Universidade do Minho

Mestrado em Engenharia Informática

Ciência de Dados

Computação Avançada

João Pimentel A80874
José Carvalho A80424
Ricardo Martins A78914

21 de Janeiro de 2020

Resumo

O presente projeto consistiu na utilização de uma arquitetura *High Throughput Computing* para permitir a distribuição de trabalho e consequente redução de tempo de execução na conversão de um vídeo *Full HD* para *HD*. Para tal, criaram-se *scripts* para realizar as várias fases da tarefa, sendo estes referentes a divisão, *downscaling* e concatenação de vários segmentos de vídeo. Numa fase de análise de resultados, observou-se que o modelo implementado foi capaz de reduzir o tempo total de execução, aumentando o número de conversões realizadas por unidade de tempo. Apesar de alcançar os requisitos do enunciado, o projeto poderia ser melhorado tanto na divisão dos fragmentos, como no método para troca de ficheiros entre os recursos e o agente.

Conteúdo

1 Introdução **1**

2 Metodologia **2**
2.1 Configurações 2
2.1.1 Máquinas Virtuais 2
2.1.2 HTCondor 2
2.2 Funcionamento do Programa 2
2.2.1 Fragmentação do Vídeo 3
2.2.2 *Downscaling* dos Fragmentos 3
2.2.2.1 *Script* de Compressão 4
2.2.3 Concatenação dos Fragmentos 4

3 Resultados e Discussão **5**
3.1 Capacidades do Programa 5
3.2 Ganho Temporal 5

4 Conclusões e Trabalho Futuro **6**

Lista de Tabelas

1 Tabela 1 - Comparação dos tempos de execução da tarefa 5

Lista de Extratos

1 Extrato 1 - Conteúdo do ficheiro *sub.dag*. 3
2 Extrato 2 - Conteúdo do ficheiro *split.sh*. 3
3 Extrato 3 - Conteúdo do ficheiro *compress.sub* 3
4 Extrato 4 - Conteúdo do ficheiro *compress.sh* 4
5 Extrato 5 - Conteúdo do ficheiro *merge.sh*. 4

1 Introdução

O presente relatório é o resultado da resolução do trabalho prático da unidade curricular de Computação Avançada. O foco deste trabalho passou por implementar um sistema de *High Throughput Computing* que permita o *downscale* de um ficheiro de vídeo com resolução *Full HD* para *HD*. Para tal, teve-se por base a utilização das funcionalidades fornecidas pela ferramenta *HTCondor*, bem como métodos e arquiteturas deste tipo de programas mencionados nas aulas da unidade curricular.

A linha de pensamento base da resolução do projeto passou por dividir o vídeo em vários fragmentos, sendo que a operação é efetuada sobre todos estes, tirando proveito da rede existente e dos recursos computacionais disponíveis. No final, os vários segmentos do vídeo devem ser concatenados, de forma a obter o resultado pretendido.

Dito isto, o grande objetivo deste projeto foi o aumento da experiência dos alunos no âmbito de sistemas de *High Throughput Computing*, ou seja, sistemas capazes de satisfazer um elevado número de tarefas por unidade de tempo. Quer isto dizer que se pretendia efetuar o desenvolvimento de uma ferramenta capaz de, tendo em conta a arquitetura base, dividir, operar e concatenar todos os fragmentos de um vídeo, tirando proveito dos recursos disponíveis.

Nos capítulos seguintes serão demonstrados os problemas, formas de resolução e testes efetuados de modo a obter os resultados ideais para o problema proposto.

2 Metodologia

2.1 Configurações

O primeiro passo para a resolução do projeto passa por configurar as máquinas na rede, de modo a que toda a infraestrutura funcione como pretendido. Esta é composta por três máquinas virtuais, ligadas entre si por uma rede local. Tendo em conta os seus papeis no que toca à arquitetura do *HTCondor*, todas podem executar trabalhos, mas apenas uma é responsável pela administração do *cluster*.

Em seguida serão apresentados os passos para a configuração dos vários componentes associados ao projeto.

2.1.1 Máquinas Virtuais

Tendo em conta a maior facilidade em implementar este tipo de sistema que interliga vários nodos através do uso de máquinas virtuais, foi decido que esta seria a abordagem a seguir.

