

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Curso: Engenharia de Computação

Disciplina: Sistemas Distribuídos

Ano/Semestre: 2017/2

Professora: Anolan Yamilé Milanés Barrientos

Atividade: Trabalho Prático 1 - Implementação de um serviço de consultas a um log distribuído



Arley dos Santos Ribeiro

Bruno Marques Maciel

Cassiano de Brito Andrade

1. PROJETO DO PROGRAMA

Desenvolveu-se o código em C++ e optou-se pela abordagem *multithreading*, ou seja, ao chegar uma nova requisição no servidor, este cria uma nova *thread* para atendê-la e, após seu término, esta *thread* é finalizada. Observa-se, nesta abordagem, que há um bom aproveitamento dos recursos computacionais, por haver compartilhamento de recursos entre as *threads*, ao passo que os clientes são atendidos sob demanda, garantindo um certo grau de escalabilidade. Como as *threads* realizam apenas leitura, não há risco de resultados inesperados (leitura é idempotente) e nem há concorrência entre as mesmas.

Em comparação com a abordagem de processamento único, a abordagem *multithreading* tem a vantagem de poder atender múltiplos clientes simultaneamente. Entretanto, se não for definida uma quantidade máxima de *threads* que podem estar em execução na abordagem *multithreading*, pode-se sobrecarregar o servidor se este for exposto a uma elevada quantidade de requisições, a ponto de causar *crash* no mesmo. Isto não aconteceria na abordagem de processamento único, pois nesta atende-se uma requisição por vez, porém não tem-se nenhuma escalabilidade.

A abordagem multiprocessing, por sua vez, possui maior confiabilidade por não conter um ponto único de falha (PUF), porém consome muita memória, pois cada processo é independente e possui sua alocação de memória própria.

2. FORMATO DAS MENSAGENS

A mensagem enviada do cliente para o servidor possui, somente, o comando a ser executado e a quantidade de caracteres que este possui. A resposta do servidor ao cliente possui o resultado deste comando ao ser executado no próprio servidor, assim como o tamanho em caracteres do resultado.

Tabela 1 - Composição das mensagens trocadas entre as entidades

Mensagem enviada do cliente ao servidor	comando <i>grep</i>
Mensagem enviada do servidor ao cliente	resposta do comando <i>grep</i>

Criou-se um arquivo texto com o endereço IP e porta de todos os servidores separados por linha, de forma que, quando o cliente vai executar, ele abre este arquivo e armazena em uma variável local os endereços IPs e portas dos servidores.

Tabela 2 - Exemplo de arquivo que contém lista de todos os servidores

Endereço IP	Porta
192.168.1.114	8880
192.168.1.114	8881
192.168.1.114	8882
192.168.1.114	8883

3. UNIT TESTS

3.1 Teste de um cliente e um servidor em uma mesma máquina virtual

Inicialmente, testou-se a funcionalidade dos códigos em uma mesma máquina virtual, com o servidor sendo executado em um processo e o cliente em outro (terminais diferentes).

Este caso de teste visa garantir que:

1. O servidor irá aguardar por requisição, eventualmente atender uma requisição que chegar e retornar ao estado de escuta;
2. O cliente irá iniciar uma conexão com o servidor passando o comando desejado;
3. O servidor irá conseguir executar o comando e retornar a resposta ao cliente.

3.2 Teste de um cliente e múltiplos servidores em uma mesma máquina virtual

Executou-se em uma única máquina virtual quatro servidores e um cliente. Portanto, este caso de teste tem o intuito de testar a comunicação do cliente com múltiplos servidores, ou seja, a administração das requisições assim como a administração das respostas.

3.3 Teste de um cliente e múltiplos servidores em máquinas virtuais distintas

Para avaliar o sistema distribuído, planejou-se executar tanto o cliente quanto os servidores em máquinas virtuais distintas, de forma que o endereço IP de cada uma delas fosse diferente. Assim, procura-se testar a comunicação entre os processos pela rede local.

