Gestão de Redes
TP2 - Ferramenta de Monitorização

Mestrado Integrado
em
Engenharia Informática

Nome	Nº
Diogo Rocha	a79751

24 de fevereiro de 2021

1 Introdução

O seguinte relatório foi produzido na sequência do trabalho prático nº 2 da Unidade Curricular de **Gestão de Redes**. Foi pedido aos alunos pela equipa docente a criação de um programa para monitorização e análise de utilização dos recursos do sistema local pelos processos ativos num qualquer host. O desenvolvimento do projeto foi dividido em 3 partes , um módulo para implementação do mecanismo de monitorização , um programa de análise dos logs gerados com a consequente geração de um output em forma de texto de maneira a demonstrar os resultados obtidos dessa mesma análise.

2 Ferramentas utilizadas

Para desenvolvimento do projeto foi utilizada a linguagem de programação *java* , recorrendo à API *snmp4j* para obter os objectos pretendidos da MIB-2. Posteriormente foi utilizado um ficheiro json para armazenar os dados recolhidos.

3 Implementação do mecanismo de monitorização

3.1 Oids necessários

Em seguida serão listados todos os **oids** utilizados para obtenção de dados da MIB:

- ".1.3.6.1.2.1.25.4.2.1.2": nome dos processos
- ".1.3.6.1.2.1.25.5.1.1.1": cpu time de um processo
- "1.3.6.1.2.1.25.5.1.1.2": memória alocada para um processo
- ".1.3.6.1.2.1.25.1.6.0": número total de processos
- ".1.3.6.1.2.1.25.2.2.0": tamanho total da memória RAM

3.2 Obtenção dos dados da mib-2

Para a obtenção dos dados da mib-2 são utilizados os comandos *snmpget* para a obtenção de um único objeto , utilizado para obter o tamanho total da memória RAM e o número total de processos , e também é utilizado o comando *snmpgetbulk* para obtenção dos restantes objectos sendo este mais eficiente e permitindo usar uma menor largura de banda, após a obtenção dos dados estes são guardados em estruturas de dados permitindo depois a escrita no ficheiro de logs.

3.2.1 Classe Writes

Esta classe é a classe principal desta primeira parte. O seu método **execWrites(String host)** gere o intervalo de polling que foi definido para ser de 1 segundo, executa a classe **Manager**, responsável pela conexão à mib e extração dos objectos da mib. Após obter os dados guarda-os em estruturas de dados já organizados e escreve no ficheiro **JSON**. Esta função executa os métodos anteriores 10 vezes,foi assim definido caso exista alguma alteração na utilização do computador que cause alguma alteração significativa na utilização da RAM e do CPU esta possa ser detetada durante a execução.

3.2.2 Classe Manager

Esta classe é utilizada para realizar as ações acima referidas utilizando os métodos **getGet(OID oid)** para fazer um pedido à mib utilizando o comando *snmpget* e retorna uma String com o valor retornado do objecto dado como argumento e o método **start(OID[] oidsProcesses)** para fazer um pedido à mib utilizando o comando *snmpgetbulk* e retorna um List de variableBinding[] com os valores retirados dos objetos.

```
public class Manager {

    private Snmp snmp = null;
    private String address = null;

    public Manager(String add) { address = add; }

    public List<VariableBinding[]> start(OID[] OIDsProcesses) throws IOException {...}

    public String getGet(OID oid) throws IOException {...}


    private Target getTarget() {...}
```

Figura 1: Classe Manager

3.3 Geração de logs

Tendo os dados obtidos guardados em estruturas de dados procede-se à criação de um ficheiro **JSON** onde são a armazenados os dados.

