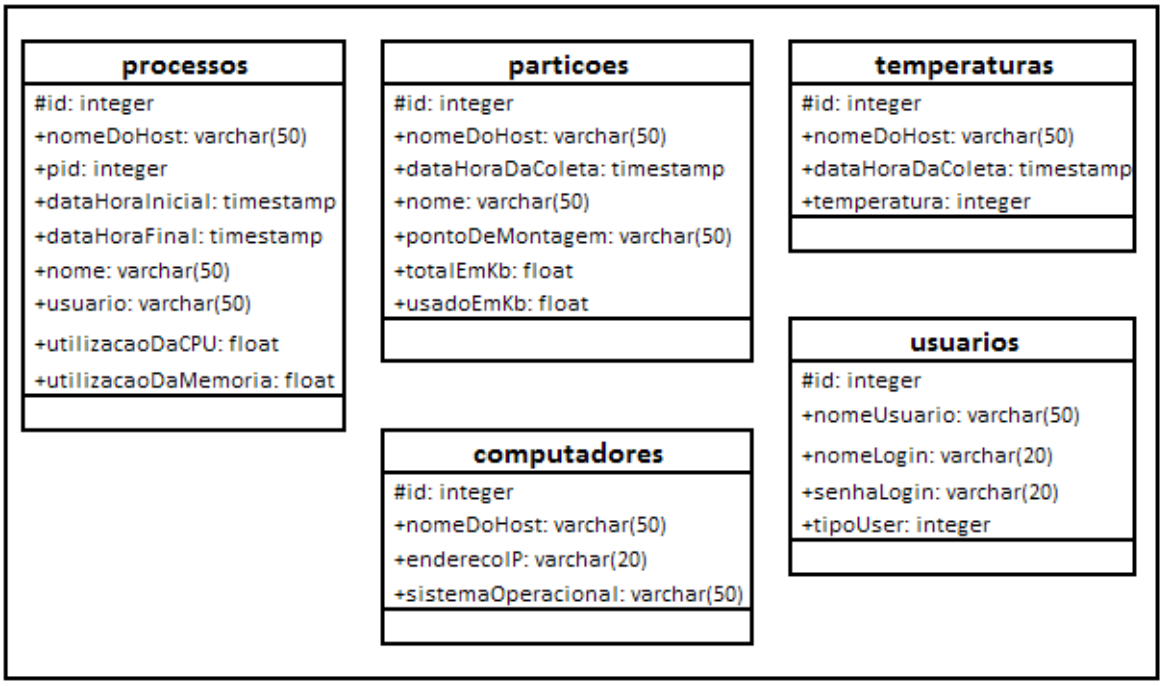


Figura 2 Estrutura da base de dados. (Fonte: STOLL, 2008, p. 55.)

2.1.4 Console

O console possui a finalidade de facilitar a consulta das informações armazenadas na base de dados. No sistema SDMR foi desenvolvida uma aplicação WEB (World Wide Web) nomeada de "console WEB" com esta finalidade. A tecnologia WEB foi escolhida em função da portabilidade, já que utiliza um servidor centralizado e tem o acesso feito via navegador de Internet. As funcionalidades implementadas no console WEB do sistema SDMR são:

a) Controle de acesso: o acesso ao sistema fica controlado pela confirmação positiva de usuário e senha. O sistema permite cadastrar e editar as informações de usuário e senha.

b) Mapa de computadores: com o objetivo de exibir o estado atual dos computadores monitorados pelo sistema. São exibidas as informações de percentual de uso de processador, de memória e de HD, e também a temperatura do processador. Utilizando o menu do console WEB é possível efetuar o filtro por sistema operacional.
c) Informações detalhadas: São exibidas as informações estatísticas da estação, assim como a lista das partições e processos da última captura realizada.
d) Consultas: três tipos de consultas foram implementadas no console WEB, são elas: consulta ao percentual de uso de processador (CPU) e percentual de uso da memória pelos processos, consulta ao percentual de uso das partições do HD e consulta à temperatura do processador (STOLL, 2008).

2.2 SNMP

O modelo utilizado para comunicação entre o agente e coletor do sistema SDMR é o SNMP, que foi escolhido por ter sido criado para ser usado na gerência e monitoração de uma rede de computadores (SPEZIA, 2007). Segundo Tanembaum (1997) o modelo SNMP é composto por quatro elementos:

a) Nós gerenciados: que podem ser computadores, ou qualquer outro dispositivo que possa se comunicar em rede. Cada nó é composto por um agente SNMP que captura e armazena informações numa estrutura de dados local.
b) Estação de gerenciamento: que é um computador genérico que emite requisições aos agentes solicitando determinadas informações e aguarda a resposta.
c) Informações de gerenciamento: são as informações adquiridas da estação monitorada.
d) Protocolo SNMP: protocolo utilizado para comunicação entre os agentes SNMP e os nós gerenciados.

O agente SNMP utiliza a estrutura de dados local denominada MIB (*Management Information Base*) para armazenar as informações sobre a utilização dos recursos do sistema operacional. No entanto, esta MIB não está configurada para armazenar as informações

capturadas pela ferramenta SDMR. Desta forma, o SDMR implementa uma MIB própria que é registrada e alimentada pelo agente específico do sistema (SPEZIA, 2007).

Na Figura 3 é possível visualizar uma representação da estrutura TCC-MIB criada e registrada especificamente pela ferramenta SDMR. A MIB é composta pelo identificador tcc, e abaixo dele foi criado o objeto *tcctabela*, que possui três outros objetos: *tccProcessTable, tccTemperatureTable e tccPartitionTable*, onde são inseridas as informações capturadas do sistema pelo agente SDMR.

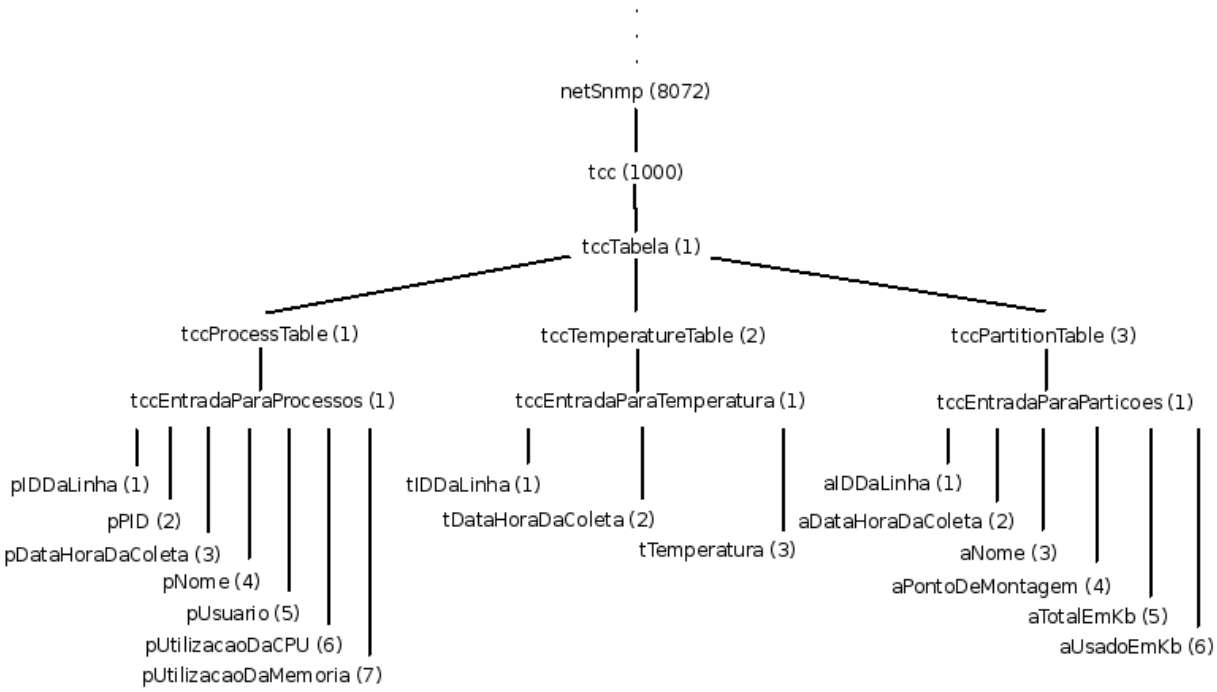


Figura 3 Visualização em árvore da MIB *tcc* (SPEZIA, 2007).

O agente SDMR opera no nó gerenciado, isto é, na estação monitorada, capturando as informações do sistema e adicionando elas na estrutura TCC-MIB registrada no sistema. O coletor SDMR atua como estação de gerenciamento e utilizando o protocolo SNMP faz requisições aos agentes, obtendo as informações dos objetos desejados das estruturas MIBs das máquinas monitoradas. O coletor requisita um objeto de cada vez, assim, ao requisitar as informações da tabela *tccProcessTable*, sete requisições devem ser feitas, uma para cada objeto que a compõe.

O protocolo SNMP define a comunicação entre o agente e o coletor do sistema SDMR. Neste processo de comunicação são utilizadas as mensagens listadas na Tabela 1, mais a mensagem de resposta. Para obter um objeto de uma estação monitorada, o coletor SDMR envia uma mensagem *get-request* passando o identificador do objeto, e aguarda a resposta do agente SDMR com a informação (SPEZIA, 2007).

Tabela 1 Tipo de mensagem da comunicação SNMP (SPEZIA, 2007)

Mensagem	Descrição
Get-request	Solicita o valor de uma ou mais variáveis do nó gerenciado
Get-next-request	Solicita ao nó gerenciado a variável seguinte a atual
Get-bulk-request	Extrai uma tabela longa do nó gerenciado
Set-request	Atualiza uma ou mais variáveis do nó gerenciado.
Inform-request	Mensagem enviada entre estações de gerenciamento para descrever uma MIB local.
SnmpV2-trap	Relatório sobre traps que é enviado de um nó gerenciado para uma estação de gerenciamento.

O modelo SNMP permite a utilização de mensagens do tipo *trap*, que são mensagens enviadas pelos agentes SNMP (nós gerenciados) para a estação de gerenciamento com a finalidade de notificar sobre a ocorrência de eventos significativos, como reinicialização, falhas, sobrecarga de rede, uma ocorrência que afete a MIB ou algum dos recursos monitorados. A estação de monitoramento deve estar preparada para receber as *traps*, e realizar o tratamento da mensagem e tomar as ações pré-estabelecidas para o evento (LEITE, 2004).

2.3 Informações do Sistema

O sistema operacional possui diversos recursos que emitem mensagens informativas de ações e da situação do sistema. Os registros gerados são chamados de *logs*. Em estações Linux são salvos no diretório "/var/log/", armazenando informações de data, hora, *host* e mensagem emitida (FERREIRA, 2003). No sistema operacional Windows os *logs* são registrados sempre que um evento significativo ocorra no sistema. Os *logs* tem por objetivo notificar os usuários e servir como auxílio na obtenção de informações do sistema ou na detecção de um problema (MICROSOFT, 2010).

A utilização dos *logs* auxilia na administração segura do sistema, porque através das informações registradas é possível obter dados sobre o funcionamento e eventos que ocorrem, como detectar a causa de algum problema ou comportamento anômalo do sistema. É possível obter informações sobre tentativas e acessos ao sistema, servindo como fonte para análise e manutenção das estações (FONSECA, 2005).

Para obtenção de informações mais específicas sobre a utilização de recursos do sistema operacional ou do seu estado atual, diversos arquivos são disponibilizados num

sistema de arquivos especial localizado diretório "/proc" (MITCHELL, 2001). Estes arquivos podem ser acessados diretamente em sistemas Linux, ou utilizando ferramentas como o CygWin em sistemas Windows. A seguir são apresentadas formas para filtragem e obtenção das informações do sistema operacional.

2.3.1 Syslog

Uma ferramenta utilizada em sistemas Linux é o Syslog, que é um *daemon* de nome *syslogd*, inicializado na carga do sistema, e é utilizado para classificar informações emitidas pelo *kernel* e de outros *daemons* do sistema, como avisos, alertas e outros tipos de mensagens. O Syslog opera as mensagens na sintaxe:

<seletor> <ação>

O campo *<seletor>* é composto por dois subcampos separados por um ponto:

<recurso>.<prioridade>

O subcampo <recurso>, também chamado de facilitador (*facility*), indica o tipo de mensagem recebida, ou a qual recurso do sistema ela está associada, e pode ser de um dos tipos indicados na Tabela 2.

Tabela 2 Mensagens identificadas no subcampo <recurso> (FERREIRA, 2003)

Recurso	**Descrição**
auth	Mensagens dos eventos de segurança
authpriv	Mensagens dos eventos de controle de acesso
kern	Mensagens do kernel
cron	Mensagens do servidor crond
daemon	Mensagens de todos os servidores
user	Mensagens dos usuários
news	Mensagens do servidor de noticias
syslog	Mensagens do servidor syslog
mail	Mensagens do servidor de e-mail
local0	Mensagens locais
local7	Mensagens de boot

O subcampo <prioridade> indica a prioridade com que a mensagem recebida será selecionada e direcionada. A Tabela 3 lista os tipos de prioridades em ordem crescente de urgência.

Tabela 3 Mensagens identificadas no subcampo <prioridade> (FERREIRA, 2003)

Prioridade	Descrição
debug	Mensagens de atividades de debug
info	Mensagens de informação
notice	Mensagens de notificação
warnign	Mensagens de advertência
err	Mensagens de erro
crit	Mensagens críticas
alert	Mensagens de alerta
emerg	Mensagens de emergência

Além das prioridades listadas acima é possível ainda utilizar caracteres especiais ou a palavra *none* no campo <recurso> ou <prioridade>, conforme listado na Tabela 4.

Tabela 4 Caracteres especiais utilizados na configuração do Syslog (FERREIRA, 2003)

Caractere	Descrição
none	Nenhuma prioridade
*	Todos os recursos ou prioridades
=	Restringe um recurso ou prioridade
-	Especifica uma exceção a determinado recurso ou prioridade
!	Especifica recursos ou prioridades que não devem ser registrados
,	Concatena múltiplos recursos
;	Concatena múltiplos seletores

O campo <ação> indica para onde a mensagem filtrada pelo campo <seletor> deverá ser enviada, podendo ser enviada a um arquivo (arquivo de log), um dispositivo, um outro computador com Syslog rodando ou uma lista de usuários. Desta forma no arquivo de configuração do Syslog, localizado em "/etc/syslog.conf", é possível configurar a ação a ser tomada para cada tipo de mensagem recebida. Na Figura 4 está representado um trecho de exemplo de configuração do arquivo syslog.conf, onde as mensagens de informação do recurso *user* serão enviadas ao arquivo "/var/log/user.log", e todas as mensagens de autenticação (*authprivate*) serão enviadas ao arquivo "/var/log/secure". Desta forma é possível filtrar mensagens de autenticação de usuários.(FERREIRA, 2003)

```
user.info                       /var/log/user.log
authprivate.*                   /var/log/secure
```

Figura 4 Exemplo de configuração do arquivo /etc/syslog.conf

Na distribuição Linux Ubuntu, a partir da versão 9.10, o *syslog* foi substituído pela ferramenta *rsyslog*, porém o princípio de funcionamento e configuração é o mesmo nas duas ferramentas. As configurações adicionadas em "/etc/syslog.conf" foram automaticamente migradas para o arquivo de configuração do *rsyslog* (LANGASEK, 2010).

2.3.2 Arquivos utmp e wtmp

Os arquivos "utmp" e "wtmp", localizados em "/var/run" e "/var/log", respectivamente, em sistemas Linux, são atualizados por programas como "login" e "init", e armazenam dados referentes às sessões dos usuários. São registrados dados como terminal da sessão, data e hora, e usuário que está usando o sistema. O arquivo "utmp" armazena dados somente de sessões abertas, enquanto o arquivo "wtmp" armazena dados de todas as sessões (FERREIRA, 2003).

```
#define UT_UNKNOWN            0
#define RUN_LVL               1
#define BOOT_TIME             2
#define NEW_TIME              3
#define OLD_TIME              4
#define INIT_PROCESS          5
#define LOGIN_PROCESS         6
#define USER_PROCESS          7
#define DEAD_PROCESS          8
#define ACCOUNTING            9

#define UT_LINESIZE           12
#define UT_NAMESIZE           32
#define UT_HOSTSIZE           256

struct exit_status {
  short int e_termination;    /* process termination status.  */
  short int e_exit;           /* process exit status.  */
};

struct utmp {
  short ut_type;              /* type of login */
  pid_t ut_pid;               /* pid of login process */
  char ut_line[UT_LINESIZE];  /* device name of tty - "/dev/" */
  char ut_id[4];              /* init id or abbrev. ttyname */
  char ut_user[UT_NAMESIZE];  /* user name */
  char ut_host[UT_HOSTSIZE];  /* hostname for remote login */
  struct exit_status ut_exit; /* The exit status of a process
                                 marked as DEAD_PROCESS. */
  long ut_session;            /* session ID, used for windowing*/
  struct timeval ut_tv;       /* time entry was made.  */
  int32_t ut_addr_v6[4];      /* IP address of remote host.  */
  char __unused[20];          /* Reserved for future use.  */
};

/* Backwards compatibility hacks.  */
#define ut_name ut_user
#ifndef _NO_UT_TIME
#define ut_time ut_tv.tv_sec
#endif
#define ut_xtime ut_tv.tv_sec
#define ut_addr ut_addr_v6[0]
```

Figura 5 Estrutura de armazenagem do arquivo binário /etc/run/utmp

O arquivo “utmp”, localizado em “/var/run/utmp”, permite descobrir quais usuários estão utilizando o sistema no momento da consulta. O arquivo é mantido sem permissão de escrita para usuários em geral, evitando que o mesmo seja modificado e perca sua integridade. O arquivo “utmp” está no formato binário e possui as informações numa sequencia de entradas que segue o padrão da estrutura representada na Figura 5, o arquivo “wtmp” possui a mesma estrutura (LMP, 2010).

Aplicativos de sistemas Linux como o *last,* utilizam o arquivo *wtmp* para buscar e exibir uma lista dos acessos realizados por usuários, exibindo os horários de entrada e saída, assim como informações sobre o tempo que o sistema ficou ativo. A Figura 6 ilustra as informações exibidas pelo comando *last* (LMP, 2010).

```
eduardod pts/2        :0.0             Thu May 13 00:40 - 02:15  (01:34)
eduardod pts/1        :0.0             Thu May 13 00:30 - 02:15  (01:44)
eduardod pts/0        :0.0             Wed May 12 23:48 - 02:15  (02:26)
eduardod tty7         :0               Wed May 12 23:37 - down   (02:57)
reboot   system boot  2.6.31-14-generi Wed May 12 23:36 - 02:35  (02:59)
```

Figura 6 Captura parcial da saída do comando *last* no Linux

Para obter as informações dos binários “utmp” e “wtmp” o arquivo “utmp.h” deve ser incluído no código da aplicação, este arquivo contém as definições das funções de acesso e da estrutura de dados. A função *getutent()* faz a leitura de uma linha do arquivo e retorna o ponteiro para a estrutura apresentada na Figura 5. A partir disso é possível a obtenção dos campos que compõem a estrutura.

2.3.3 Sistema de arquivos /proc

O diretório “/proc” é na verdade um sistema de arquivos especial onde são disponibilizadas diversas informações sobre o sistema operacional. O acesso à estas informações é feita através da leitura dos arquivos listados dentro deste diretório. No entanto, estes arquivos não estão armazenados em disco rígido, são gerados dinamicamente pelo *kernel* no momento da consulta (SPEZIA, 2007). A Figura 7 ilustra os arquivos e diretórios que fazem parte do sistema de arquivos “/proc”.

```
root@eduardod-laptop:/var/log# ls /proc/
1     13    1512  1620  24    5     6318       cmdline        misc
1042  1377  1514  1624  25    50    7270       cpuinfo        modules
1053  1392  1516  1640  26    5250  729        crypto         mounts
1076  14    1519  1672  27    5251  805        devices        mtrr
1078  1432  1520  1674  28    5252  808        diskstats      net
1090  1435  1521  1678  29    5253  821        dma            pagetypeinfo
1091  1436  1527  1694  292   5254  825        dri            partitions
1092  1440  1529  17    295   5255  829        driver         sched_debug
1097  1445  1537  1720  2981  5256  842        execdomains    schedstat
1098  1447  1542  1841  3     5257  844        fb             scsi
11    1454  1554  1880  30    5258  845        filesystems    self
1102  1462  1557  1885  32    5259  847        fs             slabinfo
1103  1464  1562  19    3288  5260  852        interrupts     softirqs
1105  1471  1564  1911  33    5261  857        iomem          stat
1107  1477  1566  1939  3306  5262  858        ioports        swaps
1110  1479  1568  2     333   5266  859        irq            sys
1112  1485  1570  20    34    5267  860        kallsyms       sysrq-trigger
1115  1491  1588  2054  36    53    868        kcore          sysvipc
1121  1493  1611  21    37    54    9          key-users      timer_list
1148  15    1612  2116  378   56    931        kmsg           timer_stats
1153  1500  1613  2117  4     57    934        kpagecount     tty
1156  1502  1614  2118  4181  58    acpi       kpageflags     uptime
1164  1504  1615  2137  442   5823  asound     latency_stats  version
12    1505  1616  2159  447   60    binder     loadavg        version_signature
1217  1508  1617  2170  46    611   buddyinfo  locks          vmallocinfo
1221  1510  1618  22    47    62    bus        mdstat         vmstat
1279  1511  1619  2351  49    6303  cgroups    meminfo        zoneinfo
```

Figura 7 Arquivos e diretórios que fazem parte do sistema de arquivos /proc.

Alguns arquivos localizados no "/proc" contém informações estatísticas sobre o uso do sistema. O arquivo "/proc/loadavg" contém informações sobra a carga do sistema, conforme ilustrado na Figura 8. Os três primeiros valores representam o número de processos ativos, que estão utilizando o processador, nos tempos 1, 5 e 15 minutos, respectivamente. O quarto valor representa uma captura instantânea dos processos em estado *running,* que estão agendados para serem executados e mais os que estão bloqueados em uma chamada do sistema, separado pela barra está o total de processos do sistema. O quinto e último valor representa o PID (*Process Identifier*) do processo mais recente que está rodando (MITCHELL, 2001).

```
root@eduardod-laptop:/proc# cat /proc/loadavg
0.13 0.14 0.09 1/292 7665
```

Figura 8 Informações contidas no arquivo /proc/loadavg

O arquivo "/proc/uptime" contém o valor de tempo desde que o sistema foi inicializado e a quantidade de tempo que o mesmo ficou em estado ocioso. Ambos os valores estão em segundos no formato de ponto flutuante. Relacionando o primeiro valor com o horário atual do sistema é possível obter o momento da inicialização. A Figura 9 ilustra as informações contidas no arquivo "/proc/uptime" (MITCHELL, 2001).

```
root@eduardod-laptop:/proc# cat /proc/uptime
47960.52 91973.55
```

Figura 9 Informações contidas no arquivo /proc/uptime

No sistema de arquivo "/proc" encontramos ainda o arquivo "/proc/meminfo", que armazena informações referentes a utilização de memória pelo sistema. Entre as informações disponibilizadas neste arquivo estão o total de memória disponível ao sistema e a quantidade livre, além das quantidades utilizadas em *buffers*, *cache* e *swap*. A Figura 10 apresentada a seguir ilustra algumas das informações contidas no arquivo. Nesta situação o total de memória disponível ao sistema é de aproximadamente 192MB, sendo que por volta de 25MB estão totalmente livres, enquanto que aproximadamente 49MB e 286MB estão sendo utilizados por *buffers* e *cache*. Estes dois últimos valores podem ser agregados ao valor de memória livre para a definição do total de memória disponível para processos.

```
eduardod@eduardod-laptop-910-64:~$ cat /proc/meminfo
MemTotal:        1924620 kB
MemFree:           27384 kB
Buffers:           39960 kB
Cached:           286340 kB
```

Figura 10 Algumas informações contidas no arquivo /proc/meminfo

Outro arquivo do diretório "/proc" que contém informações estatísticas do sistema é o "stat" ("/proc/stat"). Entre as informações armazenadas estão valores de uso do processador, quantidade de interrupções desde a inicialização do sistema, informações sobre a quantidade de processos já inicializados e que estão no momento da consulta rodando. Armazena também a informação *btime,* que contém o valor do momento em que o sistema foi inicializado, o valor armazenado é uma referência de segundos, contados deste o dia 1 de Janeiro de 1970, data conhecida como "Unix Epoch". A Figura 11 exibe a informação de *btime* contida no arquivo "/proc/stat".

```
root@eduardod-laptop:~# cat /proc/stat | grep btime
btime 1274549443
```

Figura 11 Informações *btim*e contida no arquivo /proc/stat

2.4 Escalabilidade

O termo escalabilidade não está em consenso entre os pesquisadores, sendo assim suas definições são baseadas no ambiente em que está sendo empregado o conceito. Desta forma, o termo escalabilidade será estudado a fim de obter o entendimento de seus conceitos de forma a integrá-lo com o trabalho proposto.

Um conceito mais abrangente é apresentado por Birman (2005), o qual afirma que escalabilidade é utilizada em sistemas distribuídos para determinar a habilidade de um sistema

em continuar operando mesmo quando possuir algum de seus aspectos dimensionados para uma grande escala. Como por exemplo, o aumento de ativos em uma rede de computadores monitorados por um determinado sistema irá aumentar a frequência e o número de solicitações geradas, um sistema não escalável poderia se degradar neste cenário.

Já Fowler (2006) enfoca que a escalabilidade é uma medida de como o acréscimo de recurso pode afetar o desempenho do sistema. Um sistema escalável deve permitir a adição de recurso, como de hardware, e obter proporcionalmente a melhora do sistema. Define a escalabilidade a nível de hardware suportado pelo sistema em dois sentidos:

a) Escalabilidade vertical ou escalar para cima: que significa adicionar mais recurso de hardware, como por exemplo, aumentar a memória de um servidor;
b) Escalabilidade horizontal ou escalar para fora: que significa aumentar o recurso provendo mais servidores.

Ainda segundo Fowler (2006), a escalabilidade é somente um dos fatores que afetam o desempenho de um sistema, torna-se relevante então o entendimento de alguns outros fatores, como:

a) *Throughput*: quantidade de coisas que podem ser feitas num determinado período de tempo. No caso do sistema SDMR, poderia se relacionar ao número de estações que podem ser consultadas neste período.
b) Latência: tempo mínimo para obter qualquer forma de resposta. Em ambientes que utilizam a rede de computadores a latência está relacionada ao tempo gasto entre uma solicitação e a resposta da máquina remota.
c) Carga: medida de pressão a que o sistema está submetido. No sistema SDMR, pode ser relacionado ao coletor e ao número de agentes do qual ele é responsável por monitorar.
d) Eficiência: desempenho dividido pelos recursos. Um sistema é mais eficiente se consegue executar 10 tarefas com 2 processadores, do que 15 tarefas idênticas com 4 processadores.

Por último, no contexto da multiprogramação, Cordeiro (2006), insere o termo escalabilidade em ambientes de sistemas multiprocessados como sendo um ganho de desempenho linear proporcional ao número de processadores empregados. E conclui que a forma como os componentes de software são implementados e interagem entre si definem sua eficiência sobre um ambiente multiprocessado. A arquitetura de software utilizada para a execução e divisão das tarefas do sistema deve ser avaliada para atingir a escalabilidade de um

sistema multiprocessado. Cordeiro (2006) propõe arquiteturas distintas para obtenção da escalabilidade em aplicações de software, serão comentadas as arquiteturas *multi-process* e *multi-thread*. Na Figura 12 são representados os passos sequenciais para atender a requisição de solicitação de um arquivo.

Figura 12 Estados para atender uma requisição de solicitação de arquivo (PAI, 1999)

Na arquitetura *multi-process,* são instanciados múltiplos processos da mesma aplicação no sistema operacional. Cada processo deve possuir a estrutura completa de execução de todos os passos necessários para atender a tarefa a ser desempenhada. Nesta arquitetura as atividades de acesso ao disco, uso do processador, e acesso aos recursos de memória e rede são gerenciados pelo sistema operacional, e cada processo é tratado de forma independente, concorrendo por estes recursos (CORDEIRO, 2006). A Figura 13 ilustra esta arquitetura, onde subentende-se o instanciamento de 1 até N processos para a execução das tarefas, onde cada processo possui a estrutura necessária para atender cada requisição solicitada.