3.4 Teste de um cliente e múltiplos servidores em máquinas virtuais distintas consultando arquivo log de 100MB

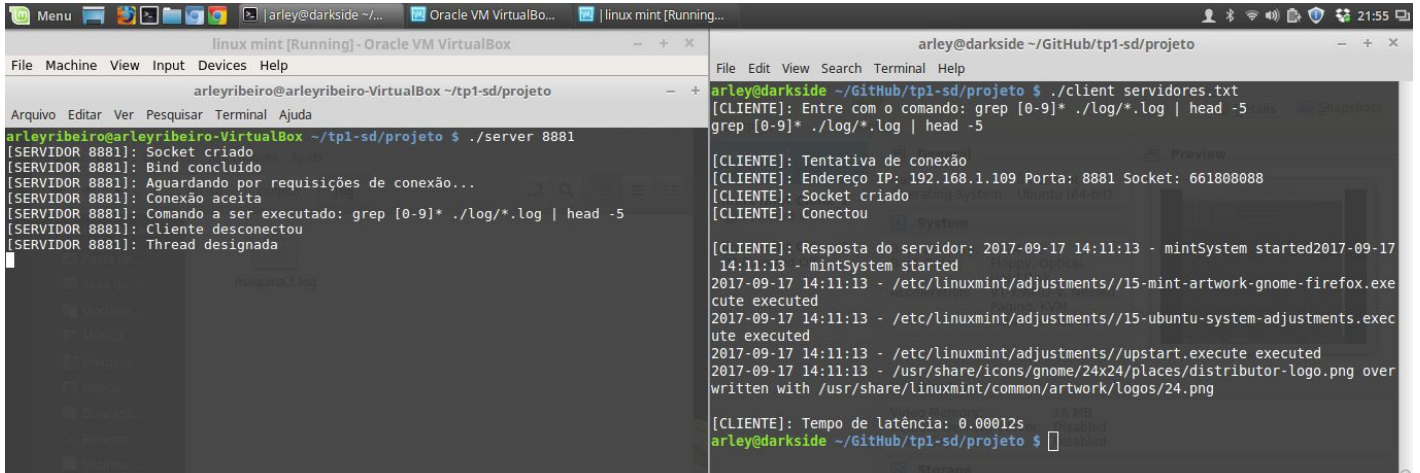
Neste caso de teste, pretende-se avaliar o tempo de latência da comunicação, desde o momento de requisição do cliente até a obtenção da resposta, em função de arquivo de log de 100MB por servidor.

4. RESULTADOS ENCONTRADOS

4.1 Teste de um cliente e um servidor em uma mesma máquina virtual

Pela Figura 1, observa-se que o cliente requisitou o comando `grep [0-9]* ./log/*.log | head -5`, ou seja, buscou-se por *matches* com o “[0-9]*” nos arquivos de extensão *log*, solicitando os cinco primeiros resultados. A resposta obtida pode ser observada na Figura 1. Observa-se um tempo de latência de 120µs.