Assim, de modo a testar o *cluster*, foram utilizadas três máquinas virtuais, sendo que todas tiveram como base o sistema operativo *CentOS* 7. Foi necessário desativar as *firewalls* em todas as máquinas, bem como instalar a ferramenta de *HTC*. Foi ainda necessária a instalação de um comando capaz de processar ficheiros de vídeo, neste caso o comando *ffmpeg*.

Por último, de modo a permitir a execução de trabalhos, bem como a comunicação das máquinas entre si, tornou-se indispensável configurar uma rede local entre os nodos. Note-se que os endereços *IP* foram definidos manualmente para cada uma das máquinas, tendo por base o ficheiro de configuração fornecido pelo docente da unidade curricular.

2.1.2 HTCondor

Finitas as configurações das máquinas virtuais, foram configurados os papeis no *HTCondor*. Para tal, cada nodo foi configurado com um ficheiro fornecido pelo docente, sendo que uma das máquinas foi configurada como *master* e as restantes como *workers*.

Como mencionado anteriormente, configurar os endereços *IP* é crucial para a gestão das máquinas no *cluster*. Deste modo, a máquina *master* tem a si associada a restrição de que as máquinas no *cluster* devem ter endereço *IP* do tipo *10.0.0.**. Já no caso das máquinas *worker*, foi definido que estas devem anunciar ao nodo *10.0.0.20* a sua disponibilidade para realizar trabalhos.

Note-se, ainda, que tendo em conta as configurações fornecidas, todas as máquinas podem submeter e executar tarefas, mas apenas o *master* pode gerir o *cluster*. Isto deve-se ao seu papel de *Central Manager*.

2.2 Funcionamento do Programa

Depois de configurado o *cluster*, o próximo passo foi o desenvolvimento da solução, tendo sido esta baseada na criação de uma tarefa que permita realizar o *downscaling* do vídeo dado como *input*.

Sendo assim, o modo de funcionamento da tarefa baseia-se no sistema *DAGman* da ferramenta *HTC*, que permite definir, através de um grafo acíclico direcionado, o fluxo do trabalho, de modo a otimizar a divisão da carga computacional pelos recursos disponíveis. Ora, dito isto, a solução dividiu-se em três grandes fases:

1. Divisão do vídeo em fragmentos;
2. *Downsclaing* de cada fragmento;
3. Concatenação dos fragmentos.

O ficheiro de submissão principal, visível no Extrato 1, teve como função principal indicar ao *HTCondor* o que devia fazer antes e depois de chamar o *job* de *downscaling* dos fragmentos. Neste extrato é visível que o *script* responsável por fragmentar as parcelas do vídeo deve ser chamado

antes da compressão (indicador *PRE*) e o *script* referente à junção dos mesmos só é efetuado no final (indicador *POST*).

```
JOB A compress.sub
SCRIPT PRE A split.sh
SCRIPT POST A merge.sh
```

Extrato 1 - Conteúdo do ficheiro *sub.dag*.

Nas sub-secções seguintes serão apresentadas, de forma detalhada, todas as etapas mencionadas, bem como explicações da sua implementação.

2.2.1 Fragmentação do Vídeo

Para inicializar o processo pretendido, foi crucial dividir o vídeo em vários segmentos, antes de executar qualquer outra operação. Esta tarefa foi efetuada com auxílio do comando *ffmpeg*, como se pode observar pelo Extrato 2, sendo a máquina que submeteu a tarefa quem o executa. A ideia passou por dividir o ficheiro de entrada em vários ficheiros de duração de 1 segundo, sendo que estes ficariam com o nome *input$i.mp4*, sendo este $i o índice do segmento no vídeo original. Note-se que esta não é a solução ideal, uma vez que para um vídeo com elevada duração, seria gerada uma elevada quantidade de fragmentos.

```
#!/bin/sh
ffmpeg -i fragmento.mp4 -c copy -map 0 -segment_time 00:00:01 -f segment input%d.mp4
```

Extrato 2 - Conteúdo do ficheiro *split.sh*.