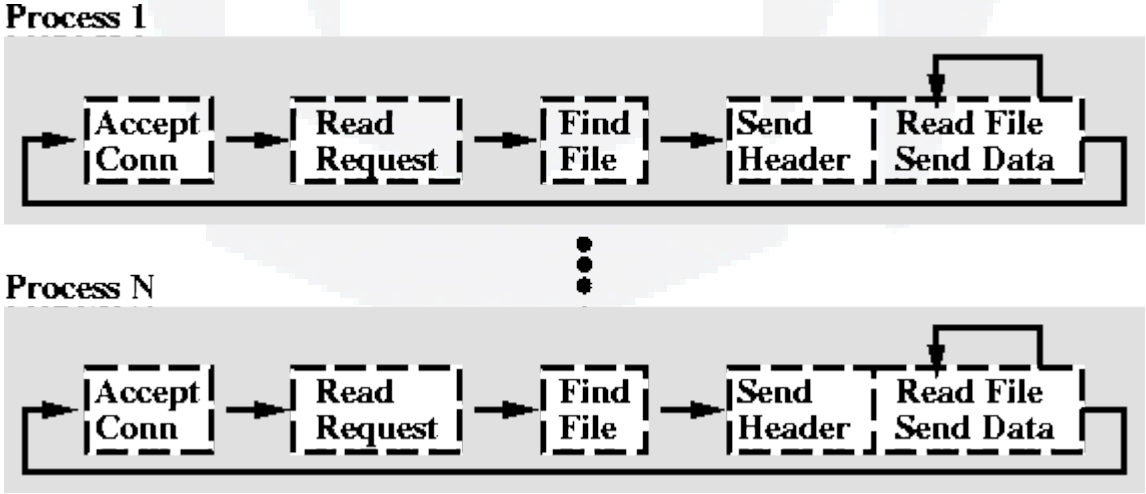


Figura 13 Atendimento da requisição utilizando *multi-process* (PAI, 1999)

A arquitetura *multi-threads* emprega o uso de múltiplas *threads* para execução das tarefas da aplicação. Um processo central é instanciado e processos filhos (*threads*) são iniciados para atender tarefas de forma paralela. Neste caso, os processos utilizam o mesmo espaço de memória compartilhado, devendo haver mecanismos para que ocorra a integridade no acesso a recursos deste tipo. Cada *thread* deve possuir a estrutura necessária para cumprir a

tarefa a ser desempenhada (CORDEIRO, 2006). A Figura 14 ilustra a arquitetura *multi-threads*, onde um processo único é instanciado e processos filhos atendem as requisições solicitadas.

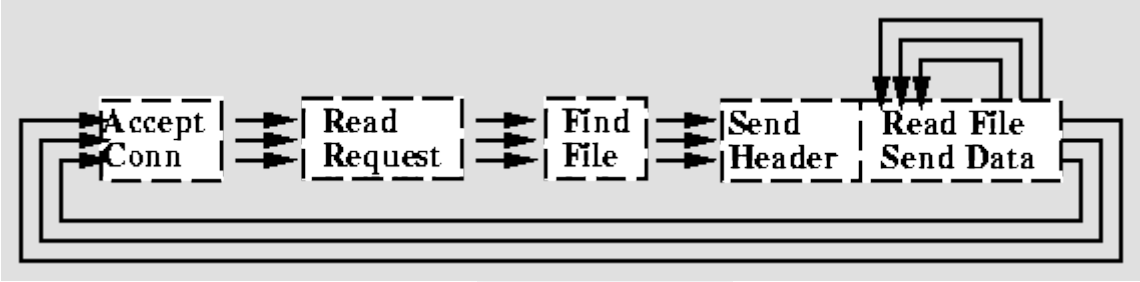


Figura 14 Atendimento da requisição utilizando *multi-thread* (PAI, 1999)

A diferenciação das arquiteturas *multi-process* e *multi-threads* é tratada por trabalhos como quase mínima, visto que sistemas operacionais modernos como o Linux implementam processos e *threads* quase da mesma forma, a diferença está no fato de que *threads* compartilham o espaço de memória com a *thread* que a criou. São elencados também pontos sobre a utilização ou não de *threads* para alcançar a escalabilidade. Alguns aspectos negativos são:

a) O acesso à regiões de memória compartilhada devem ser coordenados pelo programador;
b) Depuração mais complicada devido ao fato que *threads* introduzem dependências temporais no contexto de como foram escalonadas pelo sistema;
c) Número excessivo de *threads* pode comprometer o desempenho do sistema (CORDEIRO, 2006).

Dentre os aspectos positivos do uso de *threads* é citado que:

a) Existe um gasto menor de memória e processamento, visto que as *threads* de uma mesma *thread* principal compartilham o mesmo segmento de código, dados e espaço de endereçamento;
b) O paralelismo da aplicação pode ser alcançado no momento que o código pode ser estruturado para atender dinamicamente atividades independentes à medida que forem requisitadas (RIBEIRO, 2005);

Cabe ao final avaliar como o sistema SDMR poderia ser implementado numas destas arquiteturas de software, visto que o mesmo utiliza conceitos de sistemas distribuídos e permite a execução de múltiplos coletores. A arquitetura *multi-process* poderia ser explorada

para buscar o aumento escalar de agentes monitorados, no entanto deve-se avaliar se as múltiplas instâncias não geram sobrecarga do sistema. A utilização da arquitetura *multi-threads* permitiria a execução em paralelo do processo de obtenção realizado pelo coletor, no entanto deve-se avaliar o custo que este tipo de abordagem traz para o sistema e para a implementação, visto que esta deve ser feita de forma a isolar áreas críticas.

2.5 Robustez

Um sistema quando implantado deve ser robusto o suficiente para que seus serviços sejam prestados da forma desejada e que os componentes envolvidos em seu funcionamento se adaptem às mudanças e trabalhem da forma ao qual foram projetados. Definir a robustez de um sistema abrange a análise de vários aspectos de seu funcionamento.

A definição desejada neste trabalho tem a ver com um dos quesitos de robustez, que é a confiabilidade de execução do sistema, buscando que o mesmo seja tolerante a falhas, e que continue funcionando mesmo que algum de seus componentes pare de funcionar, e principalmente que este componente possa se recuperar desta falha ou ser novamente executado para que dê continuidade à sua tarefa (BIRMAN, 2005; ALMEIDA, 2007).

Para prover formas de recuperação de um processo ou programa em execução este deve ser monitorado, e através da obtenção de informações sobre seu estado e sobre um possível evento de fim de execução é aberta a possibilidade de tomar medidas para o tratamento deste problema, possibilitando assim que o programa possa voltar a desempenhar as tarefas para ele designadas. Uma forma de obter informações sobre processos no Linux é através do mecanismo de sinais que são enviados e recebidos dos processos em execução no sistema.

2.5.1 Sinais

O uso de sinais é um mecanismo empregado em sistemas como Linux para comunicação e manipulação de processos. São utilizados pelo sistema, por outros processos ou por usuários para comunicar à um determinado processo sobre algum evento em particular, e também para forçar o processo a executar uma função de tratamento de sinal que está previamente incluída em seu código. Um sinal é uma mensagem especial enviada a um processo, esta mensagem é processada imediatamente após sua recepção. A informação que é enviada ao processo geralmente é o número de identificação ao qual o sinal está associado, não havendo espaço para o envio de alguma informação adicional (BOVET, 2006; MITCHEL, 2001).

Quando um processo recebe um sinal a atitude a ser tomada vai depender do tipo do sinal recebido. Para cada sinal existe uma disposição padrão a ser tomada, que determina o que irá ocorrer caso o processo não tenha especificado como tratar o sinal recebido. O processo pode utilizar a declaração da função *signal()* para determinado processo para indicar como vai tratá-lo, neste instante, a execução principal é pausada, e após seu tratamento a execução continua (MITCHEL, 2001). Alguns dos sinais utilizados pelo sistema Linux são apresentados na Tabela 5, onde é descrito o nome do sinal, seu valor numérico associado, e a ação a ser tomada pelo processo.

Tabela 5 Tabela de sinais utilizados para monitorar processos em sistemas Linux (RIBEIRO, 2005)

Sinal	Valor Numérico	Ação
SIGHUP	1	Hang-Up ou desligamento. Este sinal é utilizado automaticamente quando uma sessão é desconectada ou um terminal é finalizado. É utilizado por processos servidores para invocar a releitura do arquivo de configuração.
SIGINT	2	Interrompe o processo. Emitido ao pressionar as teclas Ctrl+C.
SIGQUIT	3	Abandona o processo. Emitido ao pressionar as teclas Ctrl+D.
SIGILL	4	Emitido quando uma instrução ilegal é detectada.
SIGIOT	6	Emitido quando existe um erro de entrada e saída.
SIGFPE	8	Emitido quando existe um erro de cálculo de ponto flutuante.
SIGKILL	9	Termina o processo incondicionalmente. Pode corromper arquivos e bases de dados.
SIGSEGV	11	Emitido quando existe uma violação na segmentação ao acessar um dado fora do endereçamento do processo.
SIGSYS	12	Emitido quando existem argumentos incorretos numa chamada de sistema.
SIGPIPE	13	Emitido quando existe um erro de escrita em um pipe.
SIGTERM	15	Termina o processo de forma elegante, possibilitando que ele feche arquivo e execute suas rotinas de fim de execução.
SIGTSTP	18	Termina a execução para continuar depois. Utilizado para colocar o processo em segundo plano. Emitido ao pressionar as teclas Ctrl+Z.
SIGSTOP	19	Simplesmente termina o processo.

Existem algumas chamadas do sistema que podem ser utilizadas na codificação de um programa para que possa enviar sinais ou determinar como proceder caso um sinal seja recebido.

2.5.2 Término de um processo

O término de um processo no Linux, quando decorrido de forma normal pode acontecer de duas formas, ou o programa solicita a execução da função de saída *exit* ou a função *main* retorna, finalizando a execução. Cada processo retorna um valor ao seu processo pai ao terminar, este valor pode ser o parâmetro passado à função *exit* ou o valor retornado da função *main*. Um processo pode terminar também de forma anormal, em resposta a um sinal. Por exemplo, um sinal SIGINT é enviado à um programa quando as teclas CTRL+C são pressionadas, interrompendo sua execução (MITCHEL, 2001). Para que um sinal seja enviado de um processo para outro, a função *kill()* pode ser utilizada. Esta função é declarada na estrutura *int kill(pit_t, int sig)*.

O parâmetro *sig* indica o número do sinal a ser enviado. Enquanto o parâmetro *pid* indica o número PID do processo ou processos aos quais o sinal será enviado. O valor *pid* aceita algumas variações, que são:

a) *pid* > 0 : o sinal em *sig* é enviado somente ao processo de PID igual a *pid*;
b) *pid* = 0 : envia *sig* para todos os processos que estão no mesmo grupo do processo que está enviando o sinal;
c) *pid* = -1 : o sinal *sig* é enviado para todos os processos do sistema, com exceção de swapper(0), init(1) e o processo que está enviando o sinal;
d) *pid* < -1 : envia o sinal *sig* para todos os processos do grupo representado pelo valor absoluto em *pid* (BOVET, 2006).

2.5.3 Aguardar o término de um processo

Um processo quando iniciado através de funções como *fork* e *exec* pode ter seu término ocorrendo somente depois do fim da execução do processo pai que o criou, isto acontece porque o Linux é um sistema operacional multitarefa e escalona os processos de forma distinta, o que pode ocasionar a execução de um processo filho antes do pai.

Em alguns casos esta situação pode causar um resultado inesperado do programa, por isso algumas funções de espera são disponibilizadas para parar o fluxo de um programa até que o término de outro ocorra. Estas funções são classificadas como *wait family*, a mais simples dela é a função *wait*, que retorna ao processo pai as informações de término do processo filho. Outras funções são disponibilizadas, a opção por utilizar cada uma delas depende de quanto de informação é necessário obter sobre o término do processo monitorado.

A função *waitpid* possibilita aguardar o término de um processo específico identificando-o pelo seu *pid*. A função *wait3* retorna estatísticas de uso da CPU sobre o

processo filho e a função *wait4* permite especificar opções adicionais sobre quais os processos que se deseja aguardar (MITCHEL, 2001).

2.5.4 Processo do tipo daemon

Define-se um processo como sendo um programa em execução. Na visão de um sistema operacional o processo é a estrutura responsável pela manutenção de todas as informações necessárias à execução de um programa. Os processos são entidades independentes que como visto anteriormente, possuem um número PID associado, que entre outras informações define seus atributos.

A nível de execução os processos podem ser classificados de duas formas, ou estão em execução em primeiro plano, e são classificados como *foreground*, ou estão em execução em segundo plano, sendo classificados como *background.* Em ambas as classificações os processos são inicializados em um terminal de comando, o que os diferencia é que em processos rodando em *foreground* ocorre a interação com o usuário e o *prompt* de comando fica preso à execução do processo, diferente em tipos *background*, onde não há interação e o *prompt* fica liberado para a execução de outros processos.

Os processos podem ser classificados quanto ao seu tipo, sendo:

a) Processos interativos: são iniciados no terminal de comando a partir de uma sessão de usuário e são controlados por ele.
b) Processos em lote (*batch*): execução de um lote de processos, não precisam da interação do usuário e são executados em segundo plano, sem intervenção manual.
c) *Daemons*: são processos normalmente inicializados juntamente com o sistema e permanecendo em execução até que seja finalizado. São executados em segundo plano e não precisam da intervenção do usuário. *Daemons* são geralmente filhos do processo de PID igual a 1, o processo *init*, primeiro processo a ser iniciado pelo sistema. Por esta razão são usados para definir processos servidores, atendendo as mais diversas requisições de serviços (BOVET, 2006; FERREIRA, 2003).

2.6 Trabalhos relacionados

Nesta sessão são abordados conceitos gerais sobre algumas ferramentas que permitem o monitoramento de servidores, estações de trabalho ou ativos de uma rede de computadores. Ao final é feita uma análise de comparação entre as funcionalidades oferecidas por cada uma delas e que características levaram à escolha da ferramenta SDMR para a presente proposta.

2.6.1 Zabbix

O Zabbix é uma ferramenta que monitora vários parâmetros de uma rede, seu estado e integridade dos servidores. Utiliza um mecanismo de notificação de alertas baseados em eventos que ocorrem dentro da rede. A ferramenta oferece relatórios e visualização dos dados armazenados e possui um *frontend* que é visualizado em um navegador WEB.

Zabbix utiliza o tipo de funcionamento servidor-agente, permitindo que mais de um servidor seja executado ao mesmo tempo. O servidor recolhe os dados gerados por seus agentes rodando nas estações monitoradas. Seus agentes estão disponíveis para diversos sistemas operacionais, entre eles, Linux, MacOS X, Windows e qualquer dispositivo rodando SNMP v1, v2 e v3.

A ferramenta Zabbix oferece descoberta automática de servidores e dispositivos de rede, faz uso de *templates* para configuração das informações coletadas, monitora performance, segurança, utilização de CPU/HD. Possui a capacidade de gerar mapas da rede a partir de um ponto único central, definido pelo administrador.

As opções de alerta são definidas através de limites para cada um dos dados coletados, que são ativados e enviados aos responsáveis pelo sistema por meios de comunicação como *emails*, SMS e Jabber.

O Zabbix possui grande adoção entre os gerentes de rede e é distribuído sobre a licença GPL (General Public License), possuindo também a opção de aquisição de suporte empresarial. O servidor é suportado para Linux, MacOS X, entre outros sistemas operacionais (BLACK, 2008; ZABBIX, 2010).

2.6.2 Cacti

Cacti é uma interface *frontend* projetada para utilizar os recursos do RRDTool (Round-robin Database Tool), armazenando informações sobre o estado de uma rede de computadores. Sua interface é escrita na linguagem PHP, e utiliza as informações armazenadas para gerar gráficos para visualização do usuário.

A aquisição da informação ocorre através de mecanismo de *poller* que periodicamente obtém as informações previamente configuradas. A ferramenta permite a utilização de outras fontes de dados, como a criação de *scripts* para obtenção de determinada informação. Suporta a utilização do protocolo SNMP, o que permite a aquisição de informações através deste protocolo.

Utiliza a estrutura de *templates*, o que permite a configuração de grupo de informações a serem coletadas, permitindo aplicar este *template* a vários ativos de rede. As informações a serem obtidas são configuradas na forma de dados simples, onde cada campo corresponde à um determinado valor, como por exemplo, a aquisição do total de uso da CPU ou o total de processos em execução. A ferramenta é *open source* e possui a licença GPL (BLACK, 2008; CACTI, 2010).

2.6.3 NetEye

O NetEye foi estudado por Spezia (2007), que em seu trabalho a descreve como uma ferramenta que permite a realização de auditorias nos computadores monitorados, gerando estatísticas através de gráficos e relatórios. A ferramenta permite monitorar detalhes das atividades realizadas pelo usuários, como a utilização de softwares, acesso à Internet, arquivos utilizados, *emails* enviados e recebidos, entre outros.

O software possui a funcionalidade do administrador assumir o controle das estações monitoradas, além de receber alertas de acesso indevidos pré-configurados. Armazena também o histórico de inventário dos softwares e hardwares instalados nas estações, além da funcionalidade de atualização automática de softwares de um parque de máquinas.

O site oficial da ferramenta está em reformulação, nele é disponibilizado apenas uma versão limitada a cinco computadores para o sistema operacional Microsoft Windows. Não foram encontrados sites que disponibilizem o código fonte da ferramenta, e nem seu uso total gratuitamente, desta forma conclui-se que a ferramente não tem sua distribuição gratuita e nem é um software livre (NETEYE, 2010).

2.6.4 Zenoss

A ferramenta Zenoss é disponibilizada em duas versões, uma comercial nomeada Zenoss Enterprise, e um de código aberto de nome Zenoss Core, que é desenvolvida em conjunto com a comunidade. Esta solução é voltada para o monitoramento de TI e tem por objetivo gerenciar a configuração, a saúde, o desempenho de redes, servidores e aplicações.

Entre as funcionalidades da ferramenta estão o monitoramento de disponibilidade, que

permite monitorar serviços como HTTP, SMTP, entre outros; o monitoramento de desempenho, onde a ferramenta percorre a rede buscando informações sobre uso de memória, processos, disco rígido, além de informações sobre sistema operacional, criando assim um inventário de hardware e software; a gestão de eventos com a emissão de alertas; e o portal WEB para acesso às informações, onde são visualizados gráficos e relatórios sobre as informações coletadas (BLACK, 2008; ZENOSS, 2010).

2.6.5 Análise comparativa das ferramentas

Ao analisar as soluções descritas conclui-se que as ferramentas em software livre (Cacti, Zabbix e Zenoss) focam suas funcionalidades na obtenção de dados estatísticos de uma rede ou de ativos de rede. Elas permitem a configuração de seus agentes para obtenção de informações personalizadas, porém preservando a característica de estatística, o que permite a possibilidade de comparação com valores cadastrados.

Estas ferramentas são descritas como robustas, possuem uma grande quantidade de funcionalidades e permitem a obtenção de diversas informações. De um modo geral, capturam as informações de protocolos de rede como SNMP ou utilizam agentes para geração dos dados.

A ferramenta NetEye traz a característica de monitoração de desempenho e das atividades realizadas pelos usuários, oferecendo outras funcionalidades como inventário de hardware e software. Porém não é disponibilizada gratuitamente para ser utilizada em grande escala.

Neste contexto a opção pela ferramenta SDMR se deu à sua característica de captura e armazenamento do uso de recursos específicos utilizados por processos em execução nas estações monitoradas. Sua origem acadêmica possibilitou dar continuidade aos trabalhos realizados e avaliar sua real aplicação num ambiente de larga escala, permitindo a adição de funcionalidades especificas e aplicação de conhecimento. No próximo Capítulo são apresentadas as implementações realizadas para alcançar os objetivos deste trabalho.

3 IMPLEMENTAÇÃO

Este capítulo tem por objetivo apresentar as implementações realizadas neste trabalho para que seus objetivos fossem alcançados. Os componentes da ferramenta SDMR foram modificados e reestruturados complementando-a com novas funcionalidades que oferecem informações de tempo de atividade do computador e o tempo de utilização por usuário e por curso, e ainda dados referentes à utilização da memória física disponível.

Apresenta também a reestruturação do "Console" de administração, com a implementação de relatórios de tempo de uso por usuário ou por curso, relatórios de ociosidade e disponibilizando novas telas para cadastro das estações monitoradas.

Outras características foram agregadas ao sistema para que os objetivos do trabalho fossem alcançados. Mais adiante os assuntos são discutidos durante a apresentação da reestruturação dos componentes envolvidos. Abaixo segue a relação das principais modificações agregadas ao sistema:

a) Uso de coletores em arquitetura *multi-process*. Como a estrutura de cada coletor possui as funcionalidades suficientes para a execução total do processo de coleta de uma ou mais estações, a execução de múltiplos coletores é possível e caracteriza a aplicação da arquitetura.
b) Cadastro dos coletores na base de dados. Diversos coletores são cadastrados pelo console *WEB*, onde são associados aos computadores que por ele devem ser monitorados.
c) Lista de computadores associadas a coletores. Cada computador é associado à um coletor, o que permite ao coletor identificar quais máquinas ele está responsável por coletar as informações.
d) Novos parâmetros de configuração para o coletor. Adicionada a possibilidade de configuração de atributos de controle de coleta, onde um computador pode ser colocado em espera, neste estado a estação não estará apta a ser coletada, diminuindo assim o tempo gerado na busca de uma estação desligada.

Por fim, este capítulo foi sub-dividido da seguinte forma: num primeiro momento será apresentada a forma como as novas informações capturadas foram classificadas para posterior adaptação dos componentes envolvidos no processo. Em seguida é apresentada a reestruturação de cada um dos componentes: agente, MIB, coletor e base de dados. Após, apresentam-se as novas funcionalidades de relatórios e cadastros implementadas no Console.

3.1 Novas funcionalidades de captura de informações agregadas ao sistema

As novas funcionalidades agregadas ao sistema compreendem a captura de informações de tempo de atividade do computador, assim como o tempo de uso por cada usuário, e também a captura do uso do recurso de memória pela estação monitorada. A seguir são descritas e classificadas tais informações, definindo assim o padrão dos dados coletados.

3.1.1 Informação de tempo ativo e sessão

Dentre as informações adicionais capturadas pelo sistema SDMR e armazenadas na base de dados, as principais compreendem dados relativos ao tempo de atividade e de uso das estações monitoradas. O intervalo de tempo em que a estação permaneceu ativa, isto é, ligada e disponível para captura das informações é definido como “tempo ativo”, e o intervalo de tempo de utilização da estação por cada usuário é definido como “sessão”.

O tempo ativo da estação monitorada determina o intervalo de tempo em que a mesma permaneceu ligada e disponível para uso e captura das informações do sistema. Cada entrada de tempo ativo armazenado compreende o intervalo de tempo decorrido entre cada inicialização do sistema e o seu desligamento.

As informações das sessões determinam o intervalo de tempo de uso da estação monitorada por cada usuário. Cada entrada de sessão compreende o intervalo de tempo decorrido entre o início de uma sessão pelo usuário e o seu encerramento. O usuário é identificado por seu *login*, que posteriormente é relacionado ao seu código de aluno e consequentemente ao curso ao qual está matriculado, permitindo assim, obter estatísticas de uso por curso.

Cada estação monitorada tende a possuir diversas entradas de intervalos de tempo ativo e compreendidas nestes intervalos poderão existir sessões de usuários. A Figura 15 exemplifica uma destas situações, onde é representado o ciclo de atividade de uma estação ao longo do tempo, onde se observa o início do tempo ativo, a captura de *n* sessões, que possuem um início e fim de sessão, e o fim do tempo ativo.

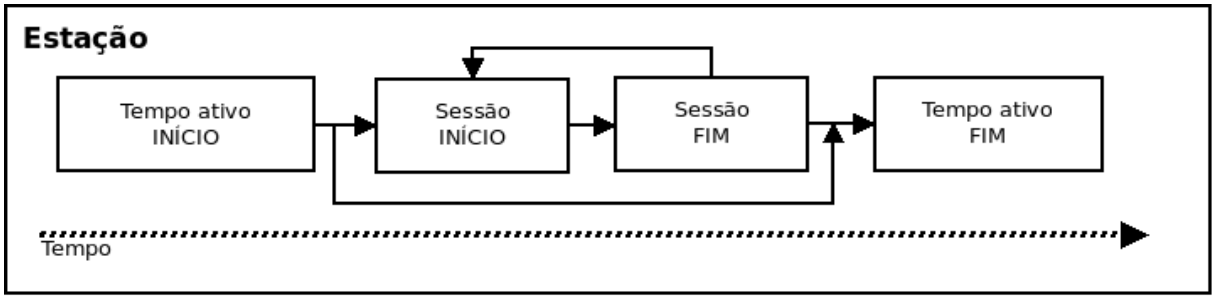


Figura 15 Representação do ciclo de estados de um tempo ativo e sessões de uma estação.

As informações de tempo ativo e sessão são capturadas em forma de eventos que contém a seguinte estrutura:

a) Nome da estação: para identificar a estação;
b) Sistema operacional: para identificar o sistema operacional ativo;
c) Data e hora: para identificar a data e hora que o evento ocorreu;
d) Usuário: usado para identificar o usuário correspondente da sessão, no caso da situação estar relacionada ao tempo ativo este campo não apresenta valor;
e) Situação: a classificação do evento capturado.

O campo Situação indica a classificação do evento que ocorreu. Define-se quatro possíveis situações conforme representado na Tabela 6.

Tabela 6 Situações possíveis para a estação monitorada

Situação	**Descrição**
LIGADO	Estação está ligada.
DESLIGADO	Estação está desligada.
LOGIN	Usuário iniciou o uso da estação.
LOGOUT	Usuário finalizou o uso da estação.

Na Figura 16 é apresentado um diagrama geral de como o sistema SDMR interage para registrar os eventos de tempo ativo e sessão. O agente captura os eventos de LIGADO, LOGIN e LOGOUT, os quais serão publicados na MIB. O coletor detecta o evento DESLIGADO caso não consiga realizar a obtenção das informações da MIB. Por fim as informações são armazenadas na base de dados do sistema.

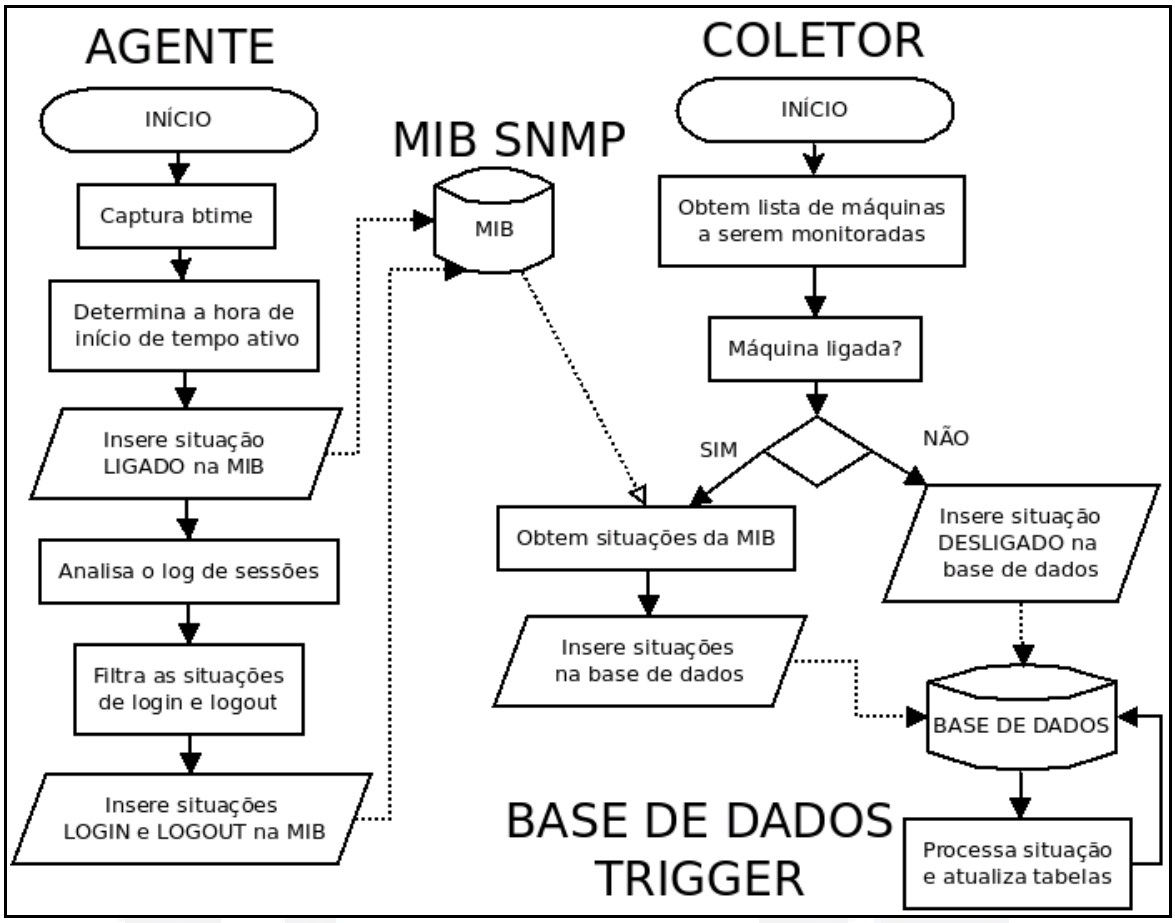


Figura 16 Diagrama geral do processo de obtenção de tempo ativo e sessões

3.1.2 Informação de utilização da memória da estação

A captura da informação de utilização da memória da estação monitorada compreende a leitura dos dados do total do recurso oferecido ao sistema operacional, e a quantidade que está disponível no momento determinado da captura. O cruzamento das informações de memória total e livre permite obter a informação da quantidade de memória utilizada na estação monitorada.

