

Universidade do Minho
Departamento de Informática

Mestrado Integrado em Engenharia Informática

Gestão de Redes



Ficha de Trabalho Prático Nº2: Ferramenta de gestão SNMP para monitorização de Processos

a86617 Gonçalo Nogueira

Braga
Fevereiro, 2020

Conteúdo

1	Introdução	3
2	Objetivos	4
3	Estrutura do Projeto	5
4	Componentes do Projeto	6
4.1	Objetos da <i>MIB</i> necessários	6
4.2	SNMPManager	7
4.2.1	getProcessData	7
4.2.2	preencheHash	7
4.2.3	Cálculos dos valores de percentagem de CPU e RAM	8
4.2.4	Geração de logs	8
4.3	Interface	9
4.3.1	Manual de Funcionamento	9
4.3.2	Intervalos de pooling	9
4.3.3	Pesquisas nos logs do SNMPManager	9
4.4	Alarmes	10
5	Conclusão	11
6	Anexo	12
7	Bibliografia	14

Lista de Figuras

1	Esquema do projeto	5
2	Ficheiro logs dos Alarms	10
3	Interface inicial	12
4	Iniciando aplicação depois de introduzido IP/Port	12
5	Informações dos processos	13
6	Gráfico com CPU e RAM "usage" de um processo ao longo do tempo	13

1 Introdução

O presente relatório, realizado no âmbito da unidade curricular de Gestão de Redes, destina-se à apresentação de um conjunto de explicações acerca do trabalho realizado sobre a criação de uma ferramenta de gestão de processos bem como a memória *RAM* e utilização de *CPU* de cada um. Incide também sobre a gestão de alarmes no caso de algum processo atinja uma percentagem demasiado alta.

Numa fase primária, este relatório tem como propósito detalhar os objetivos que se propõe alcançar e a estrutura do projeto realizado. Seguidamente, será abordada a explicação dos três componentes principais desenvolvidos bem como os objetos utilizados das *MIBs*.

Para além disso, inclui também um manual de instruções, cuja respetiva finalidade será proporcionar uma utilização correta e navegação da interface criada para a aplicação de gestão criada. Por fim, o cumprimento dos objetivos será abordado na conclusão, bem como as dificuldades encontradas ao longo da realização do projeto.

2 Objetivos

Os principais objetivos propostos são a familiarização com a arquitetura *Internet-standard Network Management Framework* (INMF), com especial atenção ao *Simple Network Management Protocol* (SNMP) e às *Management Information Bases* (MIBs).

De forma mais concreta, pretende-se através do uso de *APIs SNMP* a construção de uma aplicação modular que monitorize informações sobre os processos a correr num determinado host e capaz da geração de alarmes.

3 Estrutura do Projeto

Tal como proposto o projeto divide-se em três componentes principais capazes de comunicar entre si através da geração de logs.

Primeiramente foi desenvolvido um módulo capaz de monitorizar os processos. Em seguida, foi criado outro módulo para a criação de um interface responsável pelas interações do utilizador. Por fim, foi desenvolvido um módulo que analisa as informações recolhidas, sendo capaz de gerar alarmes sobre a forma de email caso algum processo atinja certos valores de performance.

De salientar, ainda a ausência de ficheiros de configuração neste projeto, existindo apenas a possibilidade de o user indicar um endereço IP e a porta que pretende usar, sendo a porta 161 a preferível.

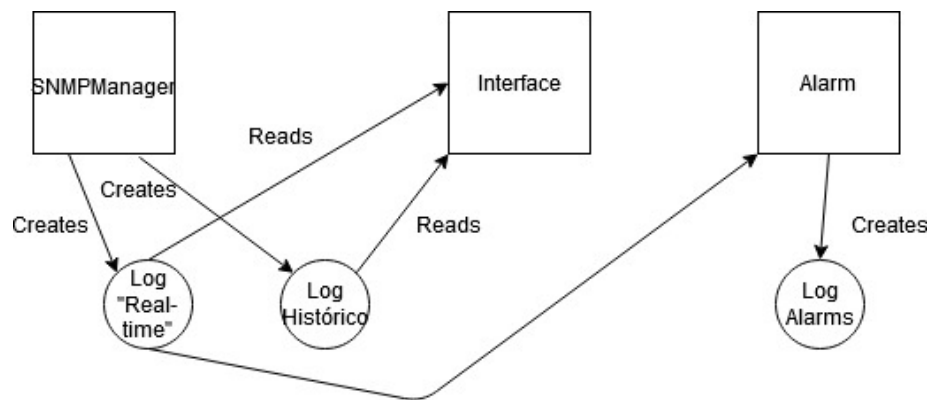


Figura 1: Esquema do projeto

4 Componentes do Projeto

4.1 Objetos da MIB necessários

Antes da construção dos vários componentes do projeto, é necessário efetuar uma seleção dos objetos da MIB que vão ser necessários para os mesmos, sendo esta escolha essencial, pois pode afetar as estratégias escolhidas para o desenvolvimento dos módulos.