2.2.2 *Downscaling* dos Fragmentos

Estando o vídeo original dividido em vários segmentos, a tarefa central pode ser efetuada, sendo esta especificada por um conjunto de componentes associados à ferramenta *HTCondor*, como se pode ver pelo Extrato 3. Neste é possível constatar que são submetidos 10 processos do *script* de *downscaling* (*compress.sh*), sendo 10 o número de fragmentos gerados anteriormente. Desta forma, cada um dos processos irá receber como argumento um determinado fragmento, vindo da máquina de origem da tarefa.

```
Executable              = compress.sh
Universe                = vanilla
log                     = log/frag_$(cluster).log
output                  = log/stdout_$(cluster).txt
error                   = log/stder_$(cluster).txt
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
transfer_input_files    = input$(process).mp4
transfer_output_files   = output$(process).mp4

Arguments       = $(process)

Queue 10
```

Extrato 3 - Conteúdo do ficheiro *compress.sub* .

Como forma de definir qual o processo que deve efetuar o *downscaling* de cada fragmento, é utilizado o número do processo (*$(process)*). Deste modo, é garantido que cada processo apenas trabalha com o fragmento correspondente. Além disso, cada um destes processos passa, também, o seu número como argumento ao *script* de modo a que o nome do *output* tenha em conta a ordem inicial do vídeo.

É, ainda, de destacar que, de modo a facilitar a compreensão da execução e possíveis problemas, os *logs* foram direcionados para uma pasta específica, onde cada um destes é identificado pelo número do *cluster* onde foi executado (*$(cluster)*).

Depois de obtidos todos os fragmentos já processados, estes são transferidos para a máquina onde o trabalho foi submetido, para que possam ser concatenados.

2.2.2.1 *Script* de Compressão

Além da compreensão do *job*, é importante compreender como é efetuado o *downscale* do vídeo. Assim, através da análise do Extrato 4, é possível observar que a ideia passa por redimensionar cada frame de entrada a 720 píxeis. Note-se que o ficheiro de saída terá o índice da sua posição no vídeo original no seu nome.

```
#!/bin/sh
ffmpeg -i input$1.mp4 -vf scale=-1:720 output$1.mp4
```

Extrato 4 - Conteúdo do ficheiro *compress.sh* .

2.2.3 Concatenação dos Fragmentos

Executadas todas as instâncias de *downscaling* e recebidos todos os fragmentos *HD* correspondentes, está tudo pronto à execução da fase final de transformação, bem como a remoção de ficheiros desnecessários.

Tal como no caso da divisão do vídeo original em vários segmentos, esta etapa é efetuada pela máquina que submeteu a tarefa. Como se pode ver pelo Extrato 5, por cada ficheiro no formato *output$i.mp4* encontrado na diretoria de execução, é criada uma entrada numa lista. Esta lista é, então, utilizada para efetuar a concatenação de todos os fragmentos, sendo argumento do comando *ffmpeg*. Note-se que a lista tem em conta a ordem dos segmentos devido ao formato do seu nome e consequente inclusão do índice no mesmo. Dito isto, o resultado final pode ser encontrado num ficheiro de vídeo com nome *fragmento_720.mp4*. Por fim, os fragmentos e a lista são eliminados.

```
#!/bin/sh
printf "file '%s'\n" ./output*.mp4 > list.txt
ffmpeg -f concat -safe 0 -i list.txt -c copy fragmento_720.mp4

rm input*.mp4
rm output*.mp4
rm list.txt
```

Extrato 5 - Conteúdo do ficheiro *merge.sh*.

3 Resultados e Discussão

3.1 Capacidades do Programa

O programa desenvolvido permite efetuar a conversão de um vídeo *Full HD* para *HD*, dividindo a carga de trabalho pelas várias máquinas da rede, tal como pretendido. Assim, foi possível reduzir o tempo necessário à conversão do vídeo, bem como a carga de trabalho associada a cada máquina, quando a quantidade de dados em tratamento assim necessita. Tenha-se a título de exemplo o vídeo fornecido. Neste, o tempo de execução na máquina local é inferior ao tempo de execução no *cluster*, pelo que a plataforma *HTC* não é a escolha ideal. Já no caso de um vídeo com duração maior, no caso de teste de 60 segundos, a plataforma provou-se bastante propensa para a conversão.