Figura 1 - Resultado obtido para o caso de teste especificado na Seção 3.1



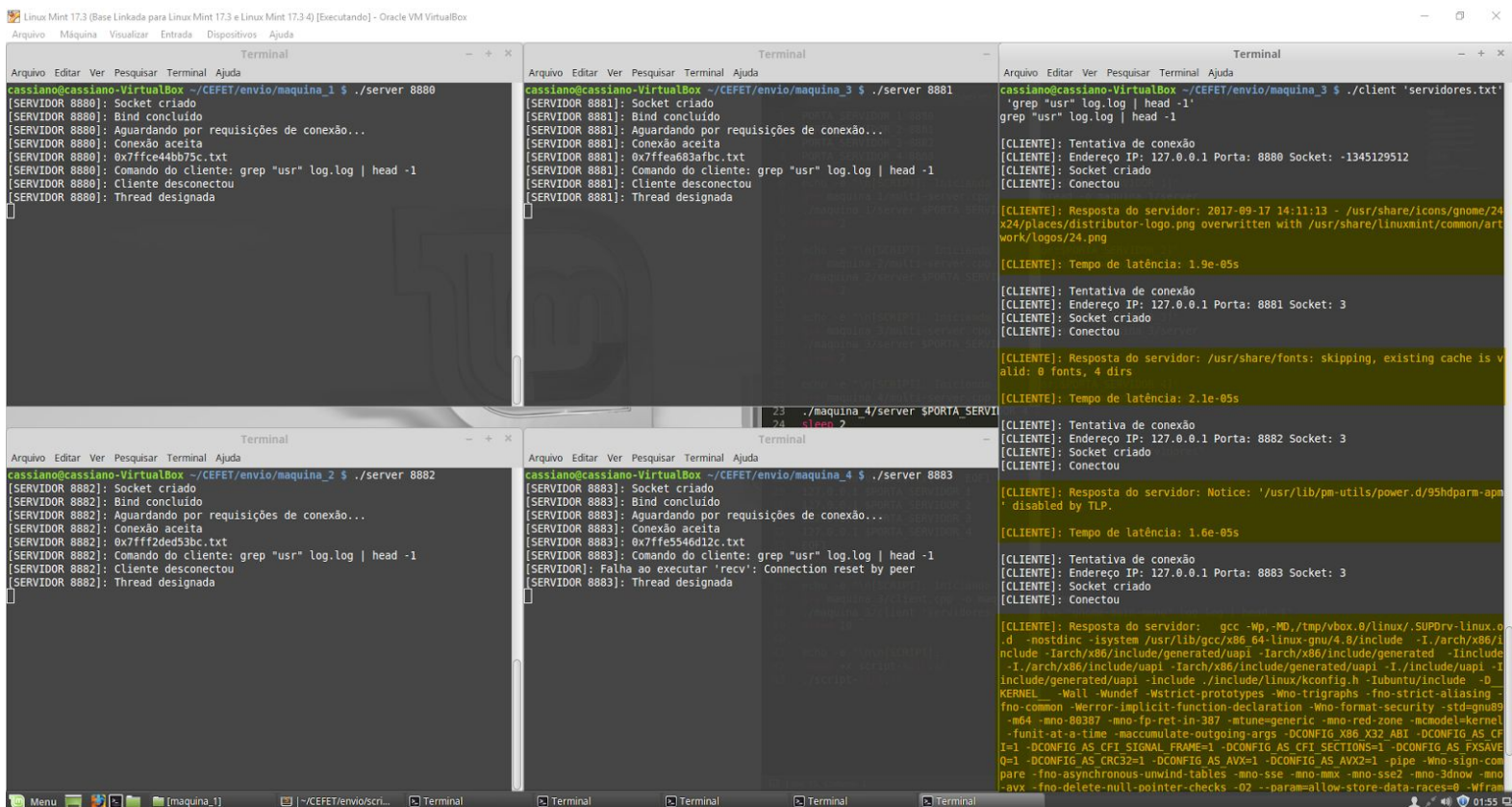
```
arley@darkside ~/tp1-sd/projeto
[SERVIDOR 8881]: Socket criado
[SERVIDOR 8881]: Bind concluido
[SERVIDOR 8881]: Aguardando por requisições de conexão...
[SERVIDOR 8881]: Conexão aceita
[SERVIDOR 8881]: Comando a ser executado: grep [0-9]* ./log/*.log | head -5
[SERVIDOR 8881]: Cliente desconectou
[SERVIDOR 8881]: Thread designada

arley@darkside ~/tp1-sd/projeto
[CLIENTE]: Tentativa de conexão
[CLIENTE]: Endereço IP: 192.168.1.109 Porta: 8881 Socket: 661808088
[CLIENTE]: Socket criado
[CLIENTE]: Conectou
[CLIENTE]: Resposta do servidor: 2017-09-17 14:11:13 - mintSystem started
2017-09-17 14:11:13 - /etc/linuxmint/adjustments//15-mint-artwork-gnome-firefox.exe
2017-09-17 14:11:13 - /etc/linuxmint/adjustments//15-ubuntu-system-adjustments.exe
2017-09-17 14:11:13 - /etc/linuxmint/adjustments//upstart.execute
2017-09-17 14:11:13 - /usr/share/icons/gnome/24x24/places/distributor-logo.png over
written with /usr/share/linuxmint/common/artwork/logos/24.png
[CLIENTE]: Tempo de latência: 0.00012s
arley@darkside ~/tp1-sd/projeto
```

4.2 Teste de um cliente e múltiplos servidores em uma mesma máquina virtual

Rodou-se quatro servidores, um em cada terminal, e o cliente em um quinto terminal. Solicitou-se pelo comando `grep "usr" log.log | head -1`, e obteve-se as quatro respostas, como destacado em amarelo.