Sendo assim, o objeto mais importante escolhido foi a **hrSWRunPerfTable** com o *OID 1.3.6.1.2.1.25.5.1* com a seguinte descrição:

The (conceptual) table of running software performance metrics.

Este objeto tem como filho **hrSWRunPerfEntry** com o *OID 1.3.6.1.2.1.25.5.1.1* que corresponde a cada entrada na tabela.

Consequentemente, cada entrada contém duas métricas sobre cada processo, **hrSWRunPerfCPU** com o *OID 1.3.6.1.2.1.25.5.1.1.1* e **hrSWRunPerfMem** com o *OID 1.3.6.1.2.1.25.5.1.1.2* com as respetivas descrições:

"The number of centi-seconds of the total system's CPU resources consumed by this process. Note that on a multi-processor system, this value may increment by more than one centi-second in one centi-second of real (wall clock) time."

"The total amount of real system memory allocated to this process."

Para a obtenção da lista de processos a correr no sistema foi usado o objeto **hrSWRunName** com o *OID 1.3.6.1.2.1.25.4.2.1.2* com a descrição:

"A textual description of this running piece of software, including the manufacturer, revision, and the name by which it is commonly known. If this software was installed locally, this should be the same string as used in the corresponding hrSWInstalledName."

Além disso, o objeto **memTotalReal** com o *OID 1.3.6.1.4.1.2021.4.5.0* foi usado para a obtenção do total da memória RAM na máquina, com a descrição:

"The total amount of real/physical memory installed on this host."

Por fim, o objeto **sysName** com o *OID 1.3.6.1.2.1.1.5* foi também utilizado para identificação na interface do nome do host que o utilizador está a monitorizar informação e tem a seguinte descrição:

"An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name. If the name is unknown, the value is the zero-length string."

4.2 SNMPManager

Tal como descrito na secção anterior, o primeiro módulo desenvolvido tem por objetivo a implementação de mecanismos capazes monitorizar os processos de um determinado host.

Para a criação deste módulo foi utilizada a API **SNMP4J** que permite o acesso às *MIBs* e posterior recolha da informação.

Esta classe contém como variáveis de instância um endereço do host, no qual se pretende recolher informação, bem como um *Map* que usa como chave o nome do processo, associando a cada um, os valores de *CPU* e de *RAM* utilizados e um objeto da classe *snmp* já definida na API utilizada.

Neste módulo estão implementados métodos para a utilização das funcionalidades já existentes no Net-SNMP como por exemplo, *snmpget* e *snmpwalk*.

Além dos métodos já conhecidos, o **getProcessData** recebe como parâmetro o nome do processo e obtém o seu index através de outro método (**getProcessIndexes**) utilizando o objeto **hrSWRunName** já descrito previamente. Após identificado o respetivo index, já é possível recolher os dados desse processo na tabela de métricas de performance.

Por fim, o método principal deste módulo é o **preencheHash**, responsável pelo preenchimento a *Map*, onde já se encontram os nomes de todos os processos, com os cálculos das métricas já feitos e associando cada processo aos seus respetivos valores.

4.2.1 getProcessData

Este método criado, recebe como parâmetro o nome do processo e tem como função devolver uma lista de **SNMPTriple**, constituído por um *OID*, o seu nome lexicográfico e o seu respetivo valor.

O algoritmo é simples e de forma sucinta verifica se o processo existe no objeto *hrSWRunName*. Caso não exista acaba logo e devolve uma lista vazia. Caso o processo exista, é encontrado o seu index que é utilizado para a obtenção dos valores na tabela de performances, acrescentando o seu index no fim dos *OIDs* de **hrSWRunPerfCPU** e **hrSWRunPerfMem**.

Sendo assim, caso tudo ocorra normalmente, é devolvida uma lista com dois **SNMPTriple** cada um para uma métrica diferente sobre o mesmo processo.

4.2.2 preencheHash

Este método, tem como argumentos um intervalo de *pooling* que representa o tempo entre recolha de informações, podendo este ser escolhido pelo utilizador e um *int* que sendo 1 representa a primeira amostra e 2 representa a segunda amostra para obtenção da percentagem de *CPU*.

