Video processing cluster

1 Introduzione

1.1 Informazioni sul progetto

Questo progetto è supervisionato dal docente Geo Petrini e sviluppato dagli allievi: Samuele Abbà, Damian Campesi e Gioele Cavallo.
Il progetto è svolto nella scuola SAMT di Canobbio nelle ore adibite per i progetti dal 27.01.2022 al 05.05.2022.

1.2 Abstract

Situazione iniziale

All'inizio di questo progetto ci è stato chiesto di realizzare un sistema che permettesse di caricare dei filmati mp4 (di massimo 500MB) in un sito. Il sito permette di ottenere delle statistiche, con relativo grafico, sulla composizione del filmato (quantità di frame I, B, P) e poterne scaricare la versione: con motion vector o con solo frame I / B / P o tutti i frame in una zip.
In breve: questo progetto è incentrato sull'elaborazione di un portale web che possa gestire la possibilità di caricare un file video e riceverne tutte le sue informazioni comodamente.

Attuazione

Per questo progetto, saranno necessarie buone conoscenze nella programmazione web e nella sistemistica, poiché complesso nella sua struttura di rete e nelle sue funzionalità che il portale web dovrà offrire. Il client si connettere al load balancer, quest'ultimo reindirizzerà il client ad uno dei due server meno carico di lavoro. Dopo il server mostrerà una pagina dalla quale sarà possibile caricare un file video mp4 di 500MB, dopo che il client effettuerà l'upload del file, il server ne verificherà tutte le caratteristiche e creerà la pagina con le statistiche, inoltre la cartella associata al client contenente il file video e tutti gli altri file necessari per la le statistiche, infine sempre dalla pagina web sarà possibile creare e scaricare la versione del video: con motion vector o con solo frame I / B / P o tutti i frame I / B / P.

Risultati

Questo progetto è realizzato da zero con i principali linguaggi web (HTML, CSS, PHP) per il sito. I server sono basati sul sistema operativo Linux. Il load balancer utilizza Nginx, mentre i due webserver utilizzano Memcached e apache.
Uno dei punti fondamentali tenuti in considerazione durante il progetto riguarda lo studio per raggiungere la facilità di utilizzo per un utente medio. Questo portale web è difatti utilizzabile anche da un'utenza che non possiede conoscenze informatiche. In conclusione, questo progetto è ora l'unico sistema che consente di caricare un filmato mp4 (di massimo 500MB) in un sito ed ottenerne le statistiche con la possibilità di scaricarne delle informazioni, tutto ciò è gestito da un load balancer e due server che condividono la memoria.

1.3 Scopo

Scopi

- Didattici:
  - Saper creare una progettazione e rispettarla, saper documentare il lavoro, creare i diari e lavorare in team.
- Operativi:
  - Saper creare un load balancer, saper creare due server in cluster, saper creare una Gui Web complessa dinamica, saper utilizzare i log, saper fare uno schema di rete, saper utilizzare Linux server e saper utilizzare la sicurezza opportuna.
    - Vedi QdC in allegato

2 Analisi

2.1 Analisi del dominio

Per questo progetto ci è stato chiesto di creare un sistema in cluster per l'elaborazione di filmati ed estrazione dei vari dati. Oggi per visualizzare le statistiche dei video bisogna passare attraverso molti programmi, ed è tutto meno che immediato. Con il cluster di server ci si assicura che il servizio sia sempre online e grazie alla GUI web il processo diventa user friendly ed intuitivo. Con questo progetto sarà poi possibile visualizzare le statistiche e scaricare il video con i motion vector, solo i frame I/B/P o tutti le immagini che compongono il video in pochi click, senza dover utilizzare svariati programmi.