Estas informações são classificadas em:

a) Nome da estação: para identificar a estação;
b) Data e hora: para identificar a data e hora que o evento ocorreu;
c) Memória total: indica o total de memória disponibilizada para uso no sistema.
d) Memória livre: indica o total de memória não utilizada.

Após definir as novas informações agregadas à estrutura do sistema SDMR serão apresentadas as adaptações e reestruturações dos componentes envolvidos:

a) Estrutura da base de dados que armazena as informações;

b) Estrutura do agente para captura das informações e publicação na MIB;
c) A MIB que suporta o recebimento das informações;
d) Estrutura do coletor para obter as informações da MIB e armazená-las na base de dados;

Após definidos os métodos de classificação das informações são apresentadas as modificações dos componentes envolvidos na execução do sistema SDMR.

3.2 Reestruturação da base de dados

A reestruturação da base de dados do sistema SDMR compreende a criação e adaptação de diversas tabelas, para que desta forma as informações capturadas pelo agente, e coletadas pelo coletor possam ser armazenadas permitindo sua análise posterior. Estas tabelas compreendem o armazenamento das informações de:

a) Tempo ativo e sessões;
b) Utilização de Memória;
c) Cadastro de computadores, prédios, laboratórios e coletores;
d) Avisos do sistema;
e) Alertas do sistema;
f) Cursos.

A seguir as tabelas criadas para o armazenamento das informações são apresentadas, assim como são especificados todos os campos que as compõem.

3.2.1 Tabelas para armazenar informações de tempo ativo e sessões

Para que seja possível o armazenamento das informações de tempo ativo e sessões adquiridas das estações monitoradas novas tabelas foram adicionadas à estrutura da base de dados. Na Figura 17 estão representadas as seis novas tabelas que suportam o armazenamento das novas funcionalidades a serem incorporadas ao sistema SDMR, que são *tempoAtivo*, *tempoAtivoAberto, sessao*, *sessaoAberto*, *eventos e sistemaOperacional.*

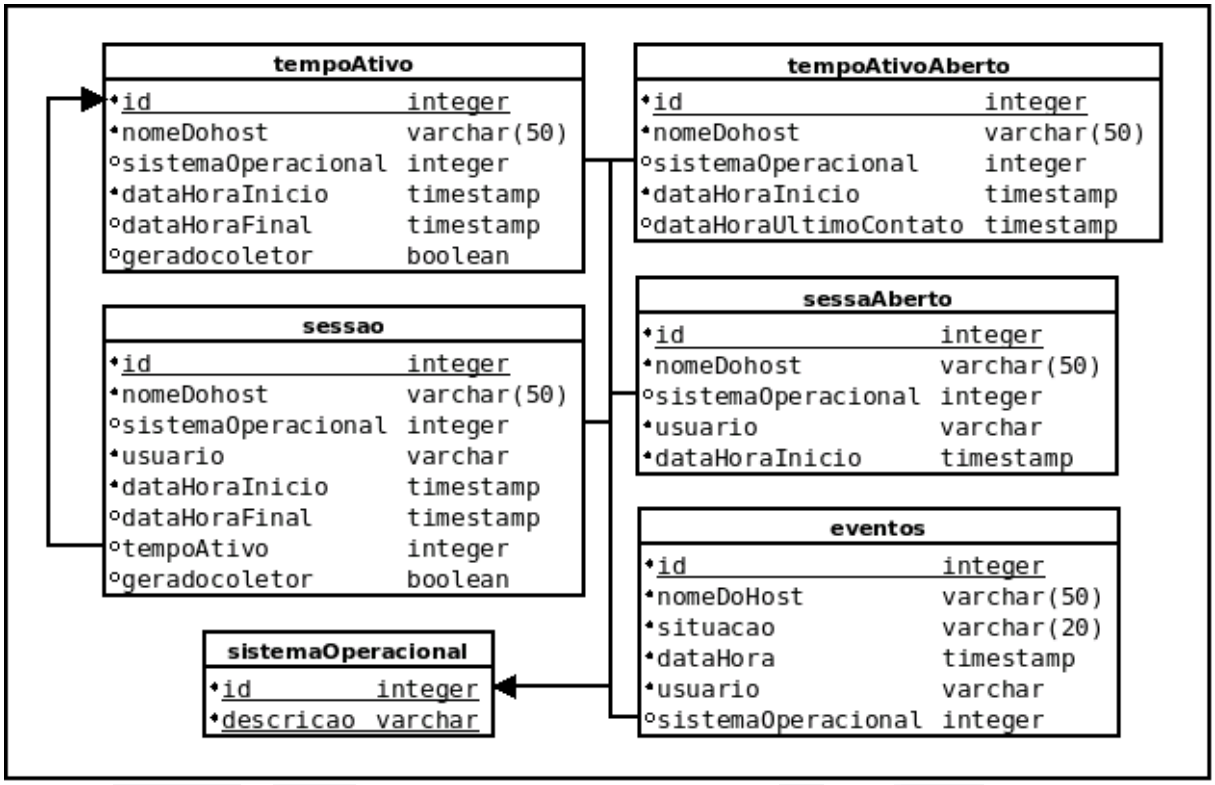


Figura 17 Novas tabelas para armazenar as informações de tempo ativo e sessões

A tabela *sistemaOperacional* armazena as informações dos sistemas operacionais suportados pelo sistema SDMR. É composto por dois campos que são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.

b) *descricao*: campo texto que contém a descrição do sistema operacional.

A tabela *tempoAtivo* armazena as informações referentes ao tempo de atividade das estações monitoradas. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.

b) *nomeDoHost*: registro de texto que identifica o nome da estação da qual foi coletada a informação.

c) *sistemaOperacional*: armazena o código do sistema operacional da estação da qual foi obtida a informação de tempo ativo. Chave estrangeira da tabela *sistemaOperacional*.

d) *dataHoraInicio*: armazena a data e hora no formato *timestamp* do início da atividade da estação monitorada.

e) *dataHoraFim*: armazena a data e hora no formato *timestamp* do fim da atividade da estação monitorada.

f) *dataHoraUltimoContato*: armazena a data e hora no formato *timestamp* da última situação LIGADO recebida.

A tabela *tempoAtivoAberto* armazena as informações de tempo ativo para estações com entradas abertas, que não receberam o evento DESLIGADO. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *nomeDoHost*: registro de texto que identifica o nome da estação da qual foi coletada a informação.
c) *sistemaOperacional*: armazena o código do sistema operacional da estação da qual foi obtida a informação de tempo ativo. Chave estrangeira da tabela *sistemaOperacional*.
d) *dataHoraInicio*: armazena a data e hora no formato *timestamp* do início da atividade da estação monitorada.
e) *geradoColetor*: campo do tipo *boolean* que indica que o evento foi gerado pelo coletor e não a partir da coleta da MIB, o que permite restaurar um estado anterior a uma falha de rede.

A tabela *sessao* armazena as informações referentes às sessões de usuário das estações monitoradas. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *nomeDoHost*: registro de texto que identifica o nome da estação da qual foi coletada a informação.
c) *sistemaOperacional*: armazena o código do sistema operacional da estação da qual foi obtida a informação de sessão. Chave estrangeira da tabela *sistemaOperacional*.
d) *usuario*: armazena o usuário da sessão.
e) *dataHoraInicio*: armazena a data e hora no formato *timestamp* do início da atividade da estação monitorada.
f) *dataHoraFim*: armazena a data e hora no formato *timestamp* do fim da atividade da estação monitorada.
g) *tempoAtivo*: campo inteiro de chave estrangeira que indica à qual tempoAtivo a sessão pertence.

A tabela *sessaoAberto* armazena as informações referentes às sessões de usuário das estações monitoradas com entradas abertas. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de

dados no momento da inserção de um registro.

b) *nomeDoHost*: registro de texto que identifica o nome da estação da qual foi coletada a informação.

c) *sistemaOperacional*: armazena o código do sistema operacional da estação da qual foi obtida a informação de sessão. Chave estrangeira da tabela *sistemaOperacional*.

d) *usuario*: armazena o usuário proprietário da sessão obtida da estação monitorada.

e) *dataHoraInicio*: armazena a data e hora no formato *timestamp* do início da atividade da estação monitorada.

f) *geradoColetor*: campo do tipo *boolean* que indica que o evento foi gerado pelo coletor e não a partir da coleta da MIB, o que permite restaurar um estado anterior a uma falha de rede.

A tabela *eventos* armazena as informações referentes aos eventos obtidos pelo sistema SDMR que darão origem as informações finais de tempo ativo e sessão. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.

b) *nomeDoHost*: registro de texto que identifica o nome da estação da qual foi coletada a informação.

c) *situação*: campo do tipo texto que armazena a situação do evento que ocorreu.

d) *dataHora*: armazena a data e hora no formato *timestamp* do momento que a situação em questão ocorreu.

e) *usuario*: armazena o usuário relacionado ao evento.

f) *sistemaOperacional:* armazena o código do sistema operacional da estação da qual foi obtida a informação. Chave estrangeira da tabela *sistemaOperacional.*

Os eventos serão inseridos diretamente na tabela *eventos* e serão tratados por uma *trigger*, a qual irá processar as informações de tempo ativo e sessão, determinando assim que tabelas devem ser atualizadas. A seguir são feitas as considerações sobre o funcionamento da *trigger* implementada.

3.2.2 *Trigger* sobre a tabela eventos

A utilização da *trigger* tem por objetivo que o próprio SGDB, que é detentor das informações consolidadas, faça a distinção da situação inserida e possa avaliar e tomar a ação necessária para atualizar as tabelas referentes a tempo ativo e sessão. A seguir são apresentados os fluxogramas de tratamento de cada uma das situações possíveis de serem inseridas.

Caso um registro com a situação LIGADO seja inserido na tabela *eventos* o algoritmo primeiro verifica se já existe um registro de tempo ativo aberto para a estação:

a) No caso de já haver, é verificado se o campo que indica o início de atividade inserido é igual ao armazenado:
   - Sendo iguais constata-se que se trata apenas de um registro de aviso de continuação de atividade e o campo de último contato é atualizado.
   - Sendo diferentes constata-se que se trata de uma nova inicialização da estação, neste caso os registros de tempo ativo e sessões são fechados e um novo registro de tempo ativo é aberto utilizando-se as novas informações recebidas;

b) No caso de não haver um registro de tempo ativo aberto, é feita uma verificação nos registros já fechados buscando-se constatar se o coletor não registrou um falso desligamento, que pode ser ocasionado por um problema de rede. Neste caso, é verificado se o último registro de tempo ativo possui o valor do campo de data e hora de início igual ao novo valor:
   - Sendo iguais constata-se que um falso desligamento foi gerado e restaura-se o tempo ativo abrindo novamente seu registro, e de todas as sessões que foram fechadas por este desligamento, que são marcadas pelo campo *geradoColetor* de cada registro armazenado na tabela sessão.
   - Sendo diferentes constata-se que se trata de uma nova inicialização da estação e um registro de tempo ativo é aberto para ela.

A Figura 18 abaixo ilustra o funcionamento do algoritmo tratando a inserção da situação LIGADO na tabela eventos e tomando as ações necessárias para o registro do tempo ativo ou a restauração de um falso desligamento.

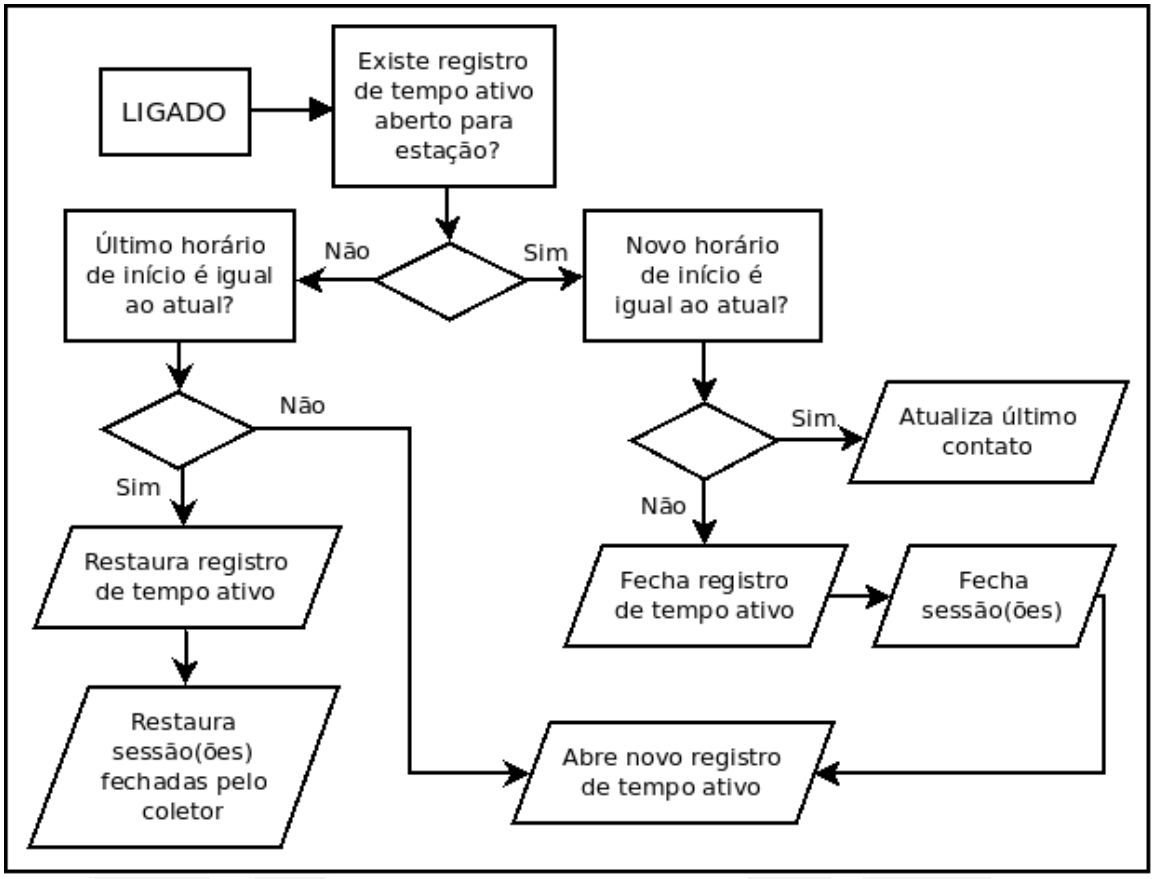


Figura 18 Fluxograma que trata o evento da situação LIGADO

Caso um registro com a situação DESLIGADO seja inserido na tabela *eventos* o algoritmo verifica se existe um registro de tempo ativo aberto, caso sim o finaliza, e em seguida verifica se existem sessões abertas associadas à este tempo ativo, e as também finaliza. Em todos os registros fechados o campo *geradoColetor* é marcado como TRUE, este campo serve para indicar que as alterações foram geradas por uma detecção do coletor, gerado pelo evento DESLIGADO e não em decorrência de um evento LIGADO como apresentado no tratamento deste evento. Nas duas situações se as verificações são negativas nada é feito conforme visualizado no fluxograma da Figura 19.

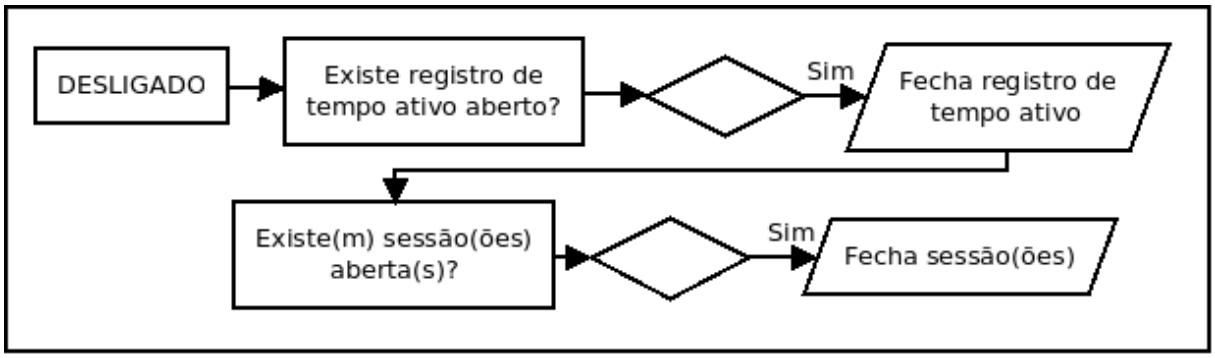


Figura 19 Fluxograma que trata o evento da situação DESLIGADO

Caso um registro possua a situação de LOGIN ou LOGOUT a ação a ser tomada é direta, visto que os eventos de LOGIN e LOGOUT são únicos e específicos. Caso a situação seja LOGIN, uma sessão é aberta com a identificação do tempo ativo da estação, caso seja LOGOUT, a sessão respectiva da máquina e usuário em questão é finalizada. Tais situações são visualizadas na Figura 20.

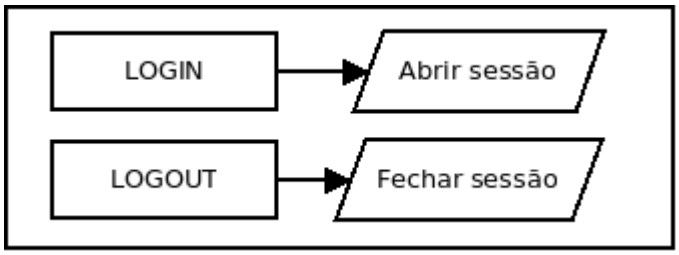


Figura 20 Fluxograma que trata os eventos da situação LOGIN e LOGOUT

3.2.3 Tabelas para informações de memória

Para o armazenamento das informações de memória adquiridas das estações monitoradas uma nova tabela foi adicionada à estrutura da base de dados, conforme representado na Figura 21.

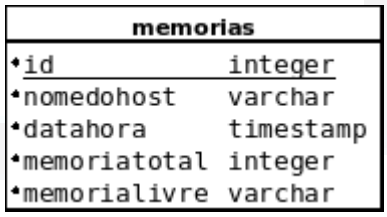


Figura 21 Nova tabela para armazenar as informações de memória

A tabela *memorias* armazena as informações referentes aos dados de memória total disponível e livre obtidos pelo sistema SDMR das estações monitoradas. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *nomeDoHost*: registro de texto que identifica o nome da estação da qual foi coletada a informação.
c) *dataHora*: armazena a data e hora no formato timestamp do momento que a informação foi capturada.
d) *memoriatotal*: armazena o valor total de memória do sistema.
e) *memorialivre:* armazena o valor de memória livre.

O cruzamento entre as informações de memória total e memória livre permitem a obtenção do montante de memória sendo ocupada na estação monitorada.

3.2.4 Tabelas para informações de cursos

Para que seja possível relacionar as informações de tempo de utilização com os cursos é necessária a criação de uma nova tabela que armazene as informações referentes aos diversos cursos e sua relação com cada usuário. A tabela esta representada na Figura 22.

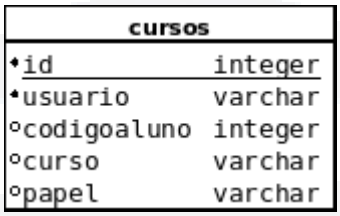


Figura 22 Nova tabela para armazenar as informações de cursos

A tabela *cursos* armazena as informações referentes ao curso atribuído a cada usuário. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *usuario*: registro que contem o nome de usuário ou login para associação com o campo curso.
c) *codigoaluno*: armazena o código de aluno associado ao usuário.
d) *curso*: armazena a descrição do curso associado ao usuário.
e) *papel:* armazena a descrição do papel associado ao usuário. Este atributo pode ser utilizado para um nível maior de filtragem, que permite agrupar os usuários em função do papel exercido, que poderia ser Aluno, Professor, Funcionário.

A tabela cursos é utilizada para gerar relatórios do tempo de utilização por determinado curso, este relatório será apresentado na reestruturação do console WEB.

3.3 Reestruturação do componente Agente

A reestruturação do componente agente, que é executado nas estações monitoradas, envolve a captura e publicação na MIB das seguintes informações:

a) Momento da inicialização da estação;
b) Eventos de sessões de usuário;

c) Utilização total de memória;

Neste capítulo é apresentada a implementação do agente para suportar estas funcionalidades, assim como a criação de um campo de controle que indica se o agente está em funcionamento ou parou abruptamente. O agente foi reestruturado para os dois sistemas operacionais suportados: Windows e Linux.

3.3.1 Captura da inicialização do sistema em Linux e Windows

No sistema operacional Linux a informação da data e hora de inicialização é obtida do arquivo de sistema */proc/stat*. A entrada que contém a informação desejada é *btime*, esta entrada contém o valor absoluto em segundos desde a data de 01 de Janeiro de 1970. A informação é obtida conforme apresentado no Capítulo 2, sendo convertida para o formato padrão de data e hora do sistema SDMR "YYYYMMDD HHMMSS".

Em Windows a informação é obtida da mesma forma, visto que o agente roda sobre o ambiente CygWin, que possui uma coleção de ferramentas em software livre para portar a execução de softwares que rodam em sistemas POSIX, como o Linux, sobre o sistema operacional Windows.

Após a captura da informação, a mesma é publicada na MIB, através de uma rotina específica para inserção de informações na estrutura da tabela de eventos inicializada no serviço SNMP da estação monitorada.

3.3.2 Captura das sessões em Linux

A captura das informações de sessões no ambiente Linux é feita através de leitura e análise do arquivo de *log wtmp* localizado em */var/log/wtmp*. O *wtmp* é arquivo binário composto por uma sequência de entradas que representam as informações referentes aos eventos de entrada e saída de usuários nos diversos terminais disponíveis, assim como eventos de inicialização e desligamento da estação.

Os referenciais apresentados no Capítulo 2 discutem também a utilização do Syslog para obtenção destas informações, no entanto para a implementação deste trabalho este método de obtenção foi descartado por necessitar da criação de filtros em arquivo de configuração e de uma análise de *log* buscando padrões de entrada de informação, que pode variar dependendo de qual software a tenha gerado, eliminando assim a dependência destes quesitos.

O arquivo *wtmp*, apesar de possuir uma estrutura com algumas exceções, permite através do descarte de alguns dados a obtenção das informações de sessões de usuário de uma

forma mais direta. Conforme já apresentado no Capítulo 2, são várias as informações contidas em cada entrada inserida dentro do arquivo *wtmp*, dentre as quais serão utilizadas as listadas abaixo:

a) *short ut_type*: representa o tipo de registro inserido, pode variar de 0 a 9, sendo que o tipo 7 representa o início de utilização de um *device*, e o número 8 o fim.
b) *char ut_line[]*: armazena o nome do *device* do terminal utilizado.
c) *char ut_user[]*: armazena o nome do usuário.
d) *struct timeval ut_tv*: armazena o valor em segundos a partir do dia 01 de Janeiro de 1970 do momento em que o evento ocorreu.

Na Tabela 7 abaixo pode-se visualizar diversas informações obtidas através da leitura do arquivo *wtmp*, dentre elas as necessárias para controle da entrada e saída dos usuários.

Tabela 7 Tabela com informações obtidas na leitura do arquivo wtmp.

Registro	USER	TYPE	TIME	HOST	PID	DEVICE
1	eduardod	7	1288778863	:0	1979	tty7
2	eduardod	7	1288786001	:0.0	3840	pts/0
3	teste	7	1288786012		3881	pts/0
4	teste	7	1288786013		3941	pts/0
5		8	1288786015		3881	pts/0
6	eduardod	8	1288786018		0	pts/0
7	eduardod	7	1288786020	:0.0	3976	pts/0
8	teste	7	1288789250	localhost	4731	pts/1
9	eduardod	7	1288789258	eduardod-904-32.lo	4814	pts/2
10		8	1288789259		4814	pts/2
11		8	1288789263		4731	pts/1

Observando a tabela pode-se constatar algumas ponderações que devem ocorrer quando em processo de análise dos registros. Os registro 2, 3 e 4 representam a entrada e saída do usuário *teste* no *device pts/0*, nota-se neste caso que houve uma duplicação de entrada nos registros 2 e 3. Um destes registros deve ser descartado, sendo assim, considera-se que o usuário *teste* entrou no tempo do registro 3 no *device pts/0*, o registro quatro é descartado, e o registro 5 indica a finalização da sessão deste usuário.

Outro fato a ser observado com atenção pode ser constatado nos registros 5, 10 e 12, eles representam a saída de usuário, no entanto não informam qual o usuário que efetuou tal saída. Para que seja possível fazer a distinção é necessário analisar o *device* ao qual o registro

está associado para desta forma distinguir qual sessão está sendo encerrada. Esta situação pode ser observada nos registros 8, 9, 10 e 11. No registro 8 o usuário *teste* dá início a uma sessão remota no *device pts/1*, e no registro 9 o usuário *eduardod* inicia outra sessão remota no *device pts/2*. Em seguida, observa-se o registro 10 do tipo 8, indicando que houve a saída de um *device*, no entanto não informa qual o usuário. Então analisando o *device* associado conclui-se que o registro pertence ao usuário *eduardod* por estar relacionado ao *device pts/2*.

Além disso pode-se observar que o sistema operacional Linux permite a utilização de diversos terminais simultaneamente, e desta forma diversas entrada em *devices* diferentes são inseridas no arquivo *wtmp* e devem ser tratadas para que seja possível detectar quando um usuário efetivamente termina sua sessão em uma determinada estação.

Para controlar em tempo de execução os diversos *devices* utilizados pelos usuários foi necessária a criação de uma estrutura que armazene estas "entradas" e que permita definir quando um usuário não está mais utilizando nenhum terminal, efetuando assim sua saída da estação e a finalização de sua sessão.

Para este controle duas estruturas foram criadas: a *struct entrada* e a *struct sessao*. A *struct entrada* armazena as diversas entradas do usuário nos terminais, é instanciada em forma de lista encadeada e é composta pelos atributos:

a) *char *line*: armazena a qual *device* o registro está associado.
b) *unsigned long time*: armazena o tempo que o registro ocorreu.
c) *struct entrada *proximo*: ponteiro para uma *struct* entrada, apontando aqui para uma possível próxima entrada.

A *struct sessao* armazena as sessões abertas na estação, assim como os dados necessários para a geração dos eventos de entrada e saída de um usuário. É instanciada como uma lista encadeada e é composta pelos seguintes atributos:

a) *char *usuario*: armazena o nome de usuário da sessão.
b) *int numEntradas*: variável que controla o número de entrada do usuario.
c) *struct sessao *proximo*: aponta para uma possível próxima sessão de usuário.
d) *struct entrada *listaDeEntradas*: ponteiro para a lista de entradas em terminais do usuário, permite desta forma controlar que *device* foi aberto ou fechado e determinar quando o usuário deixou a estação.

Na Figura 23 são visualizadas as duas estruturas criadas para controle das sessões no sistema operacional Linux, e o instanciamento da lista de sessões.

```
struct entrada{
    char            *line;
    unsigned long   time;
    struct entrada  *proximo;
};

struct sessao{
    char        *usuario;
    int         numEntradas;
    struct      sessao *proximo;
    struct      entrada *listaDeEntradas;
};

struct sessao *listaDeSessoes = (struct sessao*) NULL;
```

Figura 23 Estruturas de controle das sessões de usuários no sistema operacional Linux

3.3.3 Captura das sessões em Windows

No sistema operacional Windows o controle das sessões de usuários é feita de forma menos criteriosa em relação ao sistema operacional Linux. O controle é feito através da análise dos processos ativos e dos respectivos usuários associados a estes processos. Uma estrutura de controle *struct sessao* é utilizada e permite armazenar as sessões de usuários ativos, e determinar o encerramento de uma sessão quando determinado usuário não possuir mais processos associados ao seu nome de usuário.

3.3.4 Captura da utilização de memória em Linux e Windows

A captura das informações de memória é feita de forma idêntica nos sistemas operacionais Windows e Linux. A captura se baseia na leitura das informações do estado de memória do arquivo /proc/meminfo, este arquivo está presente nativamente somente no sistema operacional Linux. Porém, como o agente Windows roda sobre a plataforma CygWin, que emula o ambiente Linux sobre o Windows, é possível obter as informações da mesma forma nos dois sistemas. Caso um trabalho futuro tenha por objetivo não utilizar a plataforma CygWin será necessário rever a implementação de memória no sistema Windows.