Figura 2 - Resultado obtido para o caso de teste especificado na Seção 3.2



```
cassiano@cassiano-VirtualBox ~/CEFET/envio/maquina_1 $ ./server 8880
[SERVIDOR 8880]: Socket criado
[SERVIDOR 8880]: Bind concluido
[SERVIDOR 8880]: Aguardando por requisições de conexão...
[SERVIDOR 8880]: Conexão aceita
[SERVIDOR 8880]: 0x7ffce44bb75c.txt
[SERVIDOR 8880]: Comando do cliente: grep "usr" log.log | head -1
[SERVIDOR 8880]: Cliente desconectou
[SERVIDOR 8880]: Thread designada

cassiano@cassiano-VirtualBox ~/CEFET/envio/maquina_3 $ ./client 'servidores.txt'
'grep "usr" log.log | head -1'
grep "usr" log.log | head -1
[CLIENTE]: Tentativa de conexão
[CLIENTE]: Endereço IP: 127.0.0.1 Porta: 8880 Socket: -1345129512
[CLIENTE]: Socket criado
[CLIENTE]: Conectou
[CLIENTE]: Resposta do servidor: 2017-09-17 14:11:13 - /usr/share/icons/gnome/24
x24/places/distributor-logo.png overwritten with /usr/share/linuxmint/common/art
work/logos/24.png
[CLIENTE]: Tempo de latência: 1.9e-05s

cassiano@cassiano-VirtualBox ~/CEFET/envio/maquina_2 $ ./server 8882
[SERVIDOR 8882]: Socket criado
[SERVIDOR 8882]: Bind concluido
[SERVIDOR 8882]: Aguardando por requisições de conexão...
[SERVIDOR 8882]: Conexão aceita
[SERVIDOR 8882]: 0x7fff2ded53bc.txt
[SERVIDOR 8882]: Comando do cliente: grep "usr" log.log | head -1
[SERVIDOR 8882]: Cliente desconectou
[SERVIDOR 8882]: Thread designada

cassiano@cassiano-VirtualBox ~/CEFET/envio/maquina_4 $ ./server 8883
[SERVIDOR 8883]: Socket criado
[SERVIDOR 8883]: Bind concluido
[SERVIDOR 8883]: Aguardando por requisições de conexão...
[SERVIDOR 8883]: Conexão aceita
[SERVIDOR 8883]: 0x7ffe5546d12c.txt
[SERVIDOR 8883]: Comando do cliente: grep "usr" log.log | head -1
[SERVIDOR 8883]: Falha ao executar 'recv': Connection reset by peer
[SERVIDOR 8883]: Thread designada

cassiano@cassiano-VirtualBox ~/CEFET/envio/maquina_3 $ ./client 'servidores.txt'
'grep "usr" log.log | head -1'
grep "usr" log.log | head -1
[CLIENTE]: Tentativa de conexão
[CLIENTE]: Endereço IP: 127.0.0.1 Porta: 8881 Socket: 3
[CLIENTE]: Socket criado
[CLIENTE]: Conectou
[CLIENTE]: Resposta do servidor: Notice: '/usr/lib/pm-utils/power.d/95hdpam-apa
' disabled by TLP.
[CLIENTE]: Tempo de latência: 1.6e-05s

cassiano@cassiano-VirtualBox ~/CEFET/envio/maquina_3 $ ./client 'servidores.txt'
'grep "usr" log.log | head -1'
grep "usr" log.log | head -1
[CLIENTE]: Tentativa de conexão
[CLIENTE]: Endereço IP: 127.0.0.1 Porta: 8882 Socket: 3
[CLIENTE]: Socket criado
[CLIENTE]: Conectou
[CLIENTE]: Resposta do servidor: gcc -Wp,-MD,/tmp/vbox.0/linux/.SUPDrv-linux.o
.d -nostdinc -isystem /usr/lib/gcc/x86_64-linux-gnu/4.8/include -I./arch/x86/i
nclude -Iarch/x86/include/generated/uapi -Iarch/x86/include/generated -Iinclude
-I./arch/x86/include/uapi -Iarch/x86/include/generated/uapi -I./include/uapi -I
include/generated/uapi -include ./include/linux/kconfig.h -Iubuntu/include -D_
KERNEL_ -Wall -Wundef -Wstrict-prototypes -Wno-trigraphs -fno-strict-aliasing
-fno-common -Werror-implicit-function-declaration -Wno-format-security -std=gnu8
-m64 -mno-80387 -mno-fp-ret-in-387 -mtune=generic -mno-red-zone -mcmodel=kernel
-funit-at-a-time -maccumulate-outgoing-args -DCONFIG_X86_X32_ABI -DCONFIG_AS_CFI
-I1 -DCONFIG_AS_CFI_SIGNAL_FRAME=1 -DCONFIG_AS_CFI_SECTIONS=1 -DCONFIG_AS_FXSAVE
Q=1 -DCONFIG_AS_CRC32=1 -DCONFIG_AS_AVX=1 -DCONFIG_AS_AVX2=1 -pipe -Wno-sign-com
pare -fno-asynchronous-unwind-tables -mno-sse -mno-mmx -mno-sse2 -mno-3dnow -mno
-avx -fno-delete-null-pointer-checks -O2 --param=allow-store-data-races=0 -Wfram
```