O algoritmo percorre a *Map* pelas suas chaves e vai recolhendo a informação de cada processo usando o *getProcessData*. Depois de recolhida a informação de um processo, converte-se as strings resultantes para *Doubles*, que são guardadas num **SNMPPar** (cada *Double* representa uma métrica de performance) e posteriormente são feitos os cálculos dos valores de percentagem.

De salientar, que caso seja a primeira amostra, os dados do *CPU* não são tratados e são apenas guardados os *centi-seconds* recolhidos dos objetos.

4.2.3 Cálculos dos valores de percentagem de CPU e RAM

Relativamente à memória RAM, o cálculo da percentagem é bastante simples, sendo apenas necessário saber que o valor total de *RAM* disponível corresponde a cem por cento e também que o valor presente no objeto corresponde a X, fazendo uma regra de três simples obtemos a percentagem que determinado processo usa de memória.

O cálculo da percentagem de *CPU* é um pouco mais complexo, pois só é possível a sua obtenção com duas amostras, uma vez que o valor presente na *MIB* é o tempo que o processo consome do *CPU*, fazendo a diferença absoluta entre os valores das duas amostras e dividindo pelo intervalo de tempo entre as mesmas obtém-se a percentagem utilizada pelo processo.

4.2.4 Geração de logs

Este módulo gera dois ficheiros *logs*, um com os processos em execução atuais para a apresentação ao utilizador em "*real-time*" e outro com o histórico de todos os processos registados na ferramenta com vista a possibilitar ao utilizador uma observação do histórico de cada processo ao longo do tempo.

A estrutura deste *log* é bastante simples, contendo em cada linha o nome do processo, um valor de *CPU* e outro valor de *RAM*, separados por vírgulas para facilitar um posterior *parse* do ficheiro.

4.3 Interface

Este módulo é o responsável pela ordem como a ferramenta funciona bem como a interação com o utilizador.

Nesta componente é utilizada a API **jfree** para a criação de gráficos e **jframe** já incluída no IDE IntelliJ para a criação da interface.

4.3.1 Manual de Funcionamento

Para iniciar a ferramenta de gestão, o utilizador tem apenas de introduzir o endereço ip e a porta, escolher o tipo de pooling que pretende (sendo "high" o default) e pressionar o botão *Start*. Deve depois aguardar pela recolha de informações, que serão posteriormente visíveis podendo ser ordenadas por ordem alfabética dos processos ou pelas métricas.

Do lado direito da interface, encontra-se uma lista de processos guardados no histórico da ferramenta, podendo o utilizador introduzir o nome do processo que pretende visualizar o histórico e pressionando o botão *Search*, após isso é apresentado um gráfico com a evolução das métricas do processo pretendido ao longo do tempo.

Consultar o anexo para imagens da interface.

4.3.2 Intervalos de pooling

Após várias experiências e informações disponibilizadas pelos docentes, verificou-se que a tabela de métricas de performance dos processos atualiza apenas de trinta em trinta segundos. Com essa informação defini três possibilidades de intervalos pooling:

- High (Default) -> 30 segs (Manager) / 35 segs (Interface)
- Medium -> 60 segs (Manager) / 65 segs (Interface)
- Low -> 90 segs (Manager) / 95 segs (Interface)

Decidi colocar um intervalo de 5 segundos entre o manager para não correr o risco de a interface tentar ler o ficheiro logs enquanto o mesmo está a ser escrito pelo manager.

4.3.3 Pesquisas nos logs do SNMPManager

Relativamente ao ficheiro logs de processos em execução "real-time", este componente lê linha a linha todo o ficheiro, fazendo o parse de cada linha, separando cada uma pelas vírgulas e coloca toda a informação numa tabela da interface para observação por parte do utilizador.

Relativamente ao ficheiro logs de todo o histórico, o processo de leitura é semelhante ao anterior, mudando apenas a forma como é posteriormente guardado e apresentado ao utilizador. Para este caso, foi criada uma classe auxiliar **HashMapHist** contendo um *Map* onde a *key* é o nome do processo e o *value* uma lista de *SNMPPar* com as métricas de performance calculadas.

Sempre que um processo é lido do ficheiro e já se encontra no registo, os valores encontrados são convertidos num *SNMPPar* e adicionados à lista.