As informações obtidas no arquivo /proc/meminfo e utilizadas para determinar a utilização da memória pelos diversos processos do sistema são:

a) MemTotal: representa o valor em Kilobytes do total de memória disponível ao sistema.
b) MemFree: representa a quantidade de memória não utilizada.
c) Buffers: representa a quantidade de memoria utilizada em *buffers*.

d) Cached: representa a quantidade de memória utilizada em *cache*.

O total de memória livre é determinado pela soma dos valores de MemFree, Buffers e Cached.

3.3.5 Informação de controle de agente ativo

Para que seja possível determinar se o agente está rodando em uma estação sem ser necessário varrer todas as tabelas de informações, foi criada uma variável de controle, denominada *Agente Ativo*, que é publicada na MIB de forma a disponibilizar a consulta de um valor constante. A consulta positiva deste valor implica na conclusão que o agente está rodando na estação monitorada, visto que a manutenção da variável depende da execução do agente.

O valor constante publicado é do tipo inteiro e é utilizado também para determinar qual versão do agente está rodando. Possibilita a detecção de agentes em execução em versão inferior e não efetua a coleta das informações destes, evitando assim o armazenamento de dados que possam trazer futuras conclusões errôneas. Na reestruturação do componente Coletor é apresentada a utilização desta variável de controle e sua utilidade para manter o sistema num estado mais robusto.

3.3.6 Parametrização do arquivo de configuração do Agente

Com a complementação do sistema foi necessário reestruturar o arquivo de configuração do agente, adicionando os parâmetros de tempo de captura de eventos, que representam as entradas e saídas das sessões de usuários e o tempo de captura das informações de memória. Na Figura 24 é ilustrado o arquivo de configuração do agente com os parâmetros "*agtTempoDeCapturaParaEvento*" e "*agtTempoDeCapturaParaMemoria*" já adicionados.

```
# Arquivo de configuracao do agented

#1 captura todos os processos ativos, 0 captura somente processos com uso CPU > 0
agtTipoCaptura = 1

# Configuracao do agente
agtTempoDeExecucaoDoLaco = 10
agtTempoDeCapturaParaProcesso = 20
agtTempoDeCapturaParaTemperatura = 480
agtTempoDeCapturaParaParticao = 180
agtTempoDeCapturaParaEvento = 10
agtTempoDeCapturaParaMemoria = 30

#agtProcTemperatura = /proc/acpi/thermal_zone/TZ01/temperature

# 1 exibe noticias
agtExibeNoticia = 1
```

Figura 24 Arquivo de configuração do agente

Os parâmetros de configuração presentes no arquivo são:

a) *agtTipoCaptura*: indica o tipo de captura a ser realizada. A configuração do valor 0 habilita a captura somente dos processos que tiveram algum tipo de processamento durante o intervalo de captura das informações. Já o valor 1 habilita a captura de todos os processos ativos no sistema.
b) *agtTempoDeExecucaoDoLaco*: especifica o intervalo de tempo em segundos entre as capturas de informações do sistema. Este valor deve ser igual ou inferior ao menor tempo de captura informado nos tempos de captura.
c) *agtTempoDeCapturaParaProcesso*: especifica o intervalo de tempo em segundos entre as capturas das informações de processos.
d) *agtTempoDeCapturaParaTemperatura*: especifica o intervalo de tempo em segundos entre as capturas das informações de temperatura do processador.
e) *agtTempoDeCapturaParaParticao*: especifica o intervalo de tempo em segundos entre as capturas das informações de partições do disco rígido.
f) *agtTempoDeCapturaParaEvento:* especifica o intervalo de tempo em segundos entre as capturas das informações de eventos de sessões e tempo ativo.
g) *agtTempoDeCapturaParaMemoria*: especifica o intervalo de tempo em segundos entre as capturas das informações de utilização de memória.
h) *agtProcTemperatura*: em Linux indica o caminho para obtenção da informação da temperatura do processador. Em Windows a captura é feita por uma consulta genérica.
i) *agtExibeNoticia*: indica se informações de notícia devem ser exibidas, é utilizado para depuração do sistema. O valor 1 habilita a exibição, enquanto o valor 0 desabilita.

Utilizando o caractere # é possível adicionar comentários ou desabilitar uma determinada configuração.

3.4 Reestruturação do componente MIB

A reestruturação do componente MIB envolveu a criação das novas tabelas e sua incorporação na estrutura da MIB *tcc*, armazenada em disco no arquivo "TCC-MIB.txt". A criação das novas tabelas permite que as novas informações de eventos e utilização de

memória sejam publicadas pelo agente na estrutura SNMP e fiquem disponível para coleta do componente coletor.

Na Figura 25, a MIB *tcc* é visualizada em árvore, e já possui as complementações das tabelas *tccEventTable* para armazenagem dos eventos de tempo ativo e sessões de usuário, e *tccMemoryTable* para armazenagem das informações de utilização de memória.

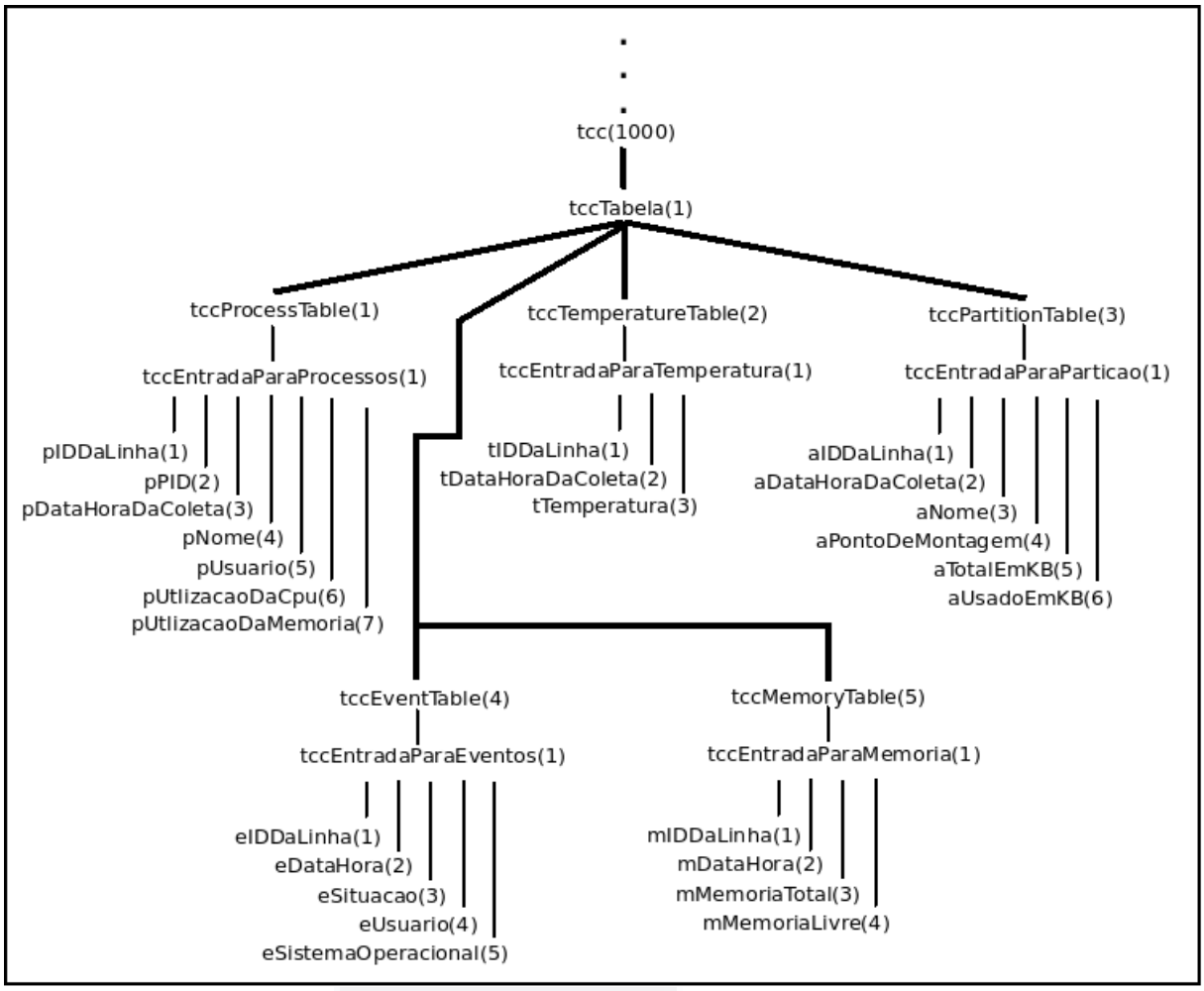


Figura 25 Visualização em árvore da MIB *tcc* reestruturada

3.5 Reestruturação do componente Coletor

Para que as novas informações disponibilizadas pelos agentes nas estações monitoradas pudessem ser coletadas e armazenadas no banco de dados foi necessária a reestruturação do componente coletor, assim como adaptações no seu processo de coleta buscando torná-lo mais prático e efetivo. As principais implementações são:

a) Coleta das informações de eventos das estações monitoradas;
b) Coleta das informações de memória das estações monitoradas;
c) Obtenção do evento DESLIGADO;

d) Obtenção da lista de computadores da base dados;
e) Instanciamento de vários coletores através de parâmetro passado para o coletor;
f) Análise de início de coleta baseado na obtenção da informação de *Agente Ativo* publicada na MIB das estações monitoradas;
g) Implementação dos parâmetros de configuração RETRY e STANDBY, que permitem ao processo de coleta guardar o estado das estações e pular ciclos de coleta;
h) Tabela de *avisosistema*, que armazena avisos gerados pelo coletor em relação as estações monitoradas.

A seguir será apresentada uma análise geral do processo de coleta original do sistema SDMR a fim de justificar as modificações realizadas na reestruturação do componente coletor que serão descritas mais adiante.

3.5.1 Análise do formato original do processo de coleta

No formato original do sistema SDMR o coletor ao iniciar o processo de coleta realizava a leitura das estações a serem monitoradas do arquivo de configuração e em seguida passava a percorrê-las em ciclos de tempo buscando obter as informações publicadas em suas MIBs *tcc*, sendo que nenhum controle era feito para detectar se estações estavam desligadas ou não. No caso de uma tentativa de coleta à uma estação desligada o tempo gasto até o *timeout* é de aproximadamente 4 segundos. Num cenário simulado com todas as estações ligadas, o coletor utiliza somente o tempo necessário para coleta, que varia dependendo do número de informações publicadas. Porém num ambiente real onde as estações possam estar desligadas o tempo de coleta fica restrito ao tempo gerado pela tentativa de coleta das estações desligadas.

Por exemplo, num ambiente com 30 estações, onde 29 estão desligadas e somente 1 está ligada, o tempo gasto no ciclo de coleta é no mínimo de 4 segundos por cada uma das 29 estações, num total de 116 segundos. Então a estação ligada teria suas informações coletadas em ciclos de tempo de no mínimo 116 segundos, independentemente do valor do ciclo de coleta especificado no arquivo de configuração do coletor.

Em vista deste cenário algumas modificações foram realizadas no processo de coleta e configuração do coletor. Estas modificações são apresentadas a seguir.

3.5.2 Coleta das novas informações

A coleta das novas informações implementadas no sistema SDMR compreende a obtenção dos dados referentes a eventos e memória publicados na MIB *tcc* das estações monitoradas. As rotinas para coleta das informações foram baseadas nas funções já existentes na versão original do SDMR e, desta forma, utilizam o mesmo processo de solicitação dos dados via protocolo SNMP e posterior armazenamento na base de dados, nas tabelas que foram especificamente criadas para este fim.

3.5.3 Detecção do evento DESLIGADO

O evento DESLIGADO é obtido de forma direta quando não houver sucesso na tentativa de coleta das informações publicadas na MIB das estações monitoradas. O processo de tentativa de coleta é iniciado somente se a tentativa de coleta da variável de "*Agente Ativo*" publicada pelo agente for realizada com sucesso, em caso negativo indica que para o sistema SDMR aquela estação está desligada e não pode ser coletada.

3.5.4 Modificação no processo de obtenção da lista de computadores monitorados

Esta modificação está relacionada com a gerência dos computadores monitorados, as informações relativas aos computadores ficam armazenadas no banco de dados e sua gerência passa a ser feita via console WEB.

Para isso a tabela que armazena os computadores foi modificada e 3 novas tabelas foram adicionadas a estrutura do banco de dados para agrupar estas estações em laboratórios e prédios, e ainda pela definição do coletor ao qual está associado sua coleta. Desta forma existe uma tabela de coletores, e cada computador está associado a um destes coletores. Ao iniciar o processo de coleta, o coletor conhece seu ID e obtêm da base de dados as máquinas sob sua responsabilidade de coleta, para então iniciar o processo de coleta das informações.

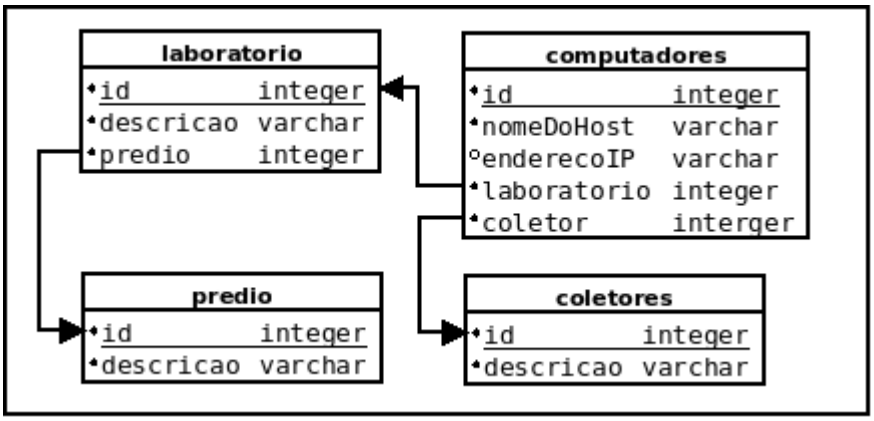


Figura 26 Tabelas predio, laboratorio, computadores e coletor

A tabela *predio* armazena as informações referentes aos prédios cadastrados. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *descricao:* registro de texto que contém a descrição do prédio.

A tabela *laboratorio* armazena as informações referentes aos laboratórios cadastrados. Cada laboratório está agrupado à um prédio da tabela *predio*. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *descricao:* registro de texto que contém a descrição ou identificação do laboratório.
c) *predio:* chave estrangeira da tabela *predio.* Indica em qual prédio o laboratório está localizado.

A tabela *coletores* armazena as informações referentes aos coletores cadastrados. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *descricao:* registro de texto que contém a descrição ou identificação do coletor.

A tabela *computadores* armazena as informações dos computadores monitorados, foi modificada para ser relacionada com as tabelas *predio* e *laboratorio*. Os campos que a compõem são:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *nomeDoHost:* identifica o nome do computador.
c) *enderecoIP:* identifica o endereço IP do computador.
d) *laboratorio:* chave estrangeira que relaciona o computador à um determinado laboratório.
e) *coletor*: chave estrangeira da tabela coletores. Indica o coletor ao qual o computador está associado.

3.5.5 Identificação e instanciamento de vários coletores

Como já apresentado, cada computador está associado a um coletor, sendo assim um coletor possui um grupo de computadores sob sua responsabilidade, conforme configurado pelo administrador via console WEB.

Para que vários coletores sejam executados cada um deles deve conhecer seu ID para que seja possível obter as informações dos computadores da base de dados e iniciar o processo de coleta das informações. Na modificação implementada, quando um coletor for executado via linha de comando, deve ser passado por parâmetro seu identificador, o coletor ao receber este parâmetro conhece seu ID e pode dar início ao processo de coleta.

Ponderações sobre a utilização de vários coletores são feitas com mais detalhes no Capítulo 4 que apresenta os resultados obtidos, onde são feitas análises de impacto desta configuração.

3.5.6 Parâmetros de configuração *RETRY* e *STANDBY*

Na parametrização do arquivo de configuração do coletor duas novas métricas foram adicionadas, uma referente a quantidade de tentativas que devem ser feitas antes de atribuir a um computador o estado de "em espera" ou como denominado *STANDBY*. Neste estado a tentativa de coleta de informações não é feita para o computador.

A outra métrica está relacionada a quantidade de ciclos que devem ocorrer para que uma nova tentativa de coleta seja feita para um computador em estado de espera, este atributo é denominado *RETRY*.

Desta forma o estado de cada estação a ser monitorada é guardado pelo coletor em tempo de execução. O estado de cada estação possui atributos para identificar se ela está em estado de espera e por quantos ciclos ela permanece assim, para que desta forma seja possível determinar o momento de uma nova tentativa de coleta, caso seja realizada com sucesso o computador é removido do estado de espera e passa a ser coletado normalmente.

Na parametrização do arquivo de configuração do coletor são apresentados os atributos a serem configurados para que os valores de RETRY e STANBY sejam definidos.

3.5.7 Tratamento da variável de controle de "Agente Ativo"

A variável de controle denominada de "*Agente Ativo*" é instanciada na MIB do agente e tem por objetivo fornecer ao coletor um mecanismo para identificar se o agente está rodando e está em estado funcional.

Caso a estação esteja apta para ser coletada o coletor inicia o processo verificando se a variável de controle pode ser obtida da MIB do agente. Em caso positivo verifica se o valor retornado é igual ao valor de controle atual do sistema.

Desta verificação 4 possíveis conclusões são feitas pelo coletor:

a) Estação desligada: Não é possível obter uma resposta do serviço SNMP rodando na estação monitorada, o sistema SDMR assume que a estação está desligada.

b) Estação ligada, mas agente não está rodando: É possível obter uma resposta do serviço SNMP, porém não é possível coletar o valor armazenado na variável de controle. O sistema SDMR assume que o agente está em estado não funcional e insere um aviso de sistema no banco de dados informando esta situação.

c) Estação com versão inferior: A obtenção do valor da variável de controle é realizada com sucesso, porém o valor retornado é inferior ao valor atual armazenado localmente que serve de comparação para determinar a versão do agente que está rodando. O sistema SDRM assume que o agente está rodando em uma versão inferior e insere um aviso de sistema no banco de dados informando esta situação.

d) Estação com agente rodando: A variável de controle é obtida com sucesso e o valor retornado corresponde ao valor atual armazenado no sistema. É iniciado então o processo de coleta da estação.

As informações geradas pelo coletor são inseridas na base de dados na tabela *avisosistema*, criada especificamente para armazenar estes avisos. Na Figura 27 é visualizada a tabela *avisosistema* e seus campos.

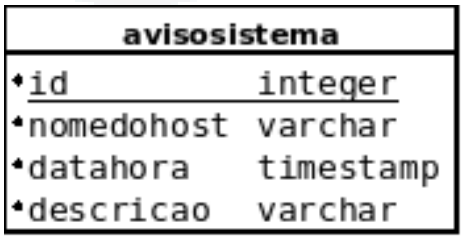


Figura 27 Tabela *avisosistema*

A tabela *avisosistema* é composta pelos campos:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.

b) *nomeDoHost*: registro de texto que identifica o nome da estação da qual o

aviso foi gerado.

c) *dataHora*: armazena a data e hora no formato timestamp do momento que a situação que gerou o aviso ocorreu.

d) *descricao*: campo texto que contém a descrição do aviso.

3.5.8 Parametrização do arquivo de configuração do Coletor

Com a complementação do coletor do sistema SDMR foi necessário reestruturar também o seu arquivo de configuração. Foram removidos os parâmetros:

a) *cltIPsParaCaptura:* que definia a lista dos computadores, informando o nome e seu endereço IP, do qual o coletor deveria obter informações. Foi removido em função da especificação dos computadores ser obtida pelo coletor em consulta na base de dados.

b) *cltSOPorHost*: que definia a lista contendo o sistema operacional de cada computador. Foi removido em função da implementação da obtenção do sistema operacional em tempo de execução, onde o agente publica o sistema operacional atual juntamente com as informações de eventos de tempo ativo e sessões.

c) *cltGravaListaNoDB*: que especificava se a lista de computadores contida em *cltIPsParaCaptura* deveria ser gravada na base de dados. Foi removido em virtude da gerência ser toda feita pelo console *WEB*.

Em contrapartida foram adicionados os seguintes parâmetros:

a) *cltQtdeStandByColeta:* que especifica a quantidade de vezes que um computador deve ser detectado como desligado para ser colocado em estado de espera. Neste estado a estação passa a não ser mais apta a coleta, a não ser que esteja num ciclo de nova tentativa definido pelo parâmetro *cltQtdeRetryColeta*.

b) *cltQtdeRetryColeta:* que especifica em quantos ciclos irá ocorrer uma nova tentativa de coleta de uma estação em estado de espera ou *STANDBY.*

O arquivo de configuração reestruturado pode ser visualizado na Figura 28.

```
# Arquivo de configuracao do coletord

# Conexao com o agente
snmpComunidade = public
snmpPorta = 161

# Parâmetros de STANDBY e RETRY
cltQtdeStandByColeta = 1
cltQtdeRetryColeta = 3

# Conexao com o servidor da base de dados
bdHost = localhost
bdNome = tcc
bdUsuario = postgres
bdPassword = sdmr
#bdPorta = 5432

# Intervalo do ciclo de tempo entre as coletas
cltTempoDeCaptura = 45

# 1 exibe noticias
cltExibeNoticia = 0
```

Figura 28 Arquivo de configuração do coletor

Os parâmetros de configuração presentes no arquivo são:

a) *snmpComunidade*: define a comunidade usada para acessar os agentes SNMP, ela identifica os níveis de privilégios no acesso às informações.
b) *snmpPorta*: define a porta na qual o SNMP irá responder, por padrão a porta utilizada é a 161.
c) *cltQtdeStandByColeta*: define a quantidade de tentativas falhas na coleta de uma estação para que ela entre em estado de espera.
d) *cltQtdeRetryColeta*: define o quantidade de ciclos que devem ocorrer para que uma nova tentativa de coleta seja feita à uma estação em estado de espera.
e) *bdHost*: define o endereço IP onde o banco de dados está instalado.
f) *bdNome*: define o nome da base de dados que contém as tabelas do sistema SDMR.
g) *bdUsuario*: define o usuário para acesso à base de dados.
h) *bdPassword*: define a senha de acesso à base de dados.
i) *bdPorta*: define a porta de acesso à base de dados.
j) *cltTempoDeCaptura*: define o tempo de espera entre os ciclos de coleta, isto é, entre uma coleta e outra.

k) *agtExibeNoticia*: indica se informações de notícia devem ser exibidas, utilizado para depuração do sistema. O valor 1 habilita a exibição, enquanto o valor 0 desabilita.

Utilizando o caractere # é possível adicionar comentários ou desabilitar uma determinada configuração.

3.6 Reestruturação do componente Console WEB

As modificações realizadas no Console WEB compreendem a reestruturação de sua interface, disponibilizando telas para análise dos novos dados coletados, assim como dos dados que já eram coletados pelo sistema SDMR. As telas implementadas foram agrupadas em 5 categorias buscando organizá-las por funcionalidade, que são: Geral, Cadastros, Relatórios, Consultas e Alertas. A estrutura visual da página foi totalmente modificada para que fosse possível o agrupamento das opções.

O desenvolvimento do console WEB foi realizado com as mesmas ferramentas utilizadas no trabalho de Stoll(2008). A linguagem de programação utilizada foi JSP[1] (*Java Server Pages*). Como ambiente de desenvolvimento para codificação e execução de testes foi utilizado o Netbeans[2]. Já como servidor de páginas para rodar a aplicação *WEB* foi optado pela aplicação GlassFish[3]. Todas as ferramentas utilizadas são gratuitas e de código fonte aberto.

A seguir são apresentadas as modificações realizadas no console WEB: a nova estrutura visual da tela, as complementações no controle de acesso, e cada uma das opções do menu principal e seus sub-menus, que compreendem as funcionalidades de análise dos dados.

3.6.1 Estrutura visual

A estrutura visual do console WEB foi reestruturada buscando agrupar as opções oferecidas pelo sistema em categorias. Cada categoria engloba um conjunto de opções afins que dão acesso às opções de interação ou visualização das informações.

O menu de opções que engloba as 5 categorias foi posicionado na parte superior da tela e o acesso aos sub-menus é feito posicionando o cursor do mouse sobre uma opção do menu, a sexta opção é para saída do sistema.

---

1 JSP (*Java Server Pages*) – tecnologia utilizada para desenvolvimento de aplicações *WEB* baseada na linguagem de programação Java. Mais informações em: <http://java.sun.com/products/jsp/> e

2 Netbeans - ferramenta que oferece um ambiente de desenvolvimento integrado com suporte a criação de aplicações *WEB*, como JSP. Mais informações em: <http://netbeans.org>

3 GlassFish – servidor de aplicações *WEB* que suporta a execução de aplicações JSP. Mais informações em: <https://glassfish.dev.java.net/>

A Figura 29 ilustra a tela principal do sistema, onde é possível visualizar o menu de opções, assim como o mouse posicionado sobre a opção Cadastros e as opções do sub-menu disponibilizado, que abrange as telas de cadastros do sistema.

Figura 29 Visualização da tela principal do console WEB

3.6.2 Controle de acesso

Buscando manter o acesso ao console WEB restrito foi mantida a utilização de uma tela de autenticação onde o usuário deve informar seu nome de usuário e sua senha de acesso para ser redirecionado para a tela principal do sistema.

Para proporcionar um nível de segurança maior ao acesso do sistema duas modificações foram realizadas no processo de autenticação. A primeira é o armazenamento da informação de senha em formato MD5[4]. A utilização deste algoritmo traz um nível de segurança superior, não permitindo que as informações de senha sejam visualizadas em texto claro.

A segunda modificação é a implementação do controle de acesso ao console WEB utilizando o recurso de sessões. Esta funcionalidade garante que sites internos não sejam acessados sem que o usuário não tenha se autenticado de forma correta. A tentativa de acesso a uma página interna sem a autenticação ocasiona o redirecionamento para a tela de autenticação, ilustrada na Figura 30.

Figura 30 Visualização da tela de autenticação do console WEB

[4] MD5 do inglês Message-Digest algorithm 5, é um algoritmo hash de 128 bits unidirecional descrito na RFC 1321 e desenvolvido pela RSA Data Security, Inc., podendo ser utilizado para criptografar informações de texto, com senhas de usuários.

3.6.3 Menu Geral

A entrada “Geral” do menu principal dá acesso a um sub-menu com opções de análise de informações de forma mais generalizada, como visualizar máquinas ligadas, sessões de usuários ou informações estatísticas do sistema. Abaixo são apresentadas as opções do sub-menu acessado através da entrada “Geral”:

a) Mapa de computadores;
b) Estatística Geral;
c) Computadores ligados;
d) Sessões abertas;
e) Avisos do sistema.

A seguir cada uma das opções é explicada de forma individual, ilustrando as telas de acesso as informações e atributos que podem ser modificados para gerar resultados diferentes.

3.6.3.1 Menu Geral - Mapa de computadores

A opção “Mapa de computadores” da entrada “Geral” do menu principal dá acesso à uma lista de computadores e estatísticas gerais de uso das últimas informações coletadas de cada estação monitorada pelo sistema.

A visualização do mapa pode ser personalizada pela escolha de dois atributos que filtram a consulta de informações, que são:

a) Seleção do estado dos computadores:
   - Somente Ligados: irá gerar o mapa somente com computadores que estão ligados no momento.
   - Ligados e desligados: irá gerar o mapa com todos os computadores cadastrados no sistema, independente da estação estar ligada ou desligada.
b) Seleção do filtro de consulta:
   - Todos os computadores: irá gerar o mapa com todos os computadores cadastrados na base de dados.
   - Somente Laboratório: permite gerar um mapa listando somente computadores de um determinado laboratório.

O cruzamento da seleção dos dois atributos permite gerar 4 estados de mapa de computadores diferentes:

a) Somente Ligados + Todos os computadores: exibe todos os computadores ligados e cadastrados no sistema.

b) Somente Ligados + Somente Laboratório: exibe os computadores ligados de um determinado laboratório.

c) Ligados e desligados + Todos os computadores: mais abrangente, exibe todos os computadores cadastrados, independente do estado de desligado ou ligado.

d) Ligados e desligados + Somente Laboratório: Exibe os computadores de um determinado laboratório, independente do estado de desligado ou ligado.

Na Figura 31 é ilustrado o mapa de computadores gerados selecionando o estado "Ligados e desligados" e o filtro "Somente Laboratório" com a opção "101-11", que compreende o laboratório de informática da sala 101 do prédio 11.