4.3 Teste de um cliente e múltiplos servidores em máquinas virtuais distintas

Executou-se os servidores em três máquinas virtuais distintas, ao passo que o cliente foi executado na máquina real. Com isso, observa-se na Figura 3 que o cliente (1º terminal) tem impresso a resposta do primeiro servidor (3º terminal) e do segundo servidor (4º terminal), ao passo que na Figura 4 já é possível ver a resposta do terceiro servidor (2º terminal).

Figura 3 - Primeira parte do resultado obtido para o caso de teste especificado na Seção 3.3

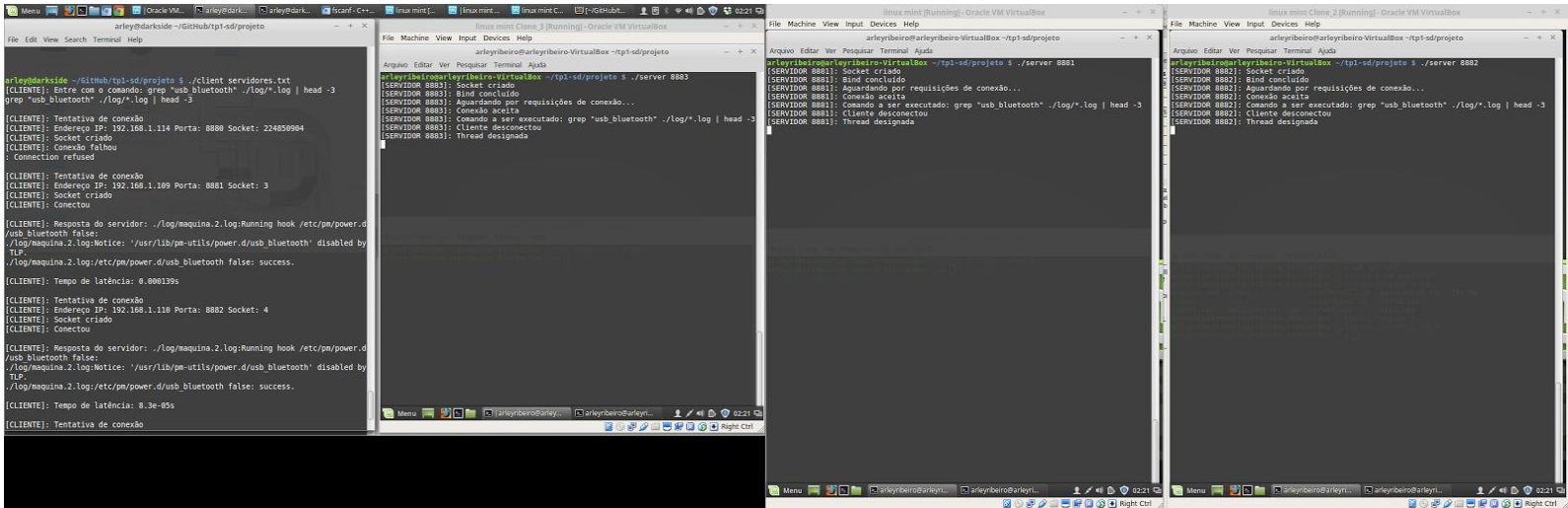
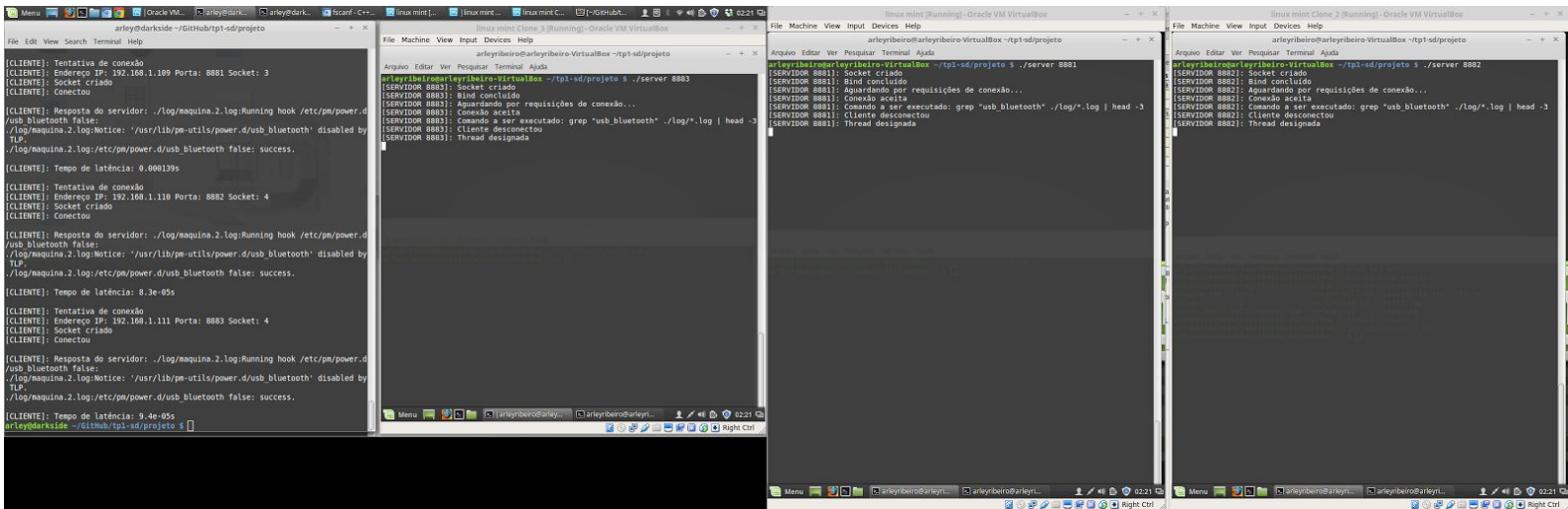