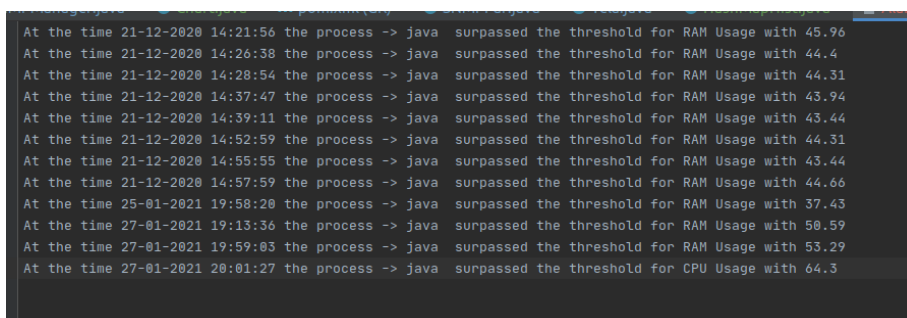
4.4 Alarmes

O último módulo desenvolvido baseia-se na geração de alarmes via email. O módulo vai analisando o ficheiro de logs de processos em execução sempre que o mesmo é atualizado pelo primeiro módulo e permite uma verificação de todos os processos em simultâneo apenas.

Foram definidos como valores de threshold cinquenta por cento, para ambas as métricas, ou seja sempre que se verificar que um processo usa metade da memória RAM ou do CPU, é gerado um alarme. Importante salientar que estes valores são fixos, ou seja, não permitem uma alteração por parte do utilizador, apenas podem ser alterados pelo administrador da ferramenta neste caso.

A API **javax.mail** foi utilizada pois permite de forma bastante simples e eficiente a criação e envio de um email.

Por fim, é também importante realçar que este módulo gera um log com a data e hora de cada alarme bem como o processo e o valor de utilização de recursos que utilizou para o despoletar do alarme.



```
At the time 21-12-2020 14:21:56 the process -> java surpassed the threshold for RAM Usage with 45.96
At the time 21-12-2020 14:26:38 the process -> java surpassed the threshold for RAM Usage with 44.4
At the time 21-12-2020 14:28:54 the process -> java surpassed the threshold for RAM Usage with 44.31
At the time 21-12-2020 14:37:47 the process -> java surpassed the threshold for RAM Usage with 43.94
At the time 21-12-2020 14:39:11 the process -> java surpassed the threshold for RAM Usage with 43.44
At the time 21-12-2020 14:52:59 the process -> java surpassed the threshold for RAM Usage with 44.31
At the time 21-12-2020 14:55:55 the process -> java surpassed the threshold for RAM Usage with 43.44
At the time 21-12-2020 14:57:59 the process -> java surpassed the threshold for RAM Usage with 44.66
At the time 25-01-2021 19:58:20 the process -> java surpassed the threshold for RAM Usage with 37.43
At the time 27-01-2021 19:13:36 the process -> java surpassed the threshold for RAM Usage with 50.59
At the time 27-01-2021 19:59:03 the process -> java surpassed the threshold for RAM Usage with 53.29
At the time 27-01-2021 20:01:27 the process -> java surpassed the threshold for CPU Usage with 64.3
```

Figura 2: Ficheiro logs dos Alarms

5 Conclusão

Arealização deste trabalho permitiu a aquisição de conhecimentos variados e extensos no que diz respeito à *API snmp4j* bem como a forma de monitorizar informação presente nas *MIBs*.

Relativamente aos objetivos propostos, trabalhei para que fossem cumpridos na íntegra, sendo que a ferramenta desenvolvida apresenta soluções e respostas para o que foi proposto pelos docentes com a realização deste projeto.

No entanto, penso que podia ter sido mais eficiente em algumas abordagens, sobretudo no que concerne o primeiro módulo, pois poderia apenas gerar um ficheiro log onde se incluía os processos em execução bem como o historial de toda a ferramenta, tendo optado pela solução apresentada neste relatório, pois inicialmente não ponderei que o programa incluísse essa funcionalidade.