3.3.1 Formato do JSON

O ficheiro **JSON** é constituído por um array de objectos. Cada objecto é relativo a um poll. Cada objecto é constituído por 5 objectos :

- TotalMemorySize - Indica a memória RAM em kbytes
- totalNumProcessesFromSnmp - Indica o número de processos a correr
- snmpData - Um array com 4 objetos com informação sobre cada processo :
  1. processAllocatedMem - Indica a memória alocada para o processo em kbytes
  2. processCPUTime - Indica o tempo de utilização do CPU do processo
  3. processName - Indica o nome do processo
  4. processId - Indica o id do processo
- host - Indica o IP do utilizador
- time - Indica a data de quando o poll foi feito

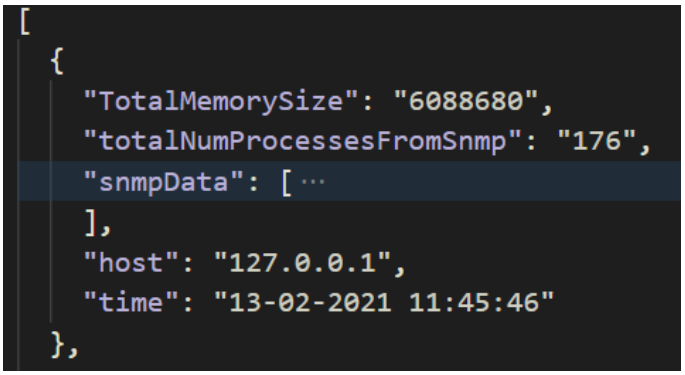

Figura 2: Formato do ficheiro JSON

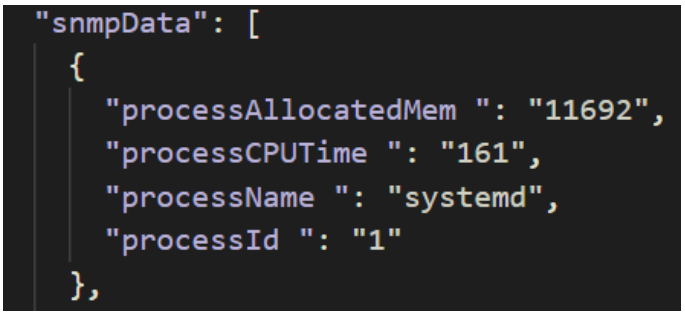

Figura 3: Formato do array **snmpData**

4 Análise dos Logs

Tendo os dados devidamente organizados no ficheiro **JSON**, prossegue-se à análise dos mesmos sobre a forma de queries , com o intuito de apresentar os resultados dessa análise ao utilizador. Foram definidas 3 queries :

- getPRamDia -Organiza os processos pelos possuem mais memória alocada e retorna os 10 que mais utilizam com a percentagem de memóriam RAM utilizada de um determinadado dia escolhido pelo utilizador.
- getPRamALLTime - Organiza os processos pelos possuem mais memória alocada e retorna os 10 que mais utilizam com a percentagem de memóriam RAM utilizada.
- getPIDsZero - Retorna todos os processos que apresentam o valor zero na utilização de RAM e de CPU.

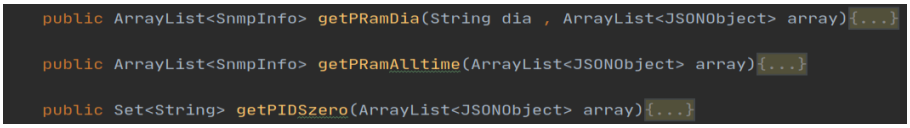

Figura 4: Querys

4.1 Mecanismo de alarmes

Para as querys desenvolvidas foi definido que se a utilização da RAM por parte do processo for igual ou superior a 20 por cento, ao apresentar a informação sobre o processo no terminal é adicionada uma mensagem de Warning.

5 Apresentação de resultados

Como referido acima após análise dos logs , os dados são apresentados como forma de texto no terminal ao utilizador, utilizando os valores retornados pelas querys definidas.