Desta forma, o conceito de *High Throughput Computing* foi conseguido, tendo sido aumentado o *throughput* de uma tarefa, tirando proveito das funcionalidades fornecidas pelo sistema.

Note-se que a tarefa desenvolvida funciona da forma desejada para o exemplo pretendido. No entanto, caso fosse utilizado um vídeo com duração diferente, o número de segmentos gerados seria diferente. Idealmente, a tarefa deveria receber a quantidade de fragmentos a gerar, em vez de a divisão ser efetuada com base na duração de cada segmento, sendo este valor 1 segundo, de momento.

3.2 Ganho Temporal

Através da conversão de um vídeo com duração de 60 segundos na máquina local em comparação com a tarefa *HTCondor* desenvolvida (valores visíveis na Tabela 1) é possível constatar que existe um ganho temporal, possibilitado pela divisão das cargas computacionais. Assim, através da divisão do trabalho pelos nodos da rede, bem como a segmentação num número não muito elevado de fragmentos, visto que estes têm a si associados um *overhead* de transferência entre máquinas, o resultado é um sistema com maior número de tarefas realizadas por unidade de tempo. Dito isto, podem-se observar as vantagens da utilização deste tipo de ferramentas aquando do tratamento de grandes quantidades de dados e/ou tarefas complexas que podem ser divididas em vários segmentos, não utilizando tantos recursos computacionais de uma só máquina, bem como reduzindo o tempo total de execução. Note-se que em certos casos a utilização do *cluster* não compensa, devido à transferência de dados associada aos segmentos, atrasando o tempo total de execução.

Tabela 1 - Comparação dos tempos de execução da tarefa

Tipo de execução	**Tempo (Minutos)**
Máquina Local	4:08
HTC - 60 Fragmentos	2:06
HTC - 10 Fragmentos	1:58
HTC - 6 Fragmentos	1:57
HTC - 3 Fragmentos	1:49

4 Conclusões e Trabalho Futuro

Ao longo do presente projeto encontra-se representado o trabalho prático da unidade curricular de Computação Avançada. Neste contexto foram abordados métodos de modo a dividir a carga de trabalho entre várias máquinas num *cluster*, tirando proveito dos recursos computacionais disponíveis, bem como da rede. Foi, desta forma, necessário tratar uma grande quantidade de dados de vídeo, efetuado a conversão de *Full HD* para *HD*.

Tendo em conta que este processo é bastante moroso quando efetuado numa só máquina, o método passou por dividir o vídeo em vários segmentos, sendo estes divididos pelas máquinas da rede. Nestas, é efetuada a conversão, permitindo uma redução na carga computacional quando em comparação com a situação inicial. Por fim, todos os segmentos devem ser concatenados, permitindo reconstruir a ordem do vídeo.

Apesar da tarefa parecer complexa, a utilização de uma ferramenta *HTC* permitiu facilitar bastante o processo. Esta permitiu a troca de ficheiros entre as entidades *shadow* e *sand-box* da arquitetura *Condor*. Permitiu, também, a associação dos vários segmentos aos vários processos gerados.

Apesar disso, a divisão do vídeo num determinado número de segmentos poderia ter sido melhorada, uma vez que esta é efetuada com base no exemplo fornecido. Quer isto dizer que são gerados 10 segmentos de vídeo, cada um com 1 segundo de duração. No caso de se alterar a duração do vídeo a converter, seria necessário alterar o valor de segmentos gerados, de modo a permitir o funcionamento correto da aplicação. Outro aspeto que poderia ser melhorado passa pela utilização de um sistema de ficheiros distribuídos com *NFS*, em que o *NFS Server* teria todos os ficheiros necessários à execução da tarefa, sendo que as máquinas a correr o *job* apenas teriam que aceder ao mesmo.

Em suma, a realização deste trabalho exigiu a aplicação de todos os conhecimentos lecionados em contexto de aula no que toca ao módulo de *High Throughput Computing*, permitindo que o grupo cumprisse todos os objetivos propostos no enuciado, conciliando assim a teoria e a prática.