6 Anexo

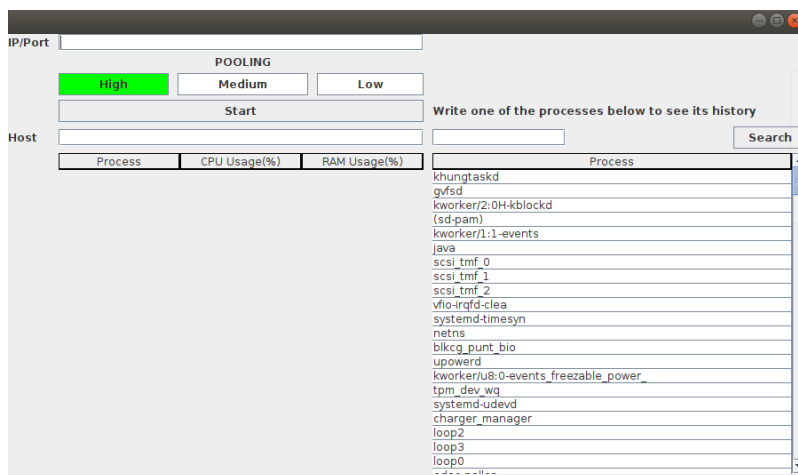


Figura 3: Interface inicial

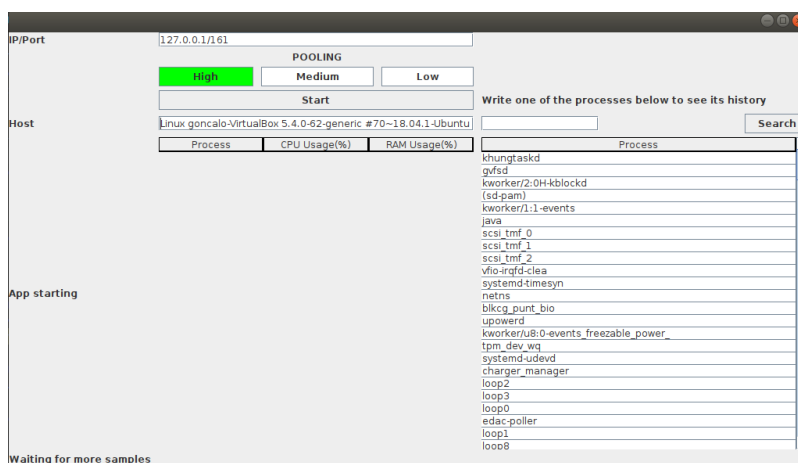


Figura 4: Iniciando aplicação depois de introduzido IP/Port

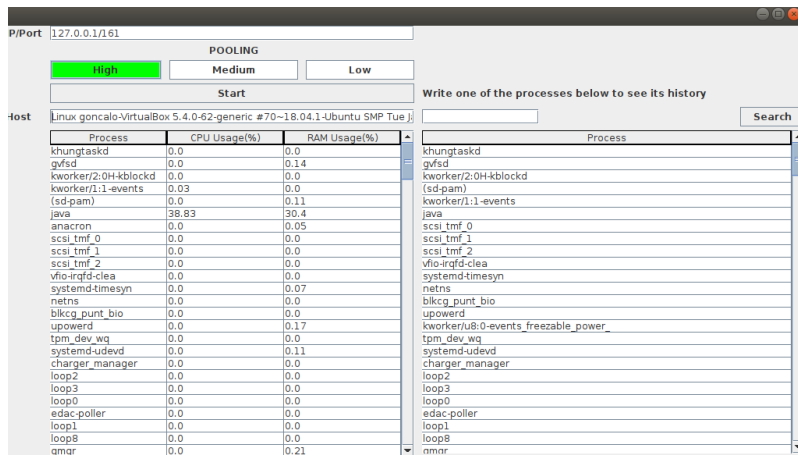


Figura 5: Informações dos processos

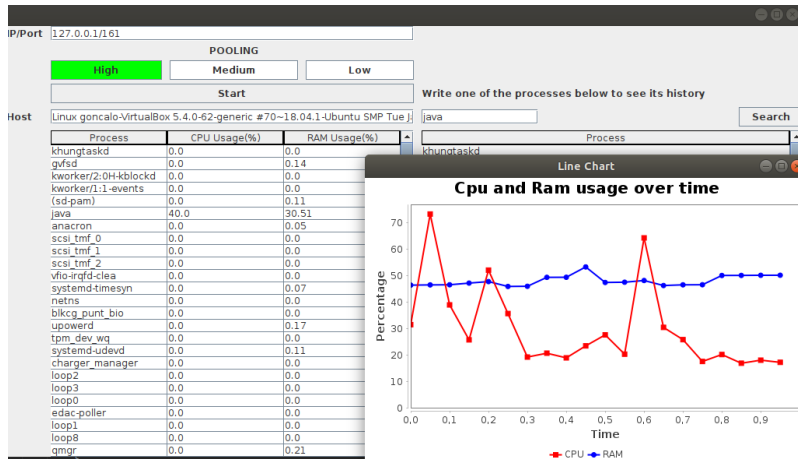


Figura 6: Gráfico com CPU e RAM "usage"de um processo ao longo do tempo

7 Bibliografia

<https://www.programcreek.com/java-api-examples/?api=org.snmp4j.util.TableUtils>

<https://stackoverflow.com/questions/3224687/getting-started-with-snmp4j>