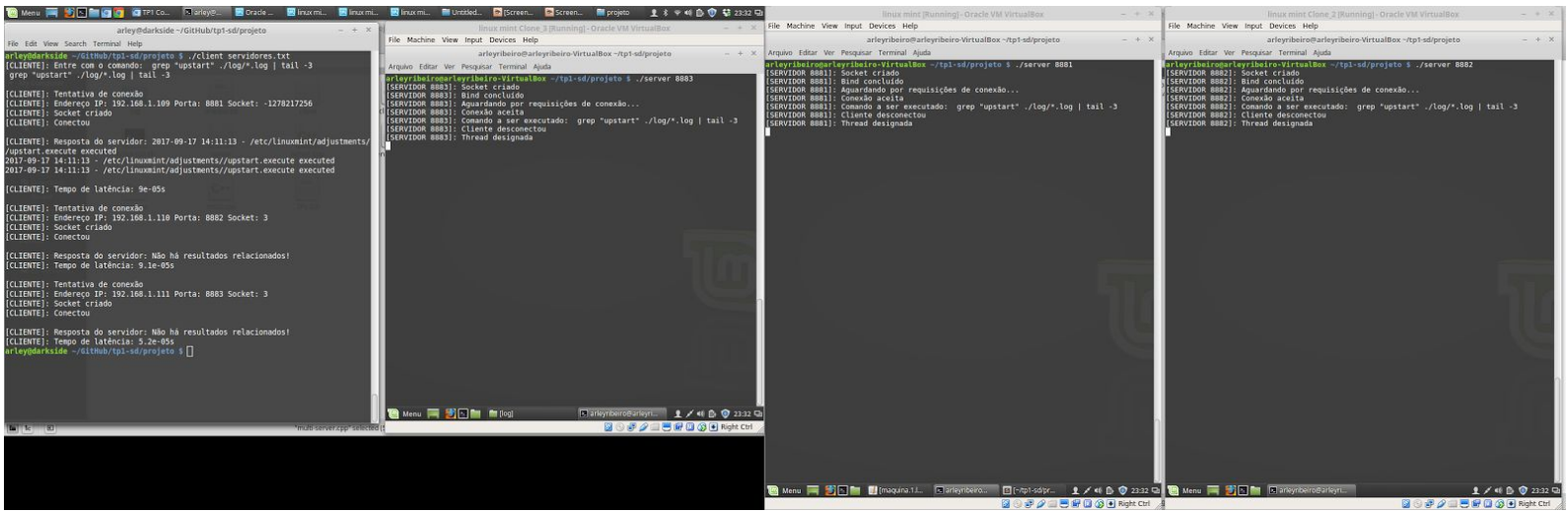
Figura 4 - Segunda parte do resultado obtido para o caso de teste especificado na Seção 3.3