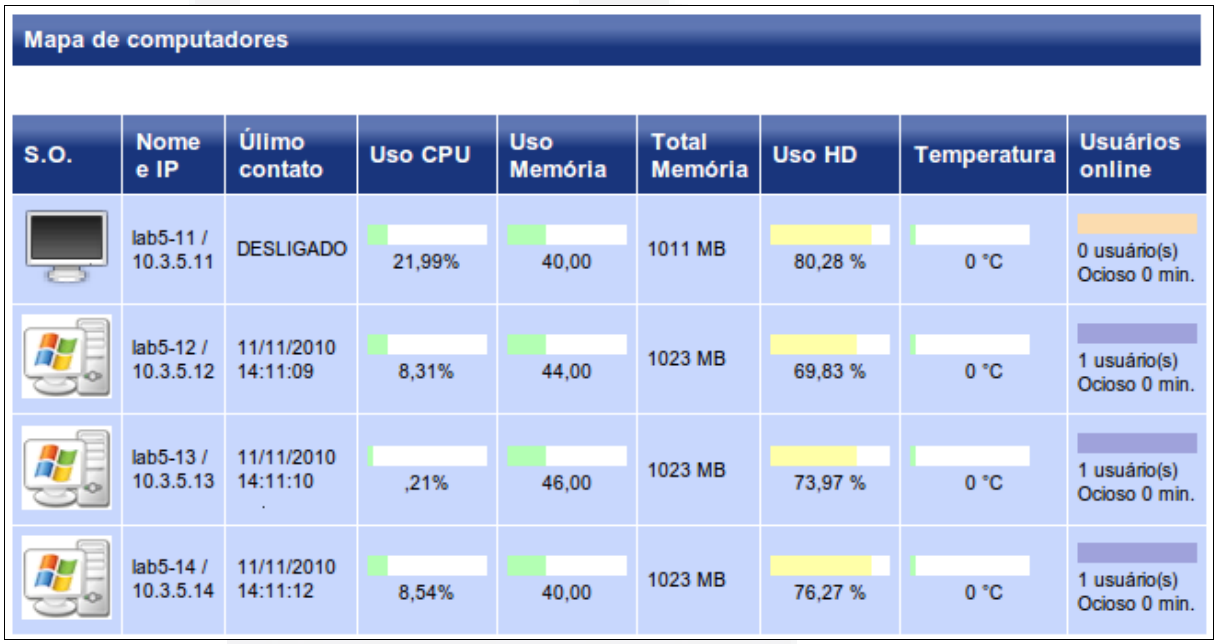


Figura 31 Exemplo de um Mapa de computadores do console WEB

Conforme ilustrado na Figura 31 o mapa de computadores é apresentado na forma de lista, diferentemente do mapa original implementado por Stoll(2008). A mudança foi feita para permitir que fosse possível visualizar informações semelhantes, como a taxa de uso do HD, numa mesma coluna, podendo fazer a comparação visual de forma mais adequada.

Cada linha compreende um computador, onde são apresentadas as informações:

a) Coluna S.O. - a informação do sistema operacional é apresentada em forma de imagem para facilitar a distinção visual, caso o computador esteja desligado uma imagem específica para este estado é apresentada.

b) Coluna Nome e IP - esta coluna apresenta as informações do nome do computador e seu endereço IP.

c) Coluna Último contato – exibe a informação do último contato feito com o computador caso ele esteja ligado. Estando desligado a mensagem "DESLIGADO" é apresentada no campo.

d) Coluna Uso CPU – coluna que exibe o percentual de CPU utilizado pelos processos ativos no último registro de coleta feito da estação. Neste campo é apresentado o valor numérico do uso de recurso e uma barra indicando através de cores o nível de utilização.

e) Coluna Uso memória - coluna que exibe o percentual de memória utilizado no sistema no último registro de coleta feito da estação. Neste campo é apresentado o valor numérico do uso de recurso e uma barra indicando através de cores o nível de utilização. Na versão implementada por Stoll(2008) o percentual era calculado através da soma do uso de memória por todos os processos, neste caso se a opção de captura do agente fosse de somente processos com uso de CPU maior que zero esta informação não estaria representando o percentual total de uso. Na reestruturação do sistema o percentual total de memória, assim como seu uso foram agregados ao sistema, desta forma esta informação passou a ser apresentada neste campo.

f) Coluna Total Memória: coluna que apresenta o total de memória coletada pelo agente na estação monitorada.

g) Coluna Uso HD - coluna que exibe o percentual de utilização do HD no último registro de coleta feito da estação. Neste campo é apresentado o valor numérico do uso de recurso e uma barra indicando através de cores o nível de utilização.

h) Coluna Temperatura: coluna que exibe o valor da temperatura obtido no último registro de coleta feito da estação. Neste campo é apresentado o valor numérico da temperatura e uma barra indicando através de cores o nível de utilização.

i) Coluna Usuários Online: nesta coluna é informado o número total de usuários online na estação no momento da consulta. Além disso é apresentada a informação do total de minutos que o computador encontra-se sem ser utilizado. Caso esteja desligado ou em uso, o tempo resultante é zero.

Alguns dos campos possuem ligação com outras telas de informações sobre a estação. Clicando sobre os campos de “S.O.” ou “Nome e IP” da estação a página é redirecionada para uma tela de informações específicas do computador, onde podem ser visualizadas as listas de processos, partições e informações de memória.

Clicando sobre o campo “Usuários” a página é redirecionada para uma tela informando quais são os usuários onlines na estação, assim como informações dos tempos de entrada de cada um deles.

Eficiência das consultas do mapa de computadores

Para disponibilizar as informações visualizadas no mapa de computadores uma série de consultas com diferentes agrupamentos de dados são feitas no banco de dados. Nas experimentações realizadas por Stoll(2008) o tempo de resposta não era significativo em função da quantidade inferior de computadores monitorados. No entanto, com 216 computadores a serem monitorados ficou constatado um período de tempo relativamente alto para que o mapa fosse gerado.

A Tabela 8 apresenta os tempos obtidos sobre uma das consultas utilizadas no processo de geração do mapa de computadores. A consulta em formato SQL utilizada para obtenção dos tempos é ilustrada na Figura 32. Esta consulta retorna a soma de utilização de CPU de determinado computador, identificado aqui pela expressão “*$computador*”, em função do último registro de coleta encontrado.

```
SELECT SUM(utilizacaodacpu) FROM processos WHERE nomedohost='$computador' AND datahorafinal = (SELECT MAX(datahorafinal) FROM processos WHERE nomedohost='$computador')
```

Figura 32 Consulta SQL sobre a tabela *processos*

Foram obtidos os tempos em consultas realizadas em 4 cenários diferentes em função de índices criados na tabela *processos*. Os cenários são:

a) Cenário 1: as consultas foram realizadas sobre a tabela *processos* em sua configuração original, sem a criação de nenhum índice.
b) Cenário 2: apenas o índice sobre o campo *datahorafinal* da tabela *processos* foi criado.
c) Cenário 3: apenas o índice sobre o campo *nomedohost* da tabela *processos* foi criado.
d) Cenário 4: foram criados índices sobre os campos *datahorafinal* e *nomedohost* da tabela *processos*.

No momento da consulta a tabela *processos* possuía 3.807.309 registros, a quantidade de registros referente a cada computador também são apresentadas na Tabela 8.

Tabela 8 Informações de desempenho de consultas sobre a tabela *processos*

Computador	Quantidade de Registros	Tempo em Cenário 1	Tempo em Cenário 2	Tempo em Cenário 3	Tempo em Cenário 4
labserver	524.374	1.344,449 ms	2,669 ms	532,429 ms	2,564 ms
lab18-10	57.825	1.304,529 ms	2,513 ms	245,227 ms	2,597 ms
lab18-20	14.514	1.325,137 ms	2,444 ms	67,195 ms	2,547 ms
lab21-10	19.154	1.264,658 ms	78,106 ms	90,661 ms	77,067 ms
lab21-17	2.776	1.279,150 ms	230,008 ms	7,308 ms	262,148 ms
Média	**-**	**1303,58 ms**	**63,15 ms**	**188,56 ms**	**69,38 ms**
Projeção para 200 computadores	**-**	**260, 72 seg.**	**12,63 seg.**	**37,71 seg.**	**13,88 seg.**

As duas últimas linhas da tabela apresentam, respectivamente, o tempo médio para consulta de um computador baseado nos tempos obtidos em cada um dos cenários, e o tempo total projetado para gerar um mapa com 200 computadores.

Constata-se que no Cenário 1 a maior média e o tempo total projetado de 260 segundos para a geração do mapa de computadores. No cenário 2, com a criação do índice sobre o campo *datahorafinal* a média obtida foi de 63,15 ms, e um tempo total de 12,63 segundo para gerar o mapa de computadores, um ganho de aproximadamente 50 vezes em relação ao Cenário 1. Nota-se que no Cenário 1 o tempo foi constante para todos os computadores, enquanto que no Cenário 2 as tabelas com mais registros tiveram um tempo de resposta menor.

No Cenário 3 o tempo de projeção foi de 37,71 segundos, apresentando um comportamento contrário ao do Cenário 2, onde computadores com mais registros tiveram um tempo de resposta maior. No Cenário 4, foram criados dois índices sobre os campos *datahorafinal* e *nomedohost*, e os tempos obtidos foram semelhantes aos do Cenário 2, apresentando um pequeno ganho.

Visto que o total de registros tende a ser crescente na tabela de processos, os Cenários 2 e 4 apresentam os melhores resultados em relação ao Cenário 3, onde o tempo tende a crescer em função do número de registros, e em relação ao Cenário 1, que apresenta tempos contantes, porém muito altos em relação ao demais.

Em análise específica dos Cenários 2 e 4 pode-se constatar que somente a criação sobre o campo *datahorafinal* traz o menor tempo de consulta médio entre os computadores

analisados. Porém a criação do índice sobre o campo *nomedohost* seria funcional em consultas onde somente este campo está presente, como por exemplo, na obtenção da média de processamento de um determinado computador. Neste caso com a criação dos dois índices o tempo de consulta seria de 292 ms, enquanto que somente com o índice sobre o campo *datahorafinal* seria de 437 ms.

Optou-se então pela criação dos índices sobre os dois campos da tabela *processos.* Além disso os mesmos índices foram criados sobre as tabelas *partições*, *temperaturas* e os campos correspondentes da tabela *memória*, que armazenam informações com a mesma finalidade e possuem comportamento semelhante.

3.6.3.2 Menu Geral – Estatística geral

A opção "Estatística geral" da entrada do menu "Geral" exibe informações de contabilidade geral do sistema como:

a) Total de computadores cadastrados no sistema;
b) Total de computadores ligados;
c) Total de sessões de usuários abertas;
d) Total de avisos do sistema gerados nos últimos 10 minutos.

3.6.3.3 Menu Geral – Computadores ligados

Na entrada do menu principal "Geral", a opção "Computadores ligados" traz em forma de tabela a lista dos computadores ligados no momento da consulta, apresentando as informações do ID do tempo ativo, nome do computador, o sistema operacional atual em execução, o momento que a máquina foi ligada, o momento do último contato realizado, e a lista de usuários online. Clicando sobre a opção do nome do computador a página é redirecionada para uma tela com informações específicas deste. A Figura 33 ilustra uma parcial da tela retornada a partir da seleção da opção "Computadores ligados".

Total de 27 computadores ligados.

Computadores Ligados

ID	Nome do host	SO	Ligado em	Último contato	Usuários ONLINE
4931	labserver	Linux	2010-10-26 14:23:22	2010-11-11 14:11:23.661745	
5169	lab1-1	Linux	2010-11-11 07:40:22	2010-11-11 14:11:23.901095	
5170	lab18-24	Windows	2010-11-11 07:58:59	2010-11-11 14:11:06.510301	
5173	lab18-36	Windows	2010-11-11 07:58:23	2010-11-11 14:11:09.581527	

Figura 33 Tela da opção "Computadores ligados" da entrada do menu "Geral"

3.6.3.4 Menu Geral – Sessões abertas

Na entrada do menu principal "Geral", a opção "Sessões abertas" traz em forma de tabela a lista dos usuários online no sistema no momento da consulta. As informações apresentadas são o ID da sessão aberta, nome do computador, o *login* do usuário, o sistema operacional atual em execução e o momento que a sessão foi aberta. Clicando sobre a opção do nome do computador a tela atual é redirecionada para uma nova tela com informações específicas.

3.6.3.5 Menu Geral – Avisos do sistema

A opção "Avisos do sistema" da entrada do menu "Geral" exibe informações sobre os avisos gerados pelo coletor em função do tratamento da tentativa de coletar as estações monitoradas através da variável de controle "Agente Ativo".

A análise destes registros de avisos permite identificar se algum computador teve a execução de seu agente interrompida, para que medidas possam ser tomadas para verificar se a interrupção ocorreu por falha no agente ou de forma intencional por algum usuário. Os registros de avisos permitem detectar também se uma estação está rodando uma versão do agente inferior a atual.

3.6.4 Menu Cadastros

A entrada "Cadastros" do menu principal disponibiliza o acesso as opções de cadastro de informações do sistema, que compreendem o cadastro de prédios, laboratórios, coletores, computadores, usuários e sistemas operacionais.

Cada uma das opções de cadastro exibe um formulário que permite ao usuário informar os atributos do objeto a ser cadastrado. E logo abaixo são listados todos os registros já armazenados no sistema, permitindo que o usuário possa alterar um registro ou excluí-lo. A Figura 34 ilustra a tela de cadastro de um computador onde pode-se observar o formulário de cadastro de um novo computador, e mais abaixo uma parcial da lista dos computadores já cadastrados no sistema.

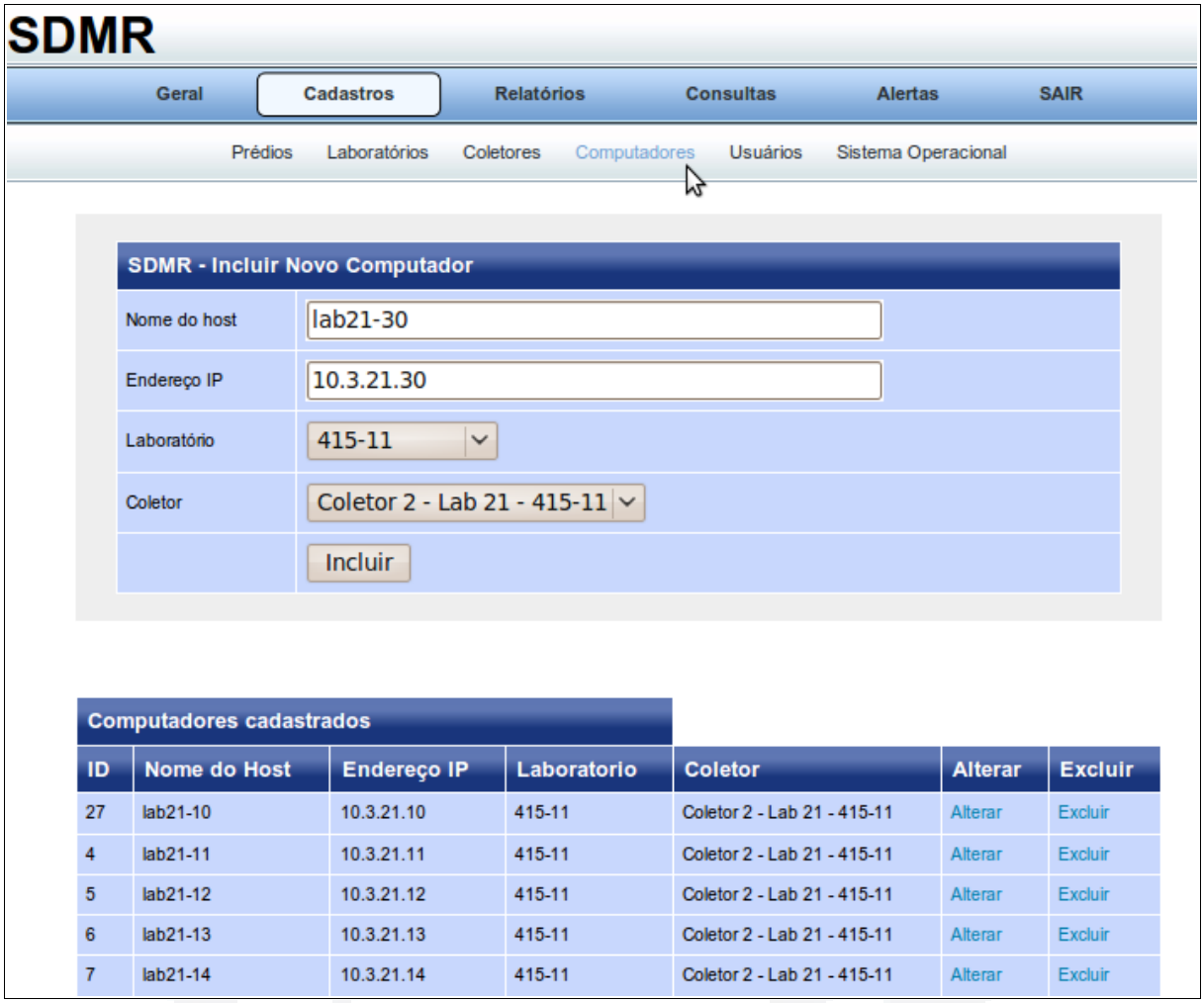


Figura 34 Visualização da tela de cadastro de computadores do console WEB

3.6.5 Menu Relatórios

A opção “Relatórios” do menu principal concentra o acesso as opções de geração de relatórios. Os relatórios implementados analisam informações de tempo de utilização e ociosidade dos computadores monitorados. A seguir cada tipo de relatório é tratado especificamente.

3.6.5.1 Menu Relatórios – Tempo de uso

A opção “Tempo de uso” da entrada de menu “Relatórios” dá acesso ao formulário para gerar relatórios de tempo de uso em horas.. Os relatórios por tempo de uso analisam as sessões de usuários em função de um determinado período de tempo e as agrupam de acordo com o parâmetro selecionado pelo administrador. Os parâmetros de agrupamento disponíveis são:

a) Por curso: agrupa as informações de sessões de usuários pelos cursos aos quais os usuários estão vinculados, possibilitando obter uma lista com a quantidade de horas que os computadores foram utilizados por usuários de determinado curso.
b) Por S.O.: agrupa as informações pelos sistemas operacionais cadastrados, possibilitando estabelecer o tempo de uso em cada um dos sistemas.
c) Por Laboratório: agrupa as informações de tempo de uso por laboratório cadastrado, permitindo determinar o tempo de uso de cada laboratório.
d) Por usuário: agrupa as informações por usuário, permitindo identificar quais usuários fazem mais uso dos computadores monitorados.
e) Por computador: agrupa as informações de sessões de usuário por computador, o que permite estabelecer o tempo de uso de cada computador monitorado.

O período de tempo deve ser definido no formulário afim de determinar o intervalo de análise e agrupamento dos dados. O usuário deve informar a data e hora inicial e data e hora final. Na Figura 35 é apresentada a tela do formulário para gerar o relatório de tempo de uso.

Figura 35 Tela de relatório de tempo de uso

3.6.5.2 Menu Relatórios – Ociosidade

Os relatórios de ociosidade são acessados pela entrada de menu “Relatórios” na opção do sub-menu “Ociosidade”. A tela apresentada é um formulário que permite ao usuário estabelecer os parâmetros de escopo do relatório, de agrupamento das informações e definir o intervalo de tempo que os dados devem se restringir. O relatório de ociosidade pode ser gerado em função de dois escopos de análise:

a) Tempo ativo X Sessões: analisa o total de tempo em que os computadores ficaram ligados em função da quantidade de tempo que foram utilizados por usuários, permite obter o tempo que as máquinas não foram utilizadas e estavam ligadas. Este tempo é definido como ociosidade funcional.
b) Tempo total X Sessões: analisa o número total de computadores e o total de horas compreendidos no intervalo informado pelo usuário, a partir destas duas informações calcula o tempo total que os computadores estariam disponíveis para utilização e relaciona estas informações com o total de sessões de usuário. Permite obter a ociosidade total das máquinas e identificar o tempo de utilização de cada uma independente dela estar ligada ou não. Este tempo é definido como ociosidade total.

Além do escopo do relatório é possível definir também o agrupamento das informações a serem geradas, que são:

a) Geral: analisa todos os computadores cadastrados no sistema e traz uma informação geral de ociosidade, agrupando todos os registros num resultado único.
b) Por Laboratório: agrupa os computadores por laboratório e gera as informações de ociosidade em relação a cada laboratório. Permite obter o percentual de ociosidade por laboratório.
c) Por computador: cada computador é analisado individualmente e os dados de ociosidade são gerados para cada um deles. Permite obter informações de utilização e ociosidade de cada estação cadastrada no sistema.

Para determinar o intervalo de análise e agrupamento dos dados o período de tempo deve ser definido no formulário. O usuário deve informar a data e hora inicial e data e hora final. Na Figura 36 é apresentada a tela do formulário para gerar o relatório de ociosidade.

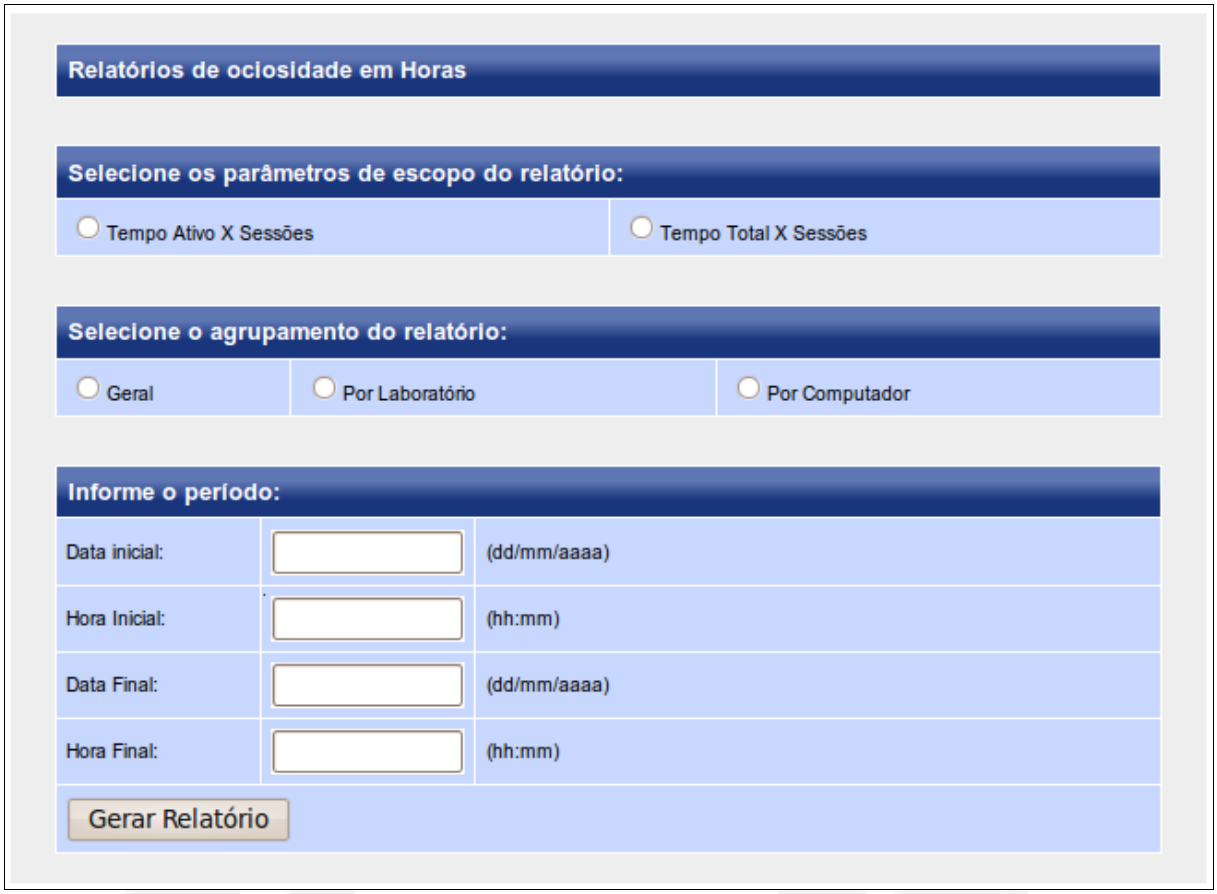


Figura 36 Tela de relatório de ociosidade

3.6.6 Menu Consultas

A entrada "Consultas" do menu principal concentra o acesso as opções de consultas migradas do console original do SDMR, originalmente implementadas por Stoll(2008). As consultas foram adaptadas e fornecem novas opções de configuração que permitem a especificação de filtros e parâmetros, oferecendo a visualização personalizada das informações. Três consultas estão disponíveis: processos, partições e temperaturas.

A tela de consulta de processos permite a configuração dos seguintes parâmetros:

a) Ordenação: As informações podem ser ordenadas por percentual de CPU, percentual de memória ou data e hora da captura.

b) Referência: As informações retornadas pela consulta podem ser filtradas por uma faixa de percentual, definida como "Percentual de corte". A faixa pode ser relacionada ao percentual de CPU ou memória. Por exemplo, é possível consultar apenas processos que tenham percentual de uso de CPU entre 70 e 90%.

c) Processos: Este parâmetro permite filtrar a consulta de um processo específico ou de todos os processos.

d) Computadores: Parâmetro que configura de quais computadores os dados serão consultados. Permite a seleção de todos os computadores, de um determinado laboratório ou um computador específico.

e) Período: Permite a configuração do intervalo de tempo que as informações serão consultadas. Devem ser informadas as datas e horas inicial e final.

As telas de consulta de partições e temperatura permitem a configuração dos parâmetros de ordenação, computadores e período do intervalo de consulta.

3.6.7 Menu Alertas

A entrada do menu principal "Alertas" disponibiliza o acesso ao recurso de alertas implementado no sistema SDMR. Este recurso tem o objetivo que o sistema, baseado nos parâmetros configurados pelo administrador, possa detectar determinada situação e gerar um registro de alerta, que pode ser visualizado pelo usuário. Este tipo de informação permite que determinada situação seja detectada automaticamente e apresentada ao administrador sem que o mesmo precise através de relatórios e consultas gerar esta informação. A funcionalidade implementada neste trabalho compreende a geração de alertas de ociosidade.

A entrada "Alertas" dá acesso a um submenu com duas opções: "Configuração" e "Visualizar". A opção "Configuração" permite configurar o tempo em minutos que um determinado computador deve permanecer ligado e sem ser utilizado para que um alerta de ociosidade seja gerado. A opção "Visualizar" permite visualizar os alertas gerados pelo sistema.

Para suportar o armazenamento da configuração dos alertas e do histórico dos alertas já gerados duas novas tabelas foram adicionadas à estrutura da base de dados. A Figura 37 ilustra a estrutura das novas tabelas *alertasconf* e *alertashistorico*.

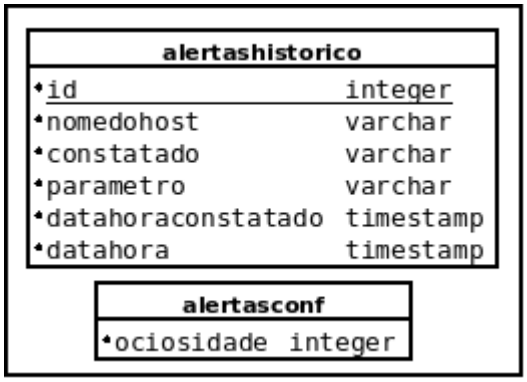


Figura 37 Tabelas *alertashistorico* e *alertasconf*

A tabela *alertasconf* é composta pelo campo *ociosidade*: campo inteiro que armazena o valor em minutos de referência para que um alerta de ociosidade seja gerado.

A tabela *alertashistorico* é composta pelos campos:

a) *id*: campo número inteiro e sequencial. Valor único definido pelo banco de dados no momento da inserção de um registro.
b) *nomedohost*: registro de texto que identifica o nome da estação da qual o alerta foi gerado.
c) *constatado*: mensagem que informa a situação que originou o alerta.
d) *parametro*: mensagem que armazena os parâmetros usados que deram origem ao alerta.
e) *dataHora*: armazena a data e hora no formato *timestamp* do momento em que o alerta foi originado.
f) *datahoraconstatato*: armazena a data e hora no formato *timestamp* do momento que o alerta foi constatado.

Este capítulo apresentou as implementações realizadas para que o sistema SDMR suportasse a captura das novas informações agregadas. As informações foram classificadas e cada um dos componentes do sistema foi reestruturado. Novas funcionalidades foram implementadas ao console WEB oferecendo um ambiente de análise dos dados gerados pelo sistema. O próximo capítulo apresenta as experimentações realizadas até a implantação final do sistema, e os resultados obtidos através da análise das informações.

4 EXPERIMENTAÇÃO E RESULTADOS OBTIDOS

Este capítulo tem por objetivo apresentar as experimentações realizadas e os resultados obtidos com o desenvolvimento e implantação do presente trabalho. O capítulo foi dividido da seguinte forma: num primeiro momento serão apresentadas as considerações sobre as experimentações realizadas e os problemas encontrados até a implantação final do sistema. Em seguida as novas informações coletadas pelo sistema serão validadas com a análise dos dados obtidos através de consultas e relatórios especificamente desenvolvidos no console WEB. Após serão apresentados os resultados de avaliação de impacto dos agentes e dos coletores em execução. E por fim, serão feitas considerações sobre os resultados obtidos nos quesitos de escalabilidade e robustez do sistema.