5.1 Menu inicial

```
Selecione uma das seguintes opções :
1 -> Ativar o mecanismo de monitorização snmp ;
2 -> Visualizar os 10 processos que maior percentagem de RAM utilizaram;
3 -> Visualizar os 10 processos que maior percentagem de RAM utilizaram num dia;
4 -> Visualizar os processos que não utilizam nem RAM nem CPU
```

Figura 5: Menu inicial

5.2 Ativar o mecanismo de monitorização

Ao selecionar esta opção o mecanismos de monitorização é ativado escrevendo novos dados no ficheiro de logs.

5.3 10 processos que mais utilizam RAM

```
Insira o IP do host no formato tcp/udp ,para o localhost/161 digite default
default
10 processos que maior percentagem de RAM utilizam :

Id: 2171, Nome: java, percentagem de uso: 24% !!! Warning de utilização de RAM!!!
Id: 1943, Nome: java, percentagem de uso: 22% !!! Warning de utilização de RAM!!!
Id: 6601, Nome: java, percentagem de uso: 20% !!! Warning de utilização de RAM!!!
Id: 1920, Nome: java, percentagem de uso: 20% !!! Warning de utilização de RAM!!!
Id: 1920, Nome: java, percentagem de uso: 20% !!! Warning de utilização de RAM!!!
Id: 2002, Nome: java, percentagem de uso: 20% !!! Warning de utilização de RAM!!!
Id: 3335, Nome: java, percentagem de uso: 19%
Id: 1943, Nome: java, percentagem de uso: 19%
Id: 1943, Nome: java, percentagem de uso: 19%
Id: 3335, Nome: java, percentagem de uso: 14%
```

Figura 6: Output

5.4 10 processos que mais utilizam RAM num dia

```
Insira o IP do host no formato tcp/udp ,para o localhost/161 digite default
default
Introduza o dia no formato dia-mes-ano
22-02-2021
10 processos que maior percentagem de RAM utilizaram no dia 22-02-2021:

Id: 2171, Nome: java, percentagem de uso: 24% !!! Warning de utilização de RAM!!!
Id: 6601, Nome: java, percentagem de uso: 20% !!! Warning de utilização de RAM!!!
Id: 1044, Nome: gnome-shell, percentagem de uso: 7%
Id: 6991, Nome: firefox, percentagem de uso: 6%
Id: 1037, Nome: gnome-shell, percentagem de uso: 5%
Id: 1299, Nome: snap-store, percentagem de uso: 5%
Id: 1332, Nome: snap-store, percentagem de uso: 5%
Id: 7116, Nome: Web Content, percentagem de uso: 3%
Id: 7039, Nome: Privileged Cont, percentagem de uso: 2%
Id: 820, Nome: Xorg, percentagem de uso: 2%
```

Figura 7: Output

5.5 Todos os processos que não utilizam nem RAM nem CPU

```
kworker/0:3-cgroup_destroy
pm_wq
kworker/u2:0-events_power_efficient
khungtaskd
kworker/u2:2-ext4-rsv-conversion
ecryptfs-kthrea
rcu_tasks_rude_
kthreadd
kworker/u3:0
kworker/u2:1-ext4-rsv-conversion
kworker/u2:2-events_freezable_power_
rcu_gp
md
ext4-rsv-conver
scsi_tmf_0
scsi_tmf_1
scsi_tmf_2
vfio-irqfd-clea
kworker/0:0
netns
blkcg_punt_bio
tpm_dev_wq
kworker/0:1-events
charger_manager
loop2
loop3
```

Figura 8: Alguns dos processos retornados

6 Conclusão e trabalho futuro

Para concluir penso que se atingiram a maior parte dos objetivos concluídos, e para isso gostava de agradecer à equipa docente nomeadamente ao Professor João por toda a disponibilidade. Como trabalho futuro, penso que permitir a análise para outro host diferente do localhost embora que o trabalho tenha sido desenvolvido nesse sentido encontrando-se praticamente preparado para essa função , realizar também uma análise mais específica em relação ao tempo de utilização do CPU e por fim apresentar os resultados em forma de gráfico pois neste momento e não sei bem o porquê mas o sistema operativo não permitiu apresentar os dados nessa forma.