4.4 Teste de um cliente e múltiplos servidores em máquinas virtuais distintas consultando arquivo log de 100MB

Executou-se os três servidores e o cliente em máquinas virtuais distintas, de forma a analisar o tempo de latência quando cada servidor consulta arquivos de 100MB cada. O resultado obtido pode ser visualizado na Figura 5, de forma que o primeiro terminal é referente ao cliente e os demais ao servidor.

Figura 5 - Resultado obtido para o caso de teste especificado na Seção 3.4



5. AVALIAÇÃO DE DESEMPENHO E CONCLUSÕES

Para o sistema implementado, observa-se que a complexidade associada ao tempo de requisição do cliente é $O(s)$, sendo s a quantidade de servidores, devido à iteração linear sobre a lista dos mesmos. Este método utilizado não é o mais eficiente, pois para um s muito grande o tempo de requisição será muito alto. No entanto, ele foi adotado devido à facilidade de implementação e depuração.

O tempo de atendimento do servidor possui complexidade $O(f)$, sendo f o tamanho total dos arquivos que ele terá de analisar. Observa-se que o número de clientes não entra na função de complexidade pois o servidor aplica a abordagem *multithreading*, ou seja, dado um número de clientes ideal, nenhum deles deverá enfrentar fila.

Na Tabela 3, observou-se entre os casos de teste 3.1 a 3.4 que o tempo de latência aumentou consideravelmente no caso 3.4, pois este abordou a situação em que os arquivos de log possuíam em torno de 100MB cada.

Tabela 3 - Exibição do tempo de latência médio para cada caso de teste

Caso de teste	Tempo de latência médio (10^{-3} segundos)
3.1	0,120
3.2	0,019
3.3	0,105
3.4	0,233

Como o sistema foi executado em uma rede local, observou-se que o tempo de latência possui seu gargalo no processamento dos servidores, pois o tempo de transmissão é baixíssimo. Com isso, destaca-se que uma possível otimização para o sistema implementado é tornar o cliente *multithreading*, pois desta forma este poderia enviar requisições paralelas aos servidores, deixando cada *thread* administrar a sua respectiva conexão para, no final, unir todas as respostas oriundas de cada *thread* em uma só.

REFERÊNCIAS

ADVANTAGES and Disadvantages of a Multithreaded/Multicontexted Application. Disponível em: <https://docs.oracle.com/cd/E13203_01/tuxedo/tux71/html/pgthr5.htm>. Acesso em: 17 set. 2017.

C++ multi threaded server and client. 2013. Disponível em: <<http://www.cplusplus.com/forum/unices/116977/>>. Acesso em: 17 set. 2017.

SERVER and client example with C sockets on Linux. 2012. Disponível em: <<http://www.binarytides.com/server-client-example-c-sockets-linux/>>. Acesso em: 17 set. 2017.

SOCKET Programming in C/C++. Disponível em: <<http://www.geeksforgeeks.org/socket-programming-cc/>>. Acesso em: 17 set. 2017.

SOCKETS Tutorial. Disponível em: <http://www.linuxhowtos.org/C_C++/socket.htm>. Acesso em: 17 set. 2017.