4.1 Experimentação I – Agente Linux em dois módulos

As novas funcionalidades de captura de informações de tempo ativo e sessões foram agregadas ao agente Linux e testadas em ambiente simulado apresentando comportamento sem falhas. No entanto, ao ser instalado num ambiente real o agente apresentou comportamento instável, tendo sua execução terminada abruptamente. Após a análise dos aspectos que envolviam o término da execução do agente foi possível constatar que o problema ocorria quando um usuário iniciava o uso do computador. Neste momento o comportamento do sistema em relação ao número de inicialização e término de processos aumentava consideravelmente, esta situação poderia estar causando o término inesperado da execução do agente.

Para buscar a origem deste problema o agente Linux, denominado "*agented*" foi instalado em sua versão original, e um novo módulo foi criado somente com as novas funcionalidades agregadas ao sistema, este módulo foi denominado de "*agentedlog*". Os dois agentes foram executados simultaneamente em *background*, e constatou-se que o *agentedlog* não apresentou problema em sua execução, enquanto que o *agented* original continuava tendo sua execução terminada inesperadamente quando ocorria o acesso de um usuário.

Num segundo momento, a utilização de tratamento de sinais através da função *signal()* foi implementado no *agented.* Desta forma buscou-se verificar a origem de sua queda através do sinal recebido por esta função de tratamento, para em seguida ignorá-la e permitir que a execução do processo continuasse. Durante os testes constatou-se que o sinal que ocasionava a queda do agente era de valor numérico 11, denominado SIGSEGV. Este sinal é enviado ao processo quando existe uma violação no acesso à memória, como o acesso a um registro fora

do endereçamento destinado ao processo.

A solução de tratamento dos sinais permitiu identificar o tipo de problema que gerava a queda do *agented*, no entanto não o resolveu, porque ao ignorar o sinal recebido do sistema operacional e permitir que o processo continuasse sua execução criava-se um *loop* de execução, visto que o processo continuava sua execução do mesmo ponto que o sinal havia sido gerado, consumindo assim alto índice de processamento, onerando o sistema de forma ineficaz.

Como alternativa testou-se ainda o agendamento de uma tarefa do tipo *guardian*, que verificava se os agentes estavam em execução e constatando que não, reiniciavam sua execução. Porém esta medida também não resolvia o problema de forma correta, só restaurava um estado de erro.

Frente a este problema que impedia a implantação total do sistema foi necessário a revisão completa do agente, com a adição de mensagens de controle buscando identificar o trecho de código que ocasionava a falha de segmentação do processo.

Após esta revisão encontrou-se a origem do problema no arquivo *agtProcesso.c* na linha 82, onde a função *fclose()* era chamada para fechar o arquivo apontado pela variável *status* do tipo *FILE*, conforme ilustrado na Figura 38. É possível verificar na linha 62 que o arquivo de endereço armazenado na variável *caminho* é aberto com a função *fopen*(), e o ponteiro para o objeto *FILE* é armazenado na variável *status*. Em seguida é feita a verificação através da função *IF* se o arquivo foi aberto corretamente, para somente então obter as informações que armazena. No entanto, caso o arquivo não tenha sido aberto corretamente a função *fclose()* é executada de qualquer forma, recebendo por parâmetro a variável *status*. Esta variável que não possui um ponteiro válido para um arquivo. Neste momento o erro de falha de segmentação é gerado e o agente tem sua execução terminada.

agtProcesso.c

```
61      snprintf(caminho, UTL_TAM_MAX_P, "%s/%d/stat", PROC_DIR, pid);
62      status = fopen(caminho, "r");
63      if (status)
64      {
65
66          fscanf(status, "%d %s %c %d "
67                         "%d %d %d %d "
68                         "%lu %lu %lu %lu "
69                         "%lu %lu %lu",
70                          &processo_atual->pid, &processo_atual->comm, &cNull, &dNull,
71                          &dNull, &dNull, &dNull, &dNull,
72                          &luNull, &luNull, &luNull, &luNull,
73                          &luNull, &processo_atual->utime, &processo_atual->stime);
74
75
76          utlUsuarioDoProcesso(processo_atual->pid, processo_atual->usuario);
77
78          float tempoTotal = (float) processo_atual->utime + processo_atual->stime;
79          processo_atual->jiffies = tempoTotal - processo_atual->ultimosJiffies;
80          processo_atual->ultimosJiffies = tempoTotal;
81      }
82      fclose(status);
```

Figura 38 Trecho de código do arquivo agtProcesso.c

O problema ocorre com mais frequência quando um usuário inicia o uso do computador em função do maior número de processos que são iniciados e finalizados, juntamente com a forma de análise que é feita dos processos. Num primeiro momento é obtida a lista de todos os arquivos existentes em */proc/*, para que assim se tenha acesso a todos os processos em execução no sistema, em seguida esta lista é percorrida e cada processo possui suas informações extraídas, conforme trecho de código da Figura 38. Porém no caso de algum processo ter terminado sua execução em meio a este processo, a função *fclose()* é executada e a falha de segmentação ocorre, pela tentativa de acesso à um endereço inválido de memória.

O problema é corrigido movendo a função *fclose()* para dentro do escopo da função *IF*, fazendo com que sua execução ocorra somente se a variável *status* possua um ponteiro válido para um arquivo.

4.2 Experimentação II – Agente Linux unificado

Na segunda experimentação o *agented* em estado funcional e o *agentedlog* que continha as novas funções de captura foram unificados e implantados no sistema operacional Linux em 202 estações de trabalho dos Laboratórios de Informática da UNIVATES. A distribuição ocorreu conforme a Tabela 9.

Tabela 9 Tabela com lista de laboratórios da Experimentação II

Laboratório	Número de computadores
101-7	31
102-7	25
103-7	25
104-7	31
101-11	25
407-12	40
415-11	25
TOTAL	202

A implantação ocorreu ao longo dos dias 11, 12 e 13 de outubro de 2010 e durante os dias 14 e 15 do mesmo mês o coletor do sistema SDMR operou coletando as informações dos computadores e armazenando-as na base de dados. Durante os dois dias de coletas a análise visual do número de estações ligadas no sistema operacional Linux foi feita e observou-se um número reduzido de estações sendo utilizadas no S.O. Linux em comparação com o total monitorado.

Então, no dia 16 de outubro de 2010 foi feita a análise dos dados através das informações armazenadas e constatou-se que a utilização por usuários do sistema operacional Linux era reduzida. O gráfico da Figura 39 ilustra o percentual de sessões de usuário no dia 14 de outubro de 2010, que foi de 7.92%, totalizando 16 sessões, e no dia 15 de outubro de 2010, onde o percentual de sessões foi de 14,35%, totalizando 29 sessões.

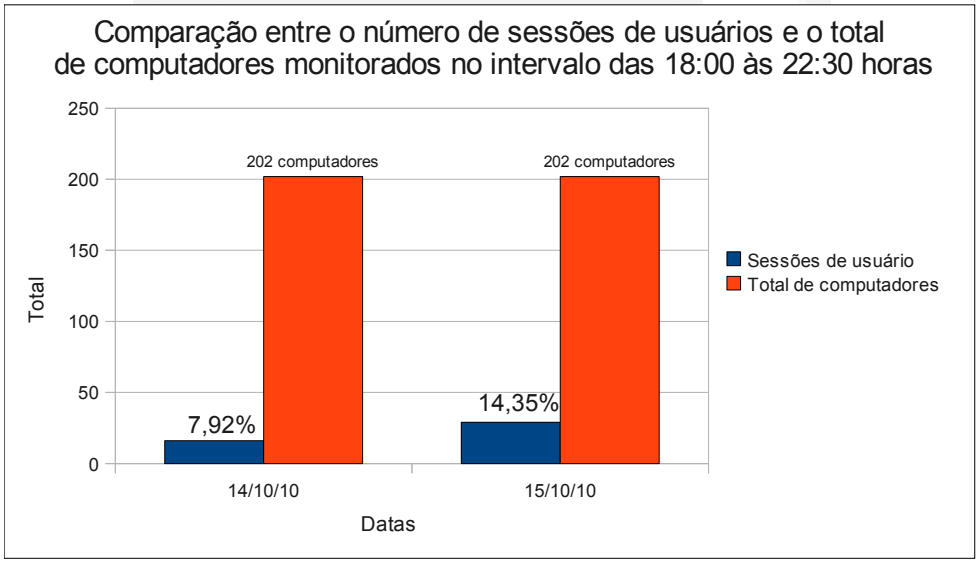


Figura 39 Comparação entre o número de sessões de usuários e o total de computadores monitorados no S.O. Linux

Frente a este cenário encontrado e verificando os objetivos do trabalho em gerar informações gerenciais para análise de utilização dos Laboratórios de Informática utilizando a ferramente SDMR, constatou-se a necessidade de ampliação do escopo do trabalho incluindo a reestruturação do agente para o sistema operacional Windows.

4.3 Experimentação III – Agente Linux e Windows

Após a implementação do agente Windows conforme descrito no Capítulo 3, uma nova experimentação foi realizada com o monitoramento de 50 computadores portando os agentes para Windows e Linux, os computadores estavam distribuídos nos Laboratórios 101-11 e 415-11. Durante o dia 19 de outubro de 2010 as estações foram monitoradas e posteriormente os dados foram analisados delimitando o mesmo período de tempo, das 18:00 às 22:30.

Em análise dos dados constatou-se que o percentual de utilização de sessões aumentou consideravelmente, totalizando 72% em relação aos 50 computadores monitorados, num total de 36 sessões, sendo que 27 sessões em Windows e 9 sessões em Linux.

O percentual de uso em sessões de Linux foi de 18%, número um pouco acima do constatado na experimentação II, mas não muito distante, enquanto que a utilização de sessões em Windows atingiu o percentual de 54%, conforme ilustrado na Figura 40. Isto confirma a maior utilização do sistema operacional Windows e justifica a necessidade da reestruturação do agente Windows para uma análise mais robusta dos dados gerados pelo sistema SDMR.

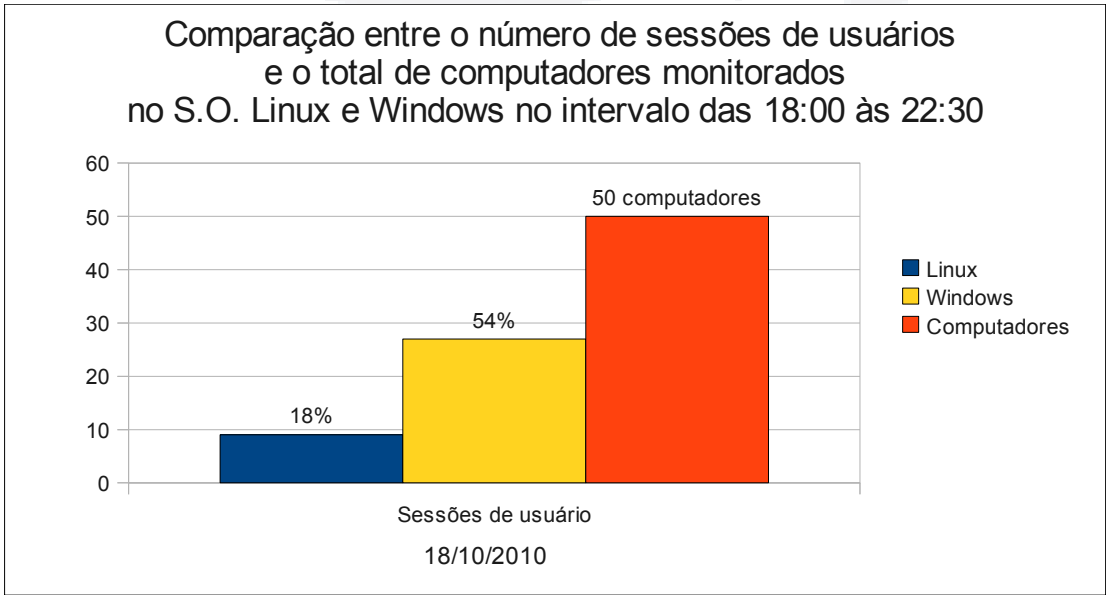


Figura 40 Comparação entre o número de sessões de usuários e o total de computadores monitorados no S.O. Linux e Windows

Considerações sobre a reestruturação e implantação do agente Windows

Conforme citado no Capítulo 2, o agente Windows implementado por Stoll(2008) roda sobre o ambiente CygWin, o que condiciona sua execução a instalação deste ambiente. Para rodar o serviço SNMP é necessário a compilação do seu código fonte, porém não foi possível obter êxito utilizando versões mais atuais do SNMP, somente utilizando a versão especificada no trabalho de Stoll(2008) e aplicando o *patch*[5] de correção.

A reestruturação do agente Windows foi facilitada na implementação de algumas das novas funcionalidades visto que o ambiente CygWin oferece uma estrutura de arquivos do diretório */proc* semelhante ao sistema Linux, e algumas informações puderam ser coletadas da mesma forma nos dois sistemas operacionais. A funcionalidade de captura de sessões teve que ser implementada novamente, através da análise dos processos em execução.

Em contrapartida, mesmo com o ambiente CygWin instalado e configurado, o agente apresentou instabilidade com algumas funções que gerenciam a liberação de memória. A fim de propiciar a funcionalidade primordial de coleta das informações e poder gerar resultados sobre sua análise o sistema foi onerado com a utilização de um percentual de memória cumulativo de crescimento linear.

A seguir serão apresentados os parâmetros de configuração do agente e coletor para que o processo de implantação final do sistema pudesse ser realizado.

4.4 Parametrização dos arquivos de configuração do agente e coletor

Os parâmetros de configuração do agente e coletor tiveram como base os resultados obtidos por Stoll(2008), que analisou o impacto do agente Windows em um ambiente de produção com 13 computadores e fez comparações com resultados obtidos por Spezia(2007) do impacto do agente Linux.

É enfatizado por Stoll(2008) que a configuração ideal do agente e coletor variam com as necessidades de cada ambiente e deve ser feita relevando os aspectos da granularidade de obtenção de informação a ser obtida e o impacto que o perfil configurado irá causar nas estações de trabalho e na rede de computadores.

Com base nestas premissas elaborou-se os seguintes perfis de configuração para o agente e coletor, apresentados respectivamente na Tabela 10 e Tabela 11.

[5] *Patch* #1805971– para a correção de registro de acesso. Mais informações no site: http://sourceforge.net/tracker/index.php?func=detail&aid=1805971&group_id=12694&atid=456380.

Tabela 10 Configuração dos parâmetros de captura do agente

Parâmetro	**Configuração**
Tipo de captura	0 (processos com uso de CPU > 0)
Captura de processos	30 segundos
Captura de temperatura	Funcionalidade não utilizada
Captura de partições	240 segundos
Captura de eventos	15 segundos
Captura de memória	120 segundos

O parâmetro do tipo de captura do agente visualizado na tabela acima foi configurado para "somente processos com uso de CPU maior que zero", causando menos impacto no sistema e buscando processos que efetivamente estão consumindo processamento. O tempo de captura das informações de processo foi configurado em 30 segundos com base na análise de impacto realizada por Stoll(2008), que foi do cenário que apresentou menor impacto às estações. A funcionalidade de captura de temperatura não foi habilitada nos agentes. A captura de informações de partições foi configurada em 240 segundos buscando detectar variações bruscas do uso das partições. A captura de eventos foi configurada para 15 segundos, para que fosse possível a detecção de utilização de intervalos pequenos de tempo e a publicação mais rápida das informações na MIB. A captura dos valores de memória foi configurada em 120 segundos, valor quatro vezes maior que o de processos, o que garante um impacto menor na estação e uma granularidade de informação conveniente para análise deste trabalho. Os impactos causados pelas escolhas realizadas serão apresentados e discutidos na sessão 4.9 deste Capítulo.

Tabela 11 Configuração dos parâmetros de captura do coletor

Parâmetro	**Configuração**
Standby	1 ciclo
Retry	3 ciclos
Ciclo de coleta	45 segundos

Conforme visualizado na tabela acima, o coletor foi configurado com o ciclo de coleta de 45 segundos, permitindo que informações geradas num espaço de tempo de aproximadamente 1 minuto sejam coletadas e armazenadas, considerando como uma atualização satisfatória das informações. O parâmetro do tempo de espera *StandBy* foi

configurado em 1 ciclo, o que ocasiona que na primeira tentativa de coleta sem sucesso o computador é colocado em modo de espera. O parâmetro de nova tentativa *Retry* foi configurado para 3 ciclos, garantindo assim que a cada três ciclos de coleta uma estação em estado de espera volte a ser coletada.

4.5 Experimentação IV - Implantação do sistema SDMR

A experimentação IV do trabalho compreende a implantação final do sistema SDMR nos Laboratórios de Informática. Os agentes do sistema foram instalados em 216 computadores nos sistemas operacionais Windows XP e Linux Ubuntu 9.04 no dia 25 de outubro de 2010. Em Linux o agente foi executado em *background* como um *daemon*, junto ao arquivo executado na inicialização do sistema */etc/rc.local*. Em Windows o agente foi executado como um serviço do sistema. Na Tabela 12 são apresentados os laboratórios onde o sistema foi implantado e o número de computadores presentes em cada um deles.

Tabela 12 Tabela com lista de laboratórios da Experimentação IV – Implantação final do sistema SDMR

Laboratório	Número de computadores
Coordenação	3
101-7	31
102-7	25
103-7	25
104-7	31
105-7	25
101-11	25
413-11	26
415-11	25
TOTAL	216

Os coletores foram instalados no computador com sistema operacional Linux Ubuntu 9.04 Server localizado na sala da coordenação dos Laboratórios de Informática. A seguir serão feitas considerações sobre a sua execução.

Utilização de coletores em *multi-process*

A execução do coletor foi realizada utilizando a arquitetura *multi-process*, onde vários coletores foram instanciados, cada um responsável por um determinado laboratório. Um dos critérios utilizados para a opção desta arquitetura de execução está relacionado com o fato de

que a estrutura física atual do setor dos Laboratórios de Informática disponibiliza apenas um computador com S.O. Linux ligado durante todo o turno de atividade do setor e passível de suportar a execução do coletor.

O segundo critério está em relação ao tempo necessário para que o coletor consiga percorrer todas as estações de trabalho. Como visto no Capítulo 3 na reestruturação do componente coletor, quando um computador está desligado o tempo total para detectar este estado dura aproximadamente 4 segundos, em função do *timeout* do protocolo SNMP. Levando em conta o cenário proposto de implantação, e citando como exemplo a situação em que 66 computadores estão ligados e 150 desligados, teríamos como consequência um tempo total de execução do ciclo em função das máquinas desligadas de 600 segundos ou 10 minutos. Neste cenário as 66 estações ligadas seriam coletadas no mínimo a cada 10 minutos, o que anularia uma configuração de tempo de ciclo de coleta menor, como a de 45 segundos configurada nesta implantação.

A Tabela 13 apresenta os coletores configurados e o laboratório ao qual cada um está associado. Cada coletor é responsável por coletar todos os computadores cadastrados para o laboratório. Todas estas configurações são feitas via console WEB.

Tabela 13 Tabela com lista de coletores da Experimentação IV

Coletor	**Laboratório**
Coletor 1	Coordenação
Coletor 2	101-7
Coletor 3	102-7
Coletor 4	103-7
Coletor 5	104-7
Coletor 6	105-7
Coletor 7	101-11
Coletor 8	413-11
Coletor 9	415-11

A execução do coletor nesta arquitetura, juntamente com as configurações de espera *STANDBY* e nova tentativa de coleta *RETRY*, permitem obter um resultado mais aproximado do valor parametrizado para o tempo de ciclo de coleta de cada coletor. Os impactos causados por esta arquitetura na estação onde ocorre a execução dos coletores será vista na sessão 4.9 deste Capítulo. A seguir serão apresentadas e analisadas as informações coletadas pelo sistema.

4.6 Análise das informações coletadas - mapa de computadores

Esta sessão e as duas seguintes tem por objetivo analisar as informações coletadas pelo sistema SDMR. Todas as informações de análise serão obtidas através do console WEB, pelas funcionalidades nele implementadas. Desta forma consolida-se a reestruturação do console, que teve o intuito de oferecer um ambiente funcional capaz de fornecer informações gerenciais que possam ser utilizadas como auxílio na tomada de decisão.

A primeira análise é feita sobre a tela "Mapa de computadores", acessado pela opção de menu "Geral". A Figura 41 ilustra a tela gerada através do console WEB utilizando como parâmetro de estado "Ligados e desligados" e como filtro de computadores a opção "Somente Laboratório", onde o Laboratório 101-7 foi selecionado. O mapa gerado tem informações de todos os computadores do laboratório 101-7 independente de estarem ligados ou desligados. Para fins de análise a figura foi editada e apresenta os 6 primeiros computadores retornados pelo mapa gerado.

Mapa de computadores

S.O.	Nome e IP	Último contato	Uso CPU	Uso Memória	Total Memória	Uso HD	Temperatura	Usuários online
	lab18-10 / 10.3.18.10	11/11/2010 14:11:01	,47%	52,00	1011 MB	94,95 %	0 °C	1 usuário(s) Ocioso 0 min.
	lab18-11 / 10.3.18.11	DESLIGADO	,18%	8,00	994 MB	11,34 %	0 °C	0 usuário(s) Ocioso 0 min.
	lab18-12 / 10.3.18.12	DESLIGADO	1,08%	16,00	994 MB	11,34 %	0 °C	0 usuário(s) Ocioso 0 min.
	lab18-13 / 10.3.18.13	11/11/2010 14:11:01	,33%	7,00	994 MB	22,74 %	0 °C	0 usuário(s) Ocioso 173 min.
	lab18-14 / 10.3.18.14	11/11/2010 14:11:01	,37%	16,00	994 MB	11,34 %	0 °C	1 usuário(s) Ocioso 0 min.
	lab18-15 / 10.3.18.15	11/11/2010 14:11:03	,18%	41,00	1015 MB	98,01 %	0 °C	0 usuário(s) Ocioso 57 min.

Figura 41 Corte da tela "Mapa de computadores" gerada pelo console WEB

A disposição em linhas dos computadores permite que a análise de um atributo seja alinhada e feita por coluna. Por exemplo, na Figura 41 podemos visualizar informações importantes como a alta taxa de utilização do HD na coluna "Uso HD" nas estações lab18-10

e lab18-15. Na coluna “Usuário online” podemos visualizar que as estações lab18-13 e lab18-15 estão ligadas e não estão sendo utilizadas no momento, estando ociosas por 173 e 57 minutos respectivamente.

A visualização destas informações permite ao administrador detectar estas situações e tomar as medidas necessárias para solucioná-las, como por exemplo, realizar a limpeza de disco nas estações com alta taxa de utilização do HD antes que o sistema emita avisos de pouco espaço em disco, e desligar as estações que estão ociosas por um longo período de tempo, gerando assim uma possível economia de energia.

4.7 Análise das informações coletadas - relatórios de tempo de uso

Esta sessão apresenta e analisa os relatórios de tempo de uso gerados através da ferramenta console WEB. Os relatórios de tempo de uso são acessados através da opção “Tempo de Uso” na entrada de menu “Relatórios” do console WEB. Na tela de configuração do relatório deve ser selecionado o agrupamento do relatório e o intervalo de tempo que será consultado. Os relatórios gerados para análise possuem como intervalo de tempo o período de 26 de outubro de 2010 as 00:00 até 9 de novembro de 2010 as 23:59.

4.7.1 Tempo de uso por curso

A Figura 42 ilustra a parte inicial do relatório gerado, com os cursos que obtiveram o maior número de horas associado. É possível observar que no período delimitado o curso com mais utilização é Engenharia da Computação Bacharelado, seguido por Direito Bacharelado, e na terceira linha identificamos a entrada “Curso CTTI”, que agrupa a utilização dos usuários associados aos cursos promovidos pelo Centro de Treinamento de Tecnologia da Informação – CTTI da UNIVATES. Na sexta linha encontra-se a entrada “SEM CURSO” que engloba usuários que não possuem um curso vinculado, como setores e alunos egressos[6].

Os parâmetros do período de análise podem ser configurados pelo administrador, permitindo gerar relatórios mais específicos, como de um turno de um determinado dia, buscando identificar os níveis de utilização por curso nos dias da semana ou em um período de tempo maior, como um semestre ou um ano, gerando valores que podem ser usados, por exemplo, para cálculo de custeio de manutenção ou novos equipamentos.

[6] Egresso indica que o aluno concluiu ou trancou um curso.

Relatório tempo de uso em horas por curso	
Período de 26/10/2010 00:00 a 09/11/2010 23:59	
Curso	**Horas**
ENGENHARIA DA COMPUTACAO, BACHARELADO	491.57
DIREITO, BACHARELADO	385.85
Curso CTTI	318.52
ENGENHARIA CIVIL, BACHARELADO	262.58
SISTEMAS DE INFORMACAO, BACHARELADO	242.67
SEM CURSO	192.68
CIENCIAS CONTABEIS, BACHARELADO	188.28
ENGENHARIA DE CONTROLE E AUTOMACAO, BACHARELADO	178.43
TECNICO EM INFORMATICA	175.03
TECNICO EM SEGURANCA DO TRABALHO	126.07
ARQUITETURA E URBANISMO, BACHARELADO	97.22
ENGENHARIA AMBIENTAL, BACHARELADO	94.12
ENFERMAGEM, BACHARELADO	86.48
ADMINISTRACAO - LFE ADMINISTRACAO DE EMPRESAS	81.07
CIENCIAS BIOLOGICAS, LICENCIATURA	80.38
ADMINISTRACAO - LFE ANALISE DE SISTEMAS	79.03
POS-GRADUACAO, EM NIVEL DE ESPECIALIZACAO, EM GESTAO UNIVERSITARIA - 2	77.42
ADMINISTRACAO, COM HABILITACAO EM COMERCIO EXTERIOR, BACHARELADO	70.78
FARMACIA, BACHARELADO	68.93

Figura 42 Relatório de tempo de uso por curso (parcial)

4.7.2 Tempo de uso por sistema operacional

Através do relatório de tempo de uso por sistema operacional, ilustrado na Figura 43, é possível visualizar o número de horas de sessões de usuários em cada um dos sistemas em monitoramento. Os resultados apresentados neste relatório reforçam a constatação realizada durante o desenvolvimento do sistema SDMR, de que o sistema operacional Microsoft Windows possuía uma taxa maior de utilização em relação ao sistema operacional Linux.

Relatório tempo de uso em horas por sistema operacional	
Período de 26/10/2010 00:00 a 09/11/2010 23:59	
Sistema Operacional	**Horas**
Linux	398.05
Windows	3818.15

Figura 43 Relatório de tempo de uso por sistema operacional

O percentual de utilização por cada sistema operacional pode ser observado em forma de gráfico na Figura 44. Observa-se que o maior percentual é do sistema operacional Windows, com 91%, enquanto do sistema operacional Linux é de 9%.

Figura 44 Gráfico de utilização por sistema operacional

O relatório de utilização por sistema operacional permite obter informações que podem justificar a compra de licenças de um determinado sistema, como por exemplo, o Windows, ou tomar alguma medida para buscar uma maior utilização de um sistema alternativo, como o Linux. Pode ainda determinar quais os rumos de treinamentos que devem ser oferecidos aos funcionários técnicos, como por exemplo, priorizar o aperfeiçoamento no sistema mais utilizado.

4.7.3 Tempo de uso por laboratório

O relatório de tempo de uso utilizando o agrupamento por laboratório permite obter o tempo de utilização em horas em cada laboratório monitorado. A Figura 45 apresenta o relatório gerado, onde é possível observar que os Laboratórios 101-7 e 101-11 possuem o maior número de horas de utilização. Esta situação reflete a realidade visto que estes são os únicos dos laboratórios monitorados que permanecem em atividade durante os três turnos de atendimento: manhã, tarde e noite. Os demais laboratórios funcionam normalmente no turno da noite, o que justifica o número menor de horas.

Observa-se uma diferença na utilização entre os laboratórios que funcionam normalmente no turno da noite, o laboratório 102-7 possui aproximadamente 516 horas de uso, enquanto que o 415-11 possui 297 horas, ambos com 25 computadores. Esta é uma

situação que deve ser melhor analisada, pois pode decorrer do fato de um deles ser utilizado em outro turno, ou que não tenha sido utilizado em algumas noites.

Relatório tempo de uso em horas por laboratório

Período de 26/10/2010 00:00 a 09/11/2010 23:59

Laboratório	Horas
101-7	904.95
101-11	867.78
104-7	550.17
102-7	516.33
413-11	451.53
103-7	325.52
105-7	302.78
415-11	297.07

Figura 45 Relatório de tempo de uso por laboratório

A análise de utilização por laboratório permite ao gerente identificar quais ambientes estão sendo mais utilizados. Com estas informações seria possível tomar medidas como disponibilizar nestes laboratórios as máquinas com configuração de hardware superior, buscando disponibilizar os melhores equipamentos à um número maior de usuários, ou criar uma rotina de rodízio, alternando os cronogramas de utilização, buscando uma distribuição de uso mais equilibrada.

A configuração do período em que o relatório é gerado permite obter informações de uso específicas, como por exemplo, do horário que antecede o início do turno da noite, buscando identificar onde se concentra o maior fluxo de utilização e usuários.

4.7.4 Tempo de uso por computador e por usuário

O relatório de utilização por computador permite obter o tempo de utilização de cada computador monitorado e a sua análise propicia ao gerente tomar medidas buscando, por exemplo, equilibrar o número de horas de utilização de cada equipamento. É possível a identificação dos locais de maior e menor utilização e criar um rodízio de computadores, trocando-os de posição.

Na Tabela 14 são apresentados doze computadores do laboratório 101-7, seis de maior utilização e seis de menor utilização. Os dados de tempo de uso foram gerados através do relatório de tempo de uso com o agrupamento por computador, em seguida as informações foram extraídas e organizadas. É possível, por exemplo, realizar a troca de posição dos computadores com maior utilização com os de menor utilização.

Tabela 14 Tabela com a lista dos computadores do laboratório 101-7 com maior e menor utilização

Mais utilizados		Menos utilizados	
Computador	**Horas**	**Computador**	**Horas**
lab18-10	132,3	lab18-40	11,42
lab18-25	61,5	lab18-11	11,37
lab18-26	47,93	lab18-27	10,63
lab18-36	47,63	lab18-19	9,68
lab18-14	43,15	lab18-12	9,45
lab18-15	42,78	lab18-32	8,22

Existe ainda a opção do relatório de tempo de uso por usuário, permitindo obter o número de horas de utilização de cada usuário. Para fins de privacidade estes dados não serão divulgados, porém através destas informações é possível, por exemplo, verificar o total de horas contabilizado por usuários com permissão administrativa sobre os computadores, normalmente utilizados para manutenções e atualizações. A seguir serão apresentados e analisados os relatórios de ociosidade na utilização dos equipamentos.

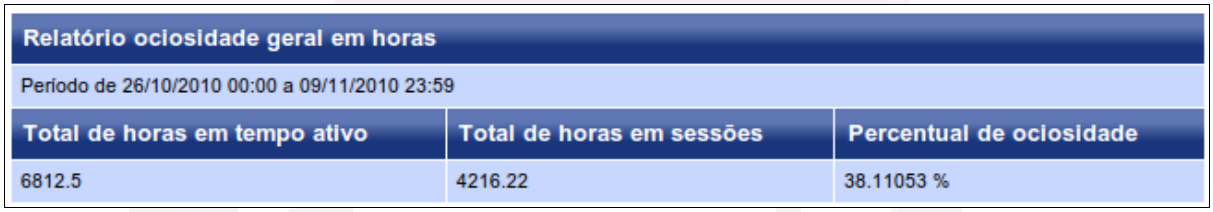
4.8 Análise das informações coletadas - relatórios de ociosidade

Esta sessão apresenta e analisa os relatórios de ociosidade gerados através da ferramenta console WEB. Os relatórios de ociosidade são acessados através da opção “Ociosidade” na entrada de menu “Relatórios” do console WEB. Na tela de configuração deve ser selecionado o parâmetro de escopo do relatório informando qual valor será usado como referência para definição do percentual de ociosidade, os dois parâmetros disponíveis são “Tempo Ativo X Sessões” onde o total de utilização por usuários é relacionado com o tempo de atividade da estação, já com o parâmetro “Tempo Total X Sessões” o tempo de utilização por usuários em sessões é relacionado com o total de tempo calculado pelo intervalo de tempo definido para a coleta, por exemplo, se o intervalo for das 19:00 às 22:00 de um mesmo dia serão contabilizadas três horas por computador. Esta opção permite verificar a ociosidade em sua totalidade de disponibilidade do equipamento e sua efetiva utilização.

4.8.1 Ociosidade geral de tempo ativo relacionado com sessões

A ociosidade geral obtida através do relacionamento entre o tempo ativo dos computadores e as sessões de usuários permitem identificar a quantidade de tempo em que as máquinas permaneceram ligadas e não utilizadas. A Figura 46 apresenta o relatório de ociosidade geral calculado sobre todos os registros de computadores no período de 26 de outubro de 2010 a 9 de novembro de 2010.

É possível visualizar que os computadores monitorados permaneceram ligados um total de 6815,5 horas e que foram utilizados em 4216,22 horas neste período de tempo. A ociosidade calculada é de 38,11%.



Relatório ociosidade geral em horas		
Período de 26/10/2010 00:00 a 09/11/2010 23:59		
Total de horas em tempo ativo	**Total de horas em sessões**	**Percentual de ociosidade**
6812.5	4216.22	38.11053 %

Figura 46 Relatório de ociosidade geral

O relatório geral permite obter valores gerais sobre os tempos de uso e ociosidade, para obter informações mais específicas o agrupamento por laboratório ou computador deve ser selecionado. Ou pode-se ainda especificar períodos de tempo por turno e identificar o comportamento neste espaço de tempo. A Tabela 15 e a Tabela 16 apresentam os valores obtidos através dos relatórios gerados no console WEB utilizando dois intervalos de horas, 13:00 as 19:10, definido como turno da tarde/pré-aula e 19:10 as 22:30 definido como turno da noite/aula, em quatro dias distintos, 4, 5, 8 e 9 de novembro de 2011.

Tabela 15 Tabela com percentual de ociosidade geral no turno da tarde/pré-aula

Dia	Intervalo de horas	Horas de tempo ativo	Horas em sessões	Percentual de ociosidade
04/11/2010	13:00 as 19:10	117,95 horas	61,82 horas	47,59%
05/11/2010	13:00 as 19:10	167,68 horas	70,3 horas	58,07%
08/11/2010	13:00 as 19:10	207,63 horas	156,35 horas	24,70%
09/11/2010	13:00 as 19:10	159,07 horas	102,88 horas	35,32%
	Média	**163,08 horas**	**97,84 horas**	**41,42%**

Tabela 16 Tabela com percentual de ociosidade geral no turno da noite/aula

Dia	Intervalo de horas	Horas de tempo ativo	Horas em sessões	Percentual de ociosidade
04/11/2010	19:10 as 22:30	490,82 horas	344,53 horas	29,81%
05/11/2010	19:10 as 22:30	310,07 horas	228,8 horas	26,21%
08/11/2010	19:10 as 22:30	395,05 horas	291,43 horas	26,23%
09/11/2010	19:10 as 22:30	386 horas	309,83 horas	19,73%
	Média	**395,49 horas**	**293,65 horas**	**25,49%**

Em análise dos dados obtidos é possível perceber que apesar da média que os computadores ficaram ligados no turno da tarde/pré-aula ser menor que no turno da noite/aula, o percentual de ociosidade é maior. Esta situação nos leva a constatar que no turno da tarde, quando o número de aulas é quase inexistente e apenas dois dos laboratórios monitorados ficam abertos e disponíveis para acesso livre dos usuários, computadores ficam ligados sem utilização. Informações como estas permitem que o gerente possa tomar medidas específicas a fim de diminuir a ociosidade no turno da tarde/pré-aula.

4.8.2 Ociosidade por laboratório relacionando tempo ativo com sessões

O relatório de ociosidade por laboratório permite identificar em quais ambientes o percentual de ociosidade é mais alto. A Figura 47 ilustra o relatório gerado no período de 26 de outubro de 2010 a partir das 00:00 até 9 de novembro de 2010 as 23:59. É possível observar que os dois laboratórios que possuem o maior percentual de ociosidade na relação entre sessões e tempo ativo são o 101-11 e 101-7, que como já apresentado anteriormente são laboratórios que estão abertos em todos os turnos de atividade.

Esta situação fica mais evidente analisando os totais de horas em tempo de atividade e sessões, que são os valores mais elevados dentre todos os laboratórios monitorados. Observa-se que os demais laboratórios possuem percentuais de ociosidade com valores mais baixos, destacando-se o laboratório 415-11, que possui um percentual de ociosidade de 8,21%. Verifica-se também que este laboratório possui os menores valores em totais de utilização de tempo ativo e sessões de usuário, indicando que sua utilização é feita de forma específica e que os computadores são ligados sob demanda.

Relatório de ociosidade por laboratório			
Período de 26/10/2010 00:00 a 09/11/2010 23:59			
Laboratório	**Total de horas em tempo ativo**	**Total de horas em sessões**	**Percentual de ociosidade**
415-11	323.65	297.07	8.212571 %
102-7	715.1	516.33	27.796108 %
105-7	417.7	302.78	27.51257 %
103-7	473.32	325.52	31.226234 %
413-11	520.8	451.53	13.30069 %
101-11	1730.5	867.78	49.853798 %
101-7	1878.87	904.95	51.83541 %
104-7	752.5	550.17	26.88771 %

Figura 47 Relatório de ociosidade por laboratório

Novas configurações podem ser aplicadas a este relatório buscando identificar em qual turno estes dois laboratórios estão mais ociosos. Como já visto na sessão anterior, constatou-se que em geral o turno, entre os dois que foram analisados, de maior percentual de ociosidade é o compreendido entre as 13:00 e 19:10, definido como tarde/pré-aula. Para reforçar esta análise a Figura 48 e a Figura 49 apresentam informações agrupadas retiradas de relatórios de ociosidade por laboratório, gerados em dois períodos de tempo no dia 9 de novembro de 2010. O primeiro período abrange o intervalo das 13:00 as 18:00, definido como tarde, e o segundo das 19:10 as 22:30, definido como noite.

Relatório de ociosidade por laboratório			
Período de 09/11/2010 13:00 a 09/11/2010 18:00			
Laboratório	**Total de horas em tempo ativo**	**Total de horas em sessões**	**Percentual de ociosidade**
415-11	0.0	0.0	0.0 %
102-7	0.0	0.0	0.0 %
105-7	0.0	0.0	0.0 %
103-7	0.0	0.0	0.0 %
413-11	0.0	0.0	0.0 %
101-11	85.22	56.1	34.170383 %
101-7	38.9	23.27	40.17995 %
104-7	0.0	0.0	0.0 %

Figura 48 Relatório de ociosidade por laboratório no turno da tarde

Relatório de ociosidade por laboratório			
Período de 09/11/2010 19:10 a 09/11/2010 22:30			
Laboratório	**Total de horas em tempo ativo**	**Total de horas em sessões**	**Percentual de ociosidade**
415-11	22.62	22.2	1.8567642 %
102-7	38.05	36.62	3.7582138 %
105-7	39.57	38.37	3.0326023 %
103-7	65.27	54.15	17.036917 %
413-11	39.62	36.47	7.950524 %
101-11	46.13	35.97	22.024712 %
101-7	73.0	38.5	47.260273 %
104-7	61.68	47.5	22.989624 %

Figura 49 Relatório de ociosidade por laboratório no turno da noite

Verifica-se que no turno da tarde os laboratórios em atividade restringiam-se ao 101-11 e 101-7, sendo que ambos apresentaram uma taxa de ociosidade alta. No turno da noite todos os laboratórios apresentaram atividade e ociosidade da relação entre tempo ativo e sessões, os valores percentuais variam e podem ainda estar relacionados com o laboratório estar em aula ou livre para uso geral. Porém observa-se que em média a ociosidade no turno da noite, que é calculada em 15, 73%, é menor do que no turno da tarde, calculada em 37,17%, mesmo que existam mais laboratórios em atividade.

Os relatórios de ociosidade apresentados relacionam o tempo de atividade da estação com o tempo de utilização em sessões de usuários, para permitir mensurar o total de tempo que os computadores permaneceram ligados e não foram utilizados. Estes relatórios oferecem base para tomada de decisão, como por exemplo, em economia de energia, uma ociosidade menor resulta num possível aproveitamento mais adequado da energia consumida pelos computadores.

Em contrapartida outro formato de relatório de ociosidade é oferecido pelo novo console WEB do sistema SDMR, este relatório utiliza como parâmetro de relacionamento o tempo total que os computadores estão disponíveis para uso e quanto foram efetivamente utilizados. A seguir estes tipos de relatórios são gerados sobre os agrupamentos disponíveis a fim de permitir a análise de ociosidade específica.

4.8.3 Ociosidade relacionando tempo total com sessões

O relatório de ociosidade relacionando o tempo total de disponibilidade dos computadores com o tempo total de sessões de usuários foi gerado utilizando um intervalo específico de tempo de um determinado dia, permitindo avaliar assim o aproveitamento de uso num período de aula.

Foram gerados relatórios utilizando os três agrupamentos disponíveis: geral, por laboratório e por computador, e utilizando o intervalo de tempo das 19:10 as 22:30 do dia 9 de novembro de 2010, definido como turno da noite/aula.

O primeiro relatório é visualizado na Figura 50, onde observa-se que o percentual total de ociosidade é de 55,98%. Este percentual leva em conta os 213 computadores cadastrados, ignorando-se os localizados na coordenação, e o total de horas compreendido entre as 19:10 e 22:30, totalizando 710 horas de tempo disponível e 313,17 horas de tempo utilizado em sessões. Um percentual ideal de ociosidade poderia ser o mais próximo possível de 0%, para que assim os recursos fossem aproveitados em sua totalidade. No entanto, existem variáveis incidentes sobre a utilização dos computadores que devem ser consideradas, como a utilização de computador pessoal, o término da aula antes do horário habitual ou a saída prematura em função de pesquisas na biblioteca.

Relatório de ociosidade geral em horas		
Período de 09/11/2010 19:10 a 09/11/2010 22:30		
213 computadores selecionados	Intervalo de 3,33 horas	Tempo total de 710,00 horas
Total de horas em tempo disponível	**Total de horas em sessões**	**Percentual de ociosidade**
710,00	313,17	55,89 %

Figura 50 Relatório de ociosidade geral no turno da noite relacionando tempo total com sessões

Para que a análise seja mais específica o agrupamento por laboratório pode ser selecionado. Na Figura 51 o relatório foi gerado utilizando-se o mesmo intervalo de tempo anterior, modificando-se apenas o agrupamento para “Por Laboratório”.

Observa-se na Figura 51 que os valores de ociosidade são mais altos quando relaciona-se o tempo total com as sessões, do que quando relaciona-se o tempo de atividade, conforme a Figura 49 da sessão anterior. E verifica-se também que o percentual de ociosidade tende a ficar na faixa dos 50%, o que permite identificar uma tendência de utilização, equivalente a ociosidade geral obtida na Figura 50.

Outro ponto importante a ser observado é o percentual de ociosidade do laboratório 415-11, que no relatório de tempo ativo relacionado com sessão foi o menor com 1,85%, e no relatório de tempo total relacionado com sessão é o maior com 73,36%. Isto indica que mesmo apresentando o melhor uso em questões de tempo ativo, apresenta o pior uso no aproveitamento da disponibilidade dos recursos neste dia e turno específico, e entre os laboratórios monitorados.

Relatório de ociosidade por laboratório

Período de 09/11/2010 19:10 a 09/11/2010 22:30

Laboratório	Calculo	Total de horas disponivel	Total de horas em sessões	Percentual de ociosidade
415-11	25 hosts X 3,33 horas	83,33	22,20	73,36 %
102-7	25 hosts X 3,33 horas	83,33	36,62	56,06 %
105-7	25 hosts X 3,33 horas	83,33	38,37	53,96 %
103-7	25 hosts X 3,33 horas	83,33	54,15	35,02 %
413-11	26 hosts X 3,33 horas	86,67	36,47	57,92 %
101-11	25 hosts X 3,33 horas	83,33	35,97	56,84 %
101-7	31 hosts X 3,33 horas	103,33	38,50	62,74 %
104-7	31 hosts X 3,33 horas	103,33	47,50	54,03 %

Figura 51 Relatório de ociosidade por laboratório no turno da noite relacionando tempo total com sessões

Analisando o caso específico do laboratório 415-11 foram gerados dois relatórios de ociosidade relacionando tempo total disponível com tempo utilizado em sessões de usuários no intervalo de tempo das 19:10 as 22:30 em duas terças-feiras, para analisar o comportamento em duas noites em que a mesma disciplina ocorreu neste laboratório.

É possível verificar na Tabela 17, que traz os dados selecionados a partir dos relatórios gerados, que o número de computadores utilizados nesta aula é baixo, no entanto o tempo médio de uso é alto, assim como a sessão mais longa, visto que o tempo total disponível é de 3.33 horas. Neste contexto verifica-se que a variável que determinou o percentual alto de ociosidade está relacionada ao número reduzido de computadores utilizados.

Tabela 17 Tabela com dados de utilização do laboratório 415-11 no turno da noite

Dia	Intervalo de horas	Computadores utilizados	Computadores não utilizados	Tempo médio de uso	Sessão mais longa
26/10/2010	19:10 as 22:30	8	17	2,77 horas	3,15 horas
09/11/2010	19:10 as 22:30	11	14	2,51horas	3,05 horas

Os relatórios oferecidos na nova implementação do console WEB do sistema SDMR disponibilizam a definição de parâmetros de agrupamento, filtros e período de tempo que permitem ao gerente diversas análises dos dados coletados. É possível obter dados sobre a utilização por curso em longos períodos de tempo, como obter informações de uso de determinados computadores de um laboratório a fim de identificar um comportamento de uso.

Nos relatórios apresentados foi possível identificar o curso que mais fez uso dos recursos dos Laboratórios de Informática em função das sessões dos usuários. Os turnos e laboratórios onde ocorrem o maior percentual de ociosidade, assim como analisar um laboratório específico num determinado dia da semana e criar seu perfil de utilização. A sessão seguinte apresenta a análise do impacto causado pelo sistema SDMR.

4.9 Análise de impacto

Esta sessão traz resultados da análise de impacto na utilização de recursos pelo agente e coletor nas estações onde o sistema SDMR foi implantado. Num primeiro momento é avaliado o impacto do agente SDMR em função da utilização dos recursos de processador e memória do agente em execução, e em seguida é analisado o impacto dos coletores instalados para coleta das informações do sistema. As informações de utilização de processador e memória foram obtidas na base de dados do sistema SDMR através do console WEB. As informações de utilização de rede foram obtidas com a utilização da ferramenta Wireshark[7].

4.9.1 Impacto do agente SDMR

A avaliação do impacto do agente SDMR nas estações monitoradas consiste em obter o percentual de utilização dos recursos de processador e memória utilizados por sua execução. Foram obtidos e tabelados dados dos agentes em execução em seis computadores de cada um dos sistemas operacionais suportados e em três configurações de hardware distintas, que podem ser visualizadas na Tabela 18.

Tabela 18 Tabela com configurações de hardware para avaliação do impacto dos agentes

Tipo Hardware	Processador	Memória RAM	HD
1	Intel Pentium IV 2.66 GHz	1GB	80GB
2	Intel Pentium D 2.8 GHz	1GB	80GB
3	Intel Pentium IV 2.26 GHz	1GB	80GB

[7] Wireshark – software utilizado para análise do tráfego de rede. Mais informações em: <http://www.wireshark.org/>

As avaliações realizadas foram feitas através da obtenção de registros de tempo ativo de estações e sessões de usuário com tempo total aproximado de três horas, buscando um período substancial de utilização do computador.

Os dados obtidos do agente em execução no sistema operacional Linux podem ser visualizados na Tabela 19. Observa-se que o percentual médio de utilização do processador no período analisado foi de 0,13 e de memória de 0,36. O percentual de utilização de memória sofre uma variação mínima entre as estações e os tipos de *hardwares* analisados, apresentando desta forma um comportamento equilibrado. O uso do processador sofreu uma variação percentual um pouco maior, que pode estar relacionada com o número de processos em execução em cada estação, mas de forma geral manteve próximo da média em todas estações.

Tabela 19 Informações de utilização de recursos do agente em execução no S.O. Linux

Estação	**Data**	**% Uso de CPU**	**% Uso de Memória**	**Total de Atividade**	**Total da Sessão**	**S.O.**	**Tipo de Hardware**
lab12-14	09/11/10	0,13%	0,36%	03:00:02	02:59:12	Linux	1
lab12-18	08/11/10	0,11%	0,36%	02:55:10	02:54:18	Linux	1
lab21-20	09/11/10	0,11%	0,37%	03:04:09	03:03:29	Linux	2
lab21-22	09/11/10	0,10%	0,36%	03:20:02	03:18:20	Linux	2
lab5-11	08/11/10	0,15%	0,36%	03:11:29	03:09:19	Linux	3
lab5-24	08/11/10	0,19%	0,37%	03:12:09	03:11:21	Linux	3
	Média	**0,13%**	**0,36%**				

Os resultados obtidos na avaliação de impacto da execução do agente no sistema operacional Windows podem ser observados na Tabela 20. Observa-se que o percentual médio de utilização do processador é de 0,08% e da memória é de 1,06%. A variação do percentual de utilização do processador entre as estações e os diferentes tipos de *hardware* sofre uma pequena variação, mas não muito distante da média, assim como a utilização da memória, que fica próxima da média de 1,06% em todas as estações.

Tabela 20 Informações de utilização de recursos do agente em execução no S.O. Windows

Estação	Data	% Uso de CPU	% Uso de Memória	Total de Atividade	Total da Sessão	S.O.	Tipo de Hardware
Lab12-13	09/11/10	0,12%	1,05%	03:07:31	03:05:09	Windows	1
Lab12-27	09/11/10	0,08%	1,13%	03:24:26	03:23:32	Windows	1
lab21-17	08/11/10	0,11%	1,07%	03:07:53	03:26:11	Windows	2
lab21-20	08/11/10	0,06%	1,03%	03:02:07	03:00:10	Windows	2
Lab5-23	09/11/10	0,08%	1,13%	03:33:31	03:30:53	Windows	2
lab5-27	09/11/10	0,06%	0,96%	03:14:30	03:13:10	Windows	3
	Média	**0,08%**	**1,06%**				

Observa-se que a utilização do recurso processador do agente em execução nos dois sistemas operacionais possui valores semelhantes entre alguns tipos de *hardware*, porém na média o comportamento em Windows causa um impacto menor. Na utilização da memória verifica-se uma disparidade maior entre os dois sistemas operacionais, em Windows o valor utilizado em média é três vezes maior que em Linux. Este resultado era esperado já que algumas funções de liberação de memória foram desabilitadas no agente Windows em prol da estabilidade de execução e possibilidade de captura de informações.

A seguir serão apresentados dois gráficos onde é possível visualizar o comportamento da utilização de processador e memória pelo agente nos dois sistemas operacionais suportados, em execução de forma contínua num período de tempo mais longo.

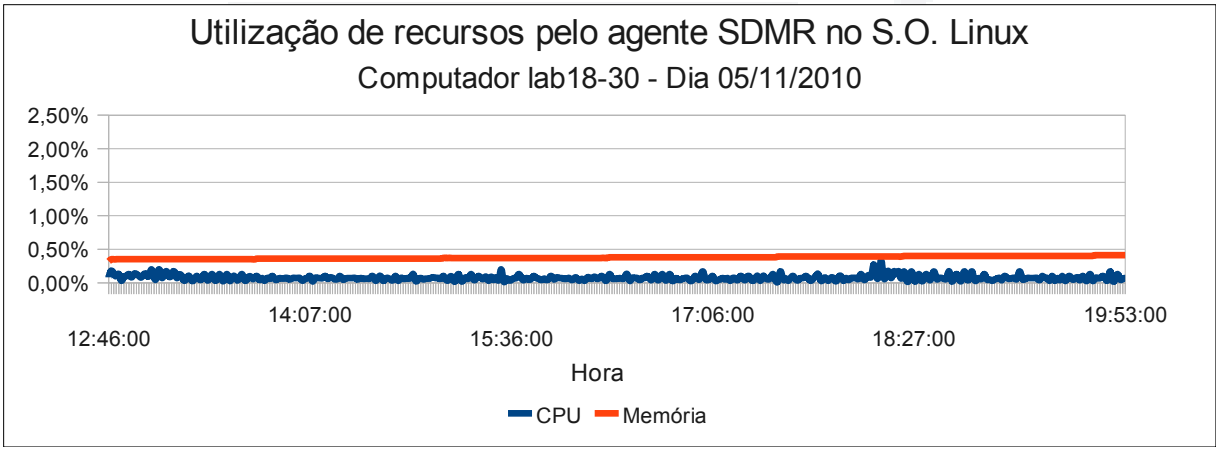


Figura 52 Utilização de recursos pelo agente SDMR no S.O. Linux

O gráfico da Figura 52 ilustra a utilização de recursos de processador e memória pelo agente SDMR em execução no sistema operacional Linux. Os dados foram obtidos do

computador lab18-10, que possui o tipo de *hardware* 1, no dia 05 de novembro de 2010. O total de atividade da estação foi de 7 horas e 10 minutos, iniciando as 12:46 e terminando as 19:56. Durante este período duas sessões de usuários foram registradas num total de 43 minutos de utilização. Observa-se um crescimento da utilização de memória mínimo, na faixa de 0,01% a cada 50 minutos aproximadamente, resultando num consumo médio de 0,38%. A utilização média do processador foi de 0,07%, apresentando alguns pontos de pico, o maior deles na faixa de 0,33%, apresentando também percentuais próximos de zero.

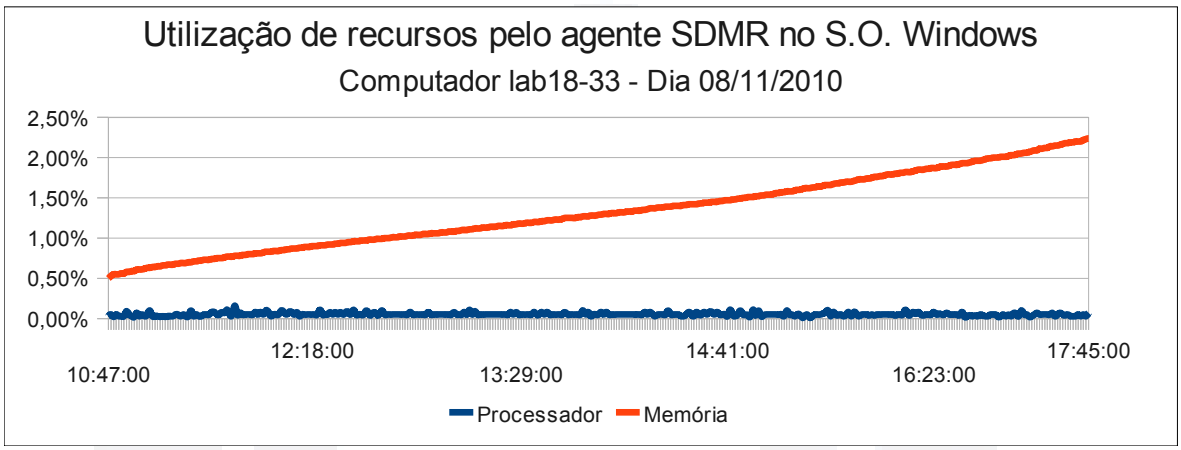


Figura 53 Utilização de recursos pelo agente SDMR no S.O. Windows

No gráfico da Figura 53 pode-se observar a utilização de recursos pelo agente SDMR no S.O. Windows obtidos do computador lab18-33, que possui o tipo de *hardware* 1, no dia 08 de novembro de 2010. O total de atividade da estação foi de 7 horas e 4 minutos, iniciando as 10:44 e terminando as 17:48, durante este período três sessões de usuários foram registradas totalizando 1 hora e 16 minutos de utilização. Observa-se que a utilização do processador não apresentou grandes variações de percentual de utilização, tendo como média o valor de 0,05% e um pico de 0.16%. A memória apresentou um crescimento linear, alcançando o pico de 2,24%. A média de utilização da memória foi de 1,33%.

Analisando os dois cenários verifica-se que a utilização do processador do agente SDMR nos dois sistemas operacionais não apresentou variações muito significativas, tendo como média de utilização valores próximos, com 0,05% em Windows e 0,07% em Linux. A utilização de memória apresentou resultados distintos, em Linux um crescimento mínimo foi observado, num total de 0,06% ao longo das 7 horas de execução. Em Windows, como já era esperado e comentado anteriormente, o percentual de utilização de memória cresceu de forma mais acentuada, atingindo o valor de 2,24%, apresentando um crescimento linear de 0,27%, em relação ao total de memória, a cada hora de execução. A seguir é apresentado o impacto causado pelo coletor do sistema.

4.9.2 Impacto do Coletor

A avaliação de impacto do coletor é feita através da obtenção dos percentuais de utilização de processador e memória das nove instâncias do processo coletor executadas. Cada coletor é responsável pela coleta de um laboratório e dos computadores que o compõe.

Tabela 21 Utilização do processador pelo coletor

Coletor	**% de CPU em 11/11/2010**	**% de CPU em 12/11/2010**
(coletord1)	0,10%	0,10%
(coletord2)	0,31%	0,12%
(coletord3)	0,49%	0,52%
(coletord4)	0,41%	0,21%
(coletord5)	0,23%	0,13%
(coletord6)	0,65%	0,02%
(coletord7)	0,36%	0,31%
(coletord8)	0,33%	0,47%
(coletord9)	0,74%	0,60%
Total médio de utilização	**3,63%**	**2,48%**

Na Tabela 21 pode-se observar os percentuais médios de utilização do processador (CPU) nos dias 11 e 12 de novembro de 2010. A soma dos percentuais médios resulta no total médio de utilização do processador em cada um dos dias, lembrando que estão sendo monitorados 216 computadores. Verifica-se uma variação de utilização média entre os coletores, esta situação ocorre em função do número de computadores em atividade a serem monitorados por cada coletor, mais adiante serão apresentados gráficos que auxiliam na constatação desta afirmação.

Tabela 22 Utilização de memória pelo coletor

Coletor	% de Memória em 11/11/2010	% de Memória em 12/11/2010
(coletord1)	0,27%	0,26%
(coletord2)	0,27%	0,26%
(coletord3)	0,27%	0,27%
(coletord4)	0,26%	0,24%
(coletord5)	0,26%	0,25%
(coletord6)	0,28%	0,27%
(coletord7)	0,26%	0,26%
(coletord8)	0,27%	0,27%
(coletord9)	0,28%	0,27%
Total médio de utilização	**2,43%**	**2,34%**

A Tabela 22 traz as informações de utilização da memória por cada coletor nos dois dias analisados. Verifica-se que os valores são próximos em todos os coletores, não variando de forma significativa em função do número de estações ativas em monitoramento.

Tabela 23 Utilização de processador pelo coletor dividido por turnos do dia 12/11/2010

Coletor	00:00 – 08:30	08:30 – 11:40	11:40 – 13:30	13:30 – 16:50	16:50 – 19:10	19:10 – 22:30	22:30 – 23:59
(coletord1)	0,06%	0,14%	0,14%	0,17%	0,12%	0,06%	0,06%
(coletord2)	0,02%	0,02%	0,02%	0,02%	0,02%	0,21%	0,02%
(coletord3)	0,04%	0,44%	0,26%	0,02%	0,11%	1,12%	0,02%
(coletord4)	0,02%	0,02%	0,02%	0,02%	0,08%	0,36%	0,02%
(coletord5)	0,02%	0,02%	0,02%	0,02%	0,05%	0,33%	0,02%
(coletord6)	0,02%	0,02%	0,02%	0,02%	0,02%	0,02%	0,02%
(coletord7)	0,02%	0,02%	0,02%	0,02%	0,12%	0,61%	0,02%
(coletord8)	0,09%	0,40%	0,31%	0,20%	0,37%	1,06%	0,04%
(coletord9)	0,05%	0,52%	0,81%	0,93%	0,71%	0,72%	0,02%
Total médio por turno	**0,33%**	**1,58%**	**1,61%**	**1,39%**	**1,59%**	**4,49%**	**0,22%**

Para mapear a variação do percentual médio de utilização do processador em função dos turnos de atividade, os dados foram agrupados em intervalos de tempo, conforme apresentados na Tabela 23. Verifica-se que o percentual médio de utilização do processador

varia dependendo do intervalo de tempo analisado, sendo que esta variação tem origem no número de computadores ativos, ou seja, ligados neste intervalo de tempo. Esta relação pode ser observada comparando a Figura 54, que ilustra o número de computadores ligados e sessões de usuários ao longo do dia 12 de novembro de 2010, com a Figura 55 que traça uma linha com o percentual de utilização do processador pelos coletores neste mesmo intervalo de tempo. Analisando os dois gráficos é possível visualizar que a variação é semelhante entre as linhas de máquinas ligadas e de percentual de uso do processador, confirmando esta relação.

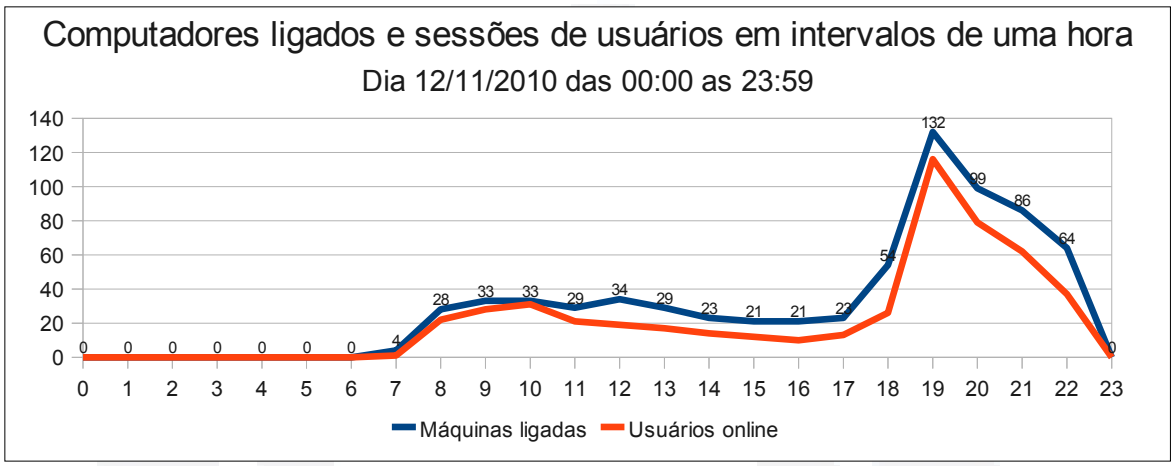


Figura 54 Gráfico com computadores ligados e sessões no dia 12/11/2010

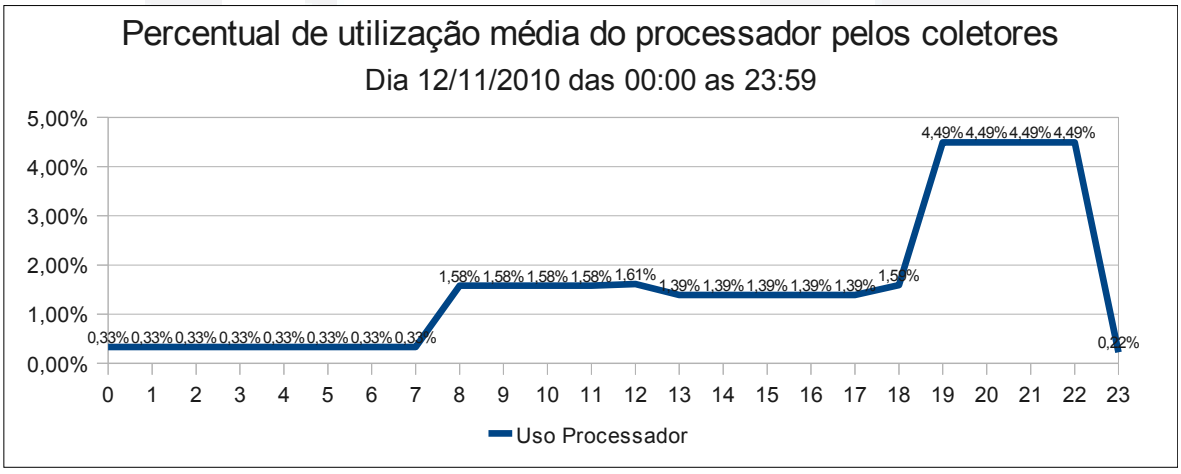


Figura 55 Gráfico em linha com o percentual médio de utilização do processador pelos coletores no dia 12/11/2010

A fim de avaliar o impacto de rede causado pelo sistema SDMR foram obtidas informações do tráfego de rede na estação onde os coletores foram executados. O software Wireshark foi utilizado para obter tais informações no dia 19 de novembro de 2010, em 10 coletas de 30 minutos, totalizando 5 horas, com início no horário das 16:17 hs e término em 21:17 hs. A Tabela 24 apresenta as informações capturadas.

Tabela 24 Informações do tráfego de rede capturados no computador executando os coletores no dia 19/11/2010

Período	Intervalo	Total de pacotes SNMP trafegados	Total de *bytes* SNMP trafegados	Média de tráfego de pacotes SNMP (*Mbits/s*)	Percentual de ocupação da banda disponível (100*MBits/s*)
1	16:17 - 16:47	128.776	12.120.308	0,05 *Mbits/s*	0,05%
2	16:47 - 17:17	222.504	20.987.412	0,09 *Mbits/s*	0,09%
3	17:17 - 17:47	188.014	17.701.798	0,08 *Mbits/s*	0,08%
4	17:47 - 18:17	148.931	14.040.520	0,06 *Mbits/s*	0,06%
5	18:17 - 18:47	197.264	18.575.881	0,08 *Mbits/s*	0,08%
6	18:47 - 19:17	306.783	28.954.608	0,13 *Mbits/s*	0,13%
7	19:17 - 19:47	555.917	52.491.373	0,23 *Mbits/s*	0,23%
8	19:47 - 20:17	581.722	54.899.487	0,24 *Mbits/s*	0,24%
9	20:17 - 20:47	548.047	51.712.921	0,23 *Mbits/s*	0,23%
10	20:47 - 21:17	470.766	44.376.655	0,19 *Mbits/s*	0,19%

Observa-se na Tabela 24 que a quantidade de pacotes e *bytes* do protocolo SNMP trafegados sofre uma pequena variação nos períodos de 1 a 4, porém a partir do período 5 observa-se um crescimento mais acentuado nos valores totais, aumentando consequentemente a média de tráfego em *Mbits/s*. É possível observar ainda que mesmo no horário de maior tráfego o percentual da largura de banda utilizado não ultrapassa os 0,24%, tendo como referência a rede disponível com largura de banda de 100 *Mbits/s*.

A varição crescente do tráfego observada a partir das 18:47 hs apresenta o mesmo comportamento do percentual de utilização do processador, que varia em função do número de computadores monitorados ligados. Os gráficos apresentados na Figura 56 e na Figura 57 permitem observar a relação deste comportamento, verifica-se que conforme o número de computadores ligados varia, a média de tráfego de rede do protocolo SNMP varia no mesmo sentido.

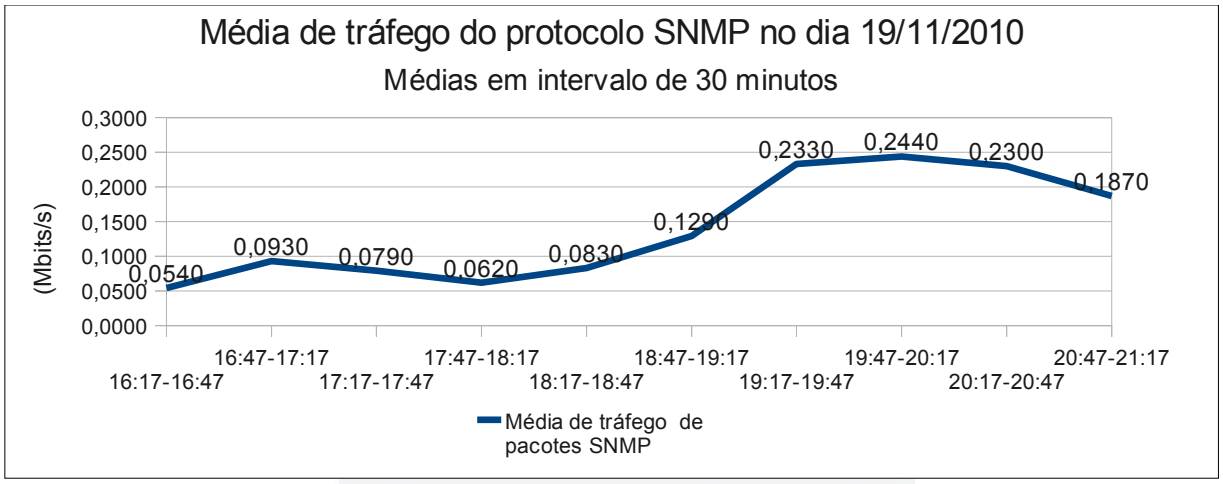


Figura 56 Gráfico em linha da média de tráfego do protocolo SNMP no dia 19/11/2010

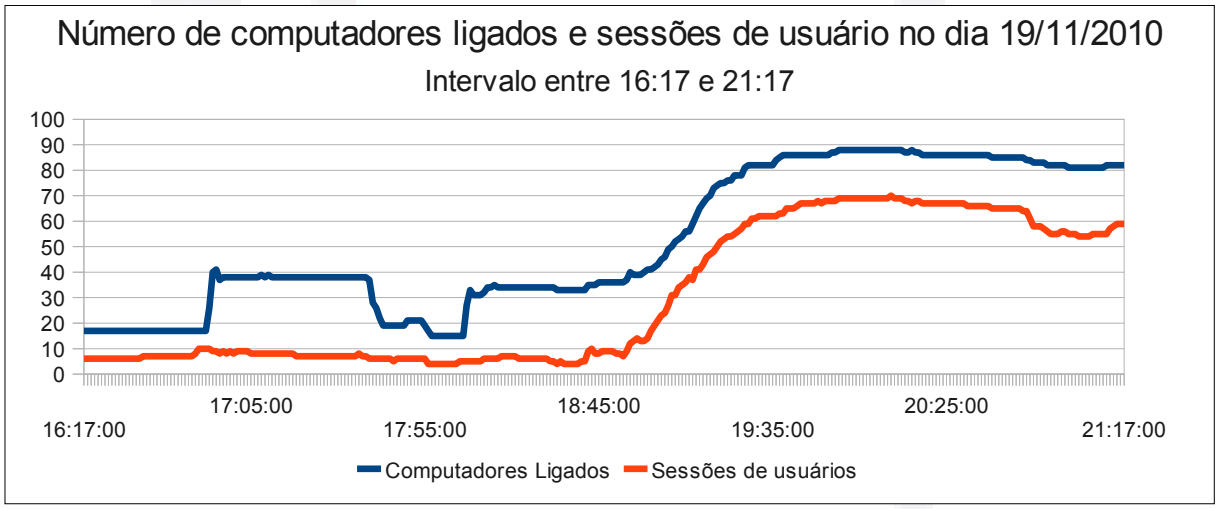


Figura 57 Gráfico em linha do número de computadores ligados no dia 19/11/2010

Por fim , observa-se que o percentual de utilização do tráfego SNMP em relação do total da banda disponível atinge 0,24% quando o número de computadores ligados é aproximadamente de 90. Uma projeção num cenário com 500 computadores monitorados atingiria o percentual de 1,33% de uma rede com largura de banda de 100 *Mbits/s*. Este impacto total varia também em relação as configurações de tempo de captura de informações do agente. Um impacto menor pode ser alcançado aumentando os tempos de captura, diminuindo assim a granularidade das informações obtidas.

Esta sessão analisou os impactos causados pela implantação e execução do sistema SDMR no cenário aplicado neste trabalho. Foram apresentados dados referentes ao impacto causado pelos agentes nas estações monitoradas e pelos múltiplos coletores na estação coletora, assim como o tráfego geral de rede ocasionado nesta estação. A seguir são

apresentadas as constatações sobre os quesitos de validação de robustez e escalabilidade observados no desenvolvimento e implantação do sistema SDMR.

4.10 Robustez

A validação da robustez do sistema esteve presente durante todo processo de desenvolvimento, experimentação e implantação do sistema. Verificou-se inicialmente que o agente Linux do sistema SDMR tinha sua execução terminada inesperadamente, e que estava relacionada com a inicialização e término acentuado de processos, que normalmente ocorria quando o computador passava a ser usado por um usuário. Para constatar que a origem do problema estava no formato original do agente o mesmo foi implantado em dois módulos, um que executava o agente em seu formato original e outro que executava as novas implementações.

Após constatado o problema buscou-se a estabilidade do agente com o tratamento dos sinais recebidos pelo processo, ignorando a solicitação do sistema operacional para que o processo fosse fechado. Esta solução não trouxe resultado, porém permitiu detectar que o problema tinha origem na tentativa de acesso a um endereço de memória inválido. Em seguida realizou-se a revisão do código fonte do agente em busca do trecho que originava o problema. O mesmo foi encontrado e corrigido conforme já apresentado na Experimentação I.

Durante o processo de busca da estabilidade do agente criou-se a variável denominada "*Agente Ativo*", que permitiu e ainda permite que o coletor verifique remotamente se o agente está em execução ou não, e ainda, se está rodando na versão mais atual. Ao detectar algum problema, avisos são registrados e podem ser visualizados pelo console WEB.

O agente para o S.O. Windows foi desenvolvido em um intervalo de tempo reduzido, após a constatação de sua necessidade para que os dados coletados tivessem algum valor mais consistente para análise. O agente apresentou alguns problemas de instabilidade durante sua execução, que foram atenuados com a remoção de trechos de código que tratavam a liberação de memória de estruturas não mais utilizadas.

4.11 Escalabilidade

A escalabilidade do sistema foi alcançada através da utilização de vários coletores na arquitetura de execução denominada *multi-process*. Cada coletor executado foi responsabilizado pela coleta das informações dos computadores de um determinado laboratório.

Outro fator modificado foi em relação ao tempo total de coleta necessário no caso de haver um grande número de máquinas desligadas. Para atenuar esta condição duas variáveis foram criadas e a gerência do processo de coleta modificado. Foram criadas as condições que permitem ao coletor colocar os computadores detectados como desligados em um estado de espera indicado pela variável *STANDBY* e realizar uma nova tentativa de coleta após um determinado número de ciclos, indicado pela variável *RETRY*.

A utilização de apenas um coletor no seu formato original afetaria a configuração do tempo do ciclo de coleta, impedindo que a configuração desejada fosse realmente alcançada. Esta situação é decorrente do tempo gasto na tentativa de coleta de uma estação desligada, em função do *timeout* do protocolo SNMP.

Para solucionar esta situação o sistema foi implantado com a múltipla execução de coletores, implicando na gerência destes processos pelo sistema operacional. Em contrapartida a coleta em intervalos de tempo não muito longos ou muito curtos equilibra a utilização de memória do processador pelos agentes, conforme já foi evidenciado nos cenários propostos por Stoll(2008).

Este capítulo apresentou os resultados obtidos durante a execução deste trabalho, as experimentações realizadas, a análise das informações disponibilizadas pela reestruturação do sistema, o impacto do mesmo sobre o ambiente e constatações sobre escalabilidade e robustez. O capítulo seguinte apresenta as conclusões finais deste trabalho.

5 CONCLUSÃO

O presente trabalho teve por objetivo implantar o sistema SDMR nos Laboratórios de Informática da UNIVATES, buscando sua validação num ambiente real através da análise das informações coletadas pelo sistema. Para disponibilizar um ambiente gerencial que ofereça informações que possam auxiliar na tomada de decisão, o sistema teve suas funcionalidades complementadas. As novas informações oferecidas são relativas ao tempo de atividade dos computadores, ao tempo de utilização por usuário e dados referentes à utilização da memória física disponível. Para que estas informações pudessem ser analisadas o trabalho englobou a reestruturação do console de administração, oferecendo telas de relatórios de tempo de uso, relatórios de ociosidade e cadastros de estações e componentes.

Foram apresentados os conceitos de funcionamento da ferramenta SDMR, os recursos que por ela eram oferecidos e o embasamento teórico para que as novas informações pudessem ser obtidas dos computadores monitorados.

A implementação deste trabalho englobou a reestruturação de todos os componentes do sistema SDMR. O agente foi modificado para suportar a captura das novas informações do sistema operacional e publicá-las nas novas tabelas adicionadas na estrutura MIB. O componente coletor foi modificado para suportar a coleta das novas informações utilizando o protocolo SNMP, assim como foi reestruturado seu processo de coleta com a adição de parâmetros de controle. Novas tabelas foram adicionadas ao banco de dados para que as informações coletadas pelo sistema fossem armazenadas de forma consistente. Reestruturou-se também o console de administração denominado Console *WEB*, oferecendo novas telas de relatórios, consultas e cadastros. Ao longo do desenvolvimento o sistema foi avaliado em questões de robustez e escalabilidade e, através de sua reestruturação, foi possível aplicá-lo nos computadores monitorados e obter as informações almejadas.

Com as experimentações realizadas e a implantação do sistema foi possível validar sua nova estrutura analisando as informações obtidas através da geração de relatórios utilizando a nova estrutura do console WEB. Verificou-se que o sistema foi capaz de obter as informações já disponibilizadas pela versão original do sistema SDMR, assim como as novas informações agregadas. E através de sua análise pode-se chegar a diversas conclusões sobre a utilização dos recursos dos Laboratórios de Informática da UNIVATES, e propor ao longo das análises possíveis medidas que pudessem oferecer uma melhor utilização destes recursos.

O trabalho desenvolvido contribui de forma geral para a continuidade de uma ferramenta de análise de utilização de recursos através de sua implantação num ambiente real de grande escala, com a obtenção de informações gerenciais com possibilidade de utilização

para tomada de decisão na realocação e aquisição de recursos. Propostas de trabalhos futuros apresentadas nos trabalhos anteriores foram avaliadas e aplicadas, como obtenção do total de memória e complementação do console.

Por fim, este trabalho uniu e aplicou os conhecimentos adquiridos durante o curso e atividade profissional relacionada, não só na avaliação de uma ferramenta, mas na sua complementação através de codificação, proporcionando um ambiente a ser utilizado por gestores e administradores de recursos computacionais.

5.1 Trabalhos futuros

Como trabalhos futuros para continuidade deste trabalho pretende-se a realização das seguintes atividades:

a) Reavaliar o agente Windows buscando remover os problemas de instabilidade e de consumo de memória.
b) Avaliar a reestruturação do agente Windows removendo sua dependência da plataforma CygWin. A ferramenta CygWin existe com o intuito da portabilidade de sistemas, mas o impacto causado por uma aplicação independente pode ser menos oneroso ao sistema como um todo.
c) Através da Mineração dos Dados proporcionar novos relatórios que possibilitem a análise mais profunda dos processos, agrupando-os por categorias de software, como: Editores de texto, Navegadores WEB, a fim de traçar um perfil de utilização.
d) Gráficos de utilização de recursos ao longo do tempo.
e) Implementação de mais alertas. Disponibilizar alertas para detectar alta utilização de CPU, memória ou disco rígido.
f) Verificar viabilidade de tornar o agente mais ativo, fazendo que ele informe ao coletor seu estado. Verificar os contras desta modificação, como por exemplo, a carga excessiva que pode ser gerada no coletor pelo acesso simultâneo de vários agentes. Tomar cuidado para não perder o conceito original do sistema que utiliza o protocolo SNMP, avaliar a utilização de *traps* SNMP. Na estrutura atual os agentes não precisam conhecer o coletor ou coletores, modificar este conceito pode tornar sua manutenção mais onerosa.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, V. A. F. **Sistemas de Redes Robustos: Modelos e Ferramentas .** Departamento de Ciência da Computação Universidade Federal de Minas Gerais - MG, 2007, .36 p. Disponível em <http://www.gta.ufrj.br/rebu/arquivos/2008-REBU-0612228926373803_01-1.pdf>. Acesso em <24 de maio de 2010>.

BIRMAN, Kenneth P. **Reliable distributed systems: technologies, Web services, and applications.** Ithaca, New York, U.S.A. Editora Springer, 2005, 688 p. ISBN 0387215093, 978038721509. Disponível parcialmente em <http://books.google.com.br/books?id=KeIENcC2BPwC&pg=PA4&dq=Distributed+Systems+scalability&as_brr=3&cd=2#v=onepage&q=Distributed%20Systems%20scalability&f=false>. Acesso em <16 de maio de 2010>.

BLACK, Tomas Lovis. **Comparação de ferramentas de gerenciamento de redes.** Instituto de Informática. Curso de Especialização em Tecnologias, Gerência e Segurança de Redes de Computadores. Universidade Federal do Rio Grande do Sul, Porto Alegre – RS, 2008, 64 p. Disponível em <http://hdl.handle.net/10183/15986>. Acesso em <22 de junho de 2010>.

BOVET, Daniel P. **Understanding the linux kernel.** Beijing, O Reilly, 2006.

CACTI. **Cacti: The Complete RRDTool-based Graphing Solution.** Disponível em <http://www.cacti.net/>. Acesso em <22 de maio de 2010>.

CETIC - Centro de Estudos sobre as Tecnologias da Informação e da Comunicação. **Pesquisa sobre o uso das Tecnologias da Informação e da Comunicação no Brasil - 2009**. Disponível em <http://www.cetic.br/empresas/2009/index.htm>. Acesso em <27 de maio de 2010>.

CORDEIRO, Daniel de A. **Estudo de escalabilidade de servidores baseados em eventos em sistemas multiprocessados: um estudo de caso completo .** Universidade de São Paulo. Instituto de Matemática e Estatística – São Paulo – SP, Dissertação apresentada para obtenção do grau de Mestre em Ciências, 2006, 103 p. Disponível em <http://grenoble.ime.usp.br/~gold/orientados/dissertacao-DanielCordeiro.pdf>. Acesso em <16 de maio de 2010>.

CYGWIN. **Cygwin Information and installation**. Disponível em <http://www.cygwin.com/>. Acesso em <10 de novembro de 2010>.

FERREIRA, Rubem E. **Linux** : guia do administrador do sistema . São Paulo, Novatec, 2003.

FIREBIRD. **Firebird Project**. Disponível em <http://www.firebirdsql.org/>. Acesso em <02 de junho de 2010>.

FONSECA, F. S. S. **Sistema de Tratamento de Arquivos de Logs.** Universidade Federal de Lavras. Departamento de Pós-graduação. Pós-graduaçã Lat Sensu em Administração de Redes Linux – Minas G erais, 2005, 59 p. Disponível em <http://www.ginux.ufla.br/node/92>. Acesso em <15 de maio de 2010>.

FOWLER, Martim. **Padrões de Arquiteturas de Aplicações Corporativas.** São Paulo – SP, Editora Bookman, 2006, 493 p. Disponível parcialmente em <http://books.google.com.br/books?id=vpHqYZcmeKsC&pg=PA30&dq=escalabilidade&as_brr=3&cd=2#v=onepage&q=escalabilidade&f=false>. Acesso em <16 de maio de 2010>.

LEITE, Silvio Luis. **Integrando ferramentas da software livre para gerenciamento e monitoração de redes locais.** Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação – Porto Alegre – RS, 2004, 109 p. Disponível em <http://www.lume.ufrgs.br/handle/10183/5968>. Acesso em <08 de maio de 2010>.

LANGASEK, Steve. **LucidLynx ReleaseNotes**. Disponível em <https://wiki.ubuntu.com/LucidLynx/ReleaseNotes>. Acesso em <26 de maio de 2010>.

LMP. Linux Man Pages. Atual. em 2010. Disponível em <http://www.kernel.org/doc/man-pages/>. Acesso em <15 de maio de 2010>.

MITCHELL, Mark; OLDHAM, Jeffrey; SAMUEL, Alex. **Advanced Linux programming**. 1 ed. New Riders Publishing, 2001. Diponível em: <http://www.advancedlinuxprogramming.com>. Acesso em <16 de maio de 2010>.

MICROSOFT. **Como exibir e gerenciar os logs de evento em Visualizar eventos no Windows XP.** Disponível em <http://support.microsoft.com/kb/308427/pt-br>. Acesso em <14 de dezembro de 2010>.

PAI, V. S.; DRUSCHEL, P.; ZWAENEPOEL, W. **Flash: An efficient and portable Web server.** USENIX 1999 Annual Technical Conference, California, EUA, 1999, 14 p. Disponível em <http://www.usenix.org/event/usenix99/full_papers/pai/pai.pdf>. Acesso em <20 de maio de 2010>.

RIBEIRO, Uira. **Sistemas distribuídos: desenvolvendo aplicações de alta performance no Linux** . Rio de Janeiro, Axcel, 2005.

SPEZIA, Jamiel. **Sistema distribuído para monitorar o uso dos recursos de hardware e software em estações de trabalho GNU/LINUX** . 2007. 157 f. Monografia (Bacharel em Engenharia da Computação) - Curso de Engenharia da Computação), Univates, Lajeado, RS, 2007.

STOLL, Rudimar. **Sistema distribuído para monitorar o uso dos recursos de hardware e software em computadores com GNU/Linux e Windows**. 2008. 121 f. Monografia (Bacharel em Sistemas de Informação) - Curso de Sistemas de Informação), Univates, Lajeado, RS, 2008.

TANENBAUM, Andrew S. **Redes de Computadores**, 4a edição, Rio de Janeiro: Campus, 1997.

UNIVATES. **Univates Tech . Laboratórios**. Disponível em <http://www.univates.br/labinfo>. Acesso em <06 de maio de 2010>.

ZABBIX. **Homepage of Zabbix**. Disponível em <http://www.zabbix.com/>. Acesso em <22 de maio de 2010>.

ZENOSS. **Open Source Server and Monitoring Network.** Disponível em <http://www.zenoss.com/>. Acesso em <22 de maio de 2010>.