2.2 Analisi e specifica dei requisiti

Analisi

Portale web che permetta di caricare e quindi di visualizzare le statistiche e scaricare un video con i motion vector, solo i frame I/B/P o tutti le immagini che compongono il video. Questo servizio deve sempre essere disponibile anche se uno dei due server ha qualche problema, inoltre i dati non devono essere condivisi e non cancellati subito ma dopo 1 ora dalla creazione, questo per aggiungere della ridondanza ai dati e quindi consistenza al servizio. Tutto questo servizio deve lavorare su un sistema di tipo Linux server in modo da utilizzare meno risorse possibili, mentre quando un client si connette ad uno dei due server ottiene una pagina web con la quale può interagire. Questo servizio potrà essere utilizzato da chiunque si possa connettere ad internet.

Specifica dei requisiti

ID: REQ-01	
Nome	Funzionamento load balancer basato sul carico
Priorità	1
Versione	1.0
Note	Il load balancer deve reindirizzare le richieste in base al carico dei server
Sotto requisiti	
01	Avere il cluster di server
02	Avere il load balancer configurato

ID: REQ-02	
Nome	Funzionamento delle sessioni
Priorità	1
Versione	1.0
Note	Il server deve gestire le sessioni
Sotto requisiti	
01	Avere il server

ID: REQ-03	
Nome	Funzionamento elaborazione in background dei video caricati
Priorità	1
Versione	1.0
Note	Il server deve elaborare i video in background
Sotto requisiti	
01	Avere il server

ID: REQ-04	
Nome	Funzionamento upload di un filmato (massimo 500MB)
Priorità	1
Versione	1.0
Note	Il server deve poter ricevere un filmato di massimo 500MB
Sotto requisiti	
01	Avere il server

ID: REQ-05	
Nome	Produrre statistiche sul video
Priorità	1
Versione	1.0
Note	Il server deve generare: • Statistiche sui frame • Grafico sui frame
Sotto requisiti	
01	Avere il server
02	REQ-04

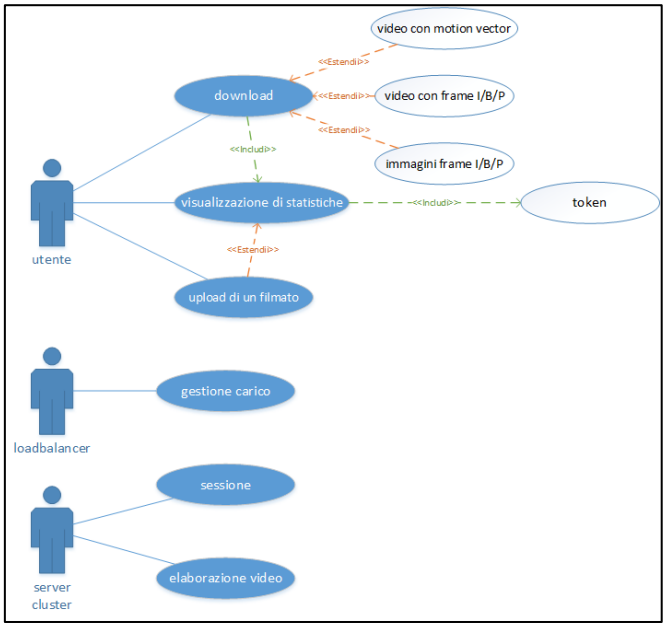
ID: REQ-06	
Nome	Possibilità di scaricare differenti contenuti
Priorità	1
Versione	1.0
Note	Il server deve generare: • Video con motion vector • Video con frame I/B/P • Immagini con tutti i frame I/B/P
Sotto requisiti	
01	Avere il server
02	REQ-04

ID: REQ-07	
Nome	Cancellazione dati sul db e sul server
Priorità	1
Versione	1.0
Note	Il server e il db devono cancellare i dati dopo 1 ora dalla loro creazione
Sotto requisiti	
01	Avere il server
02	Avere il db

Spiegazione elementi tabella dei requisiti:

- **ID**: identificativo univoco del requisito
- **Nome**: breve descrizione del requisito
- **Priorità**: indica l'importanza di un requisito nell'insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.
- **Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.
- Sulla documentazione apparirà solamente l'ultima versione, mentre le vecchie dovranno essere inserite nei diari.
- **Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.
- **Sotto requisiti**: elementi che compongono il requisito.

2.3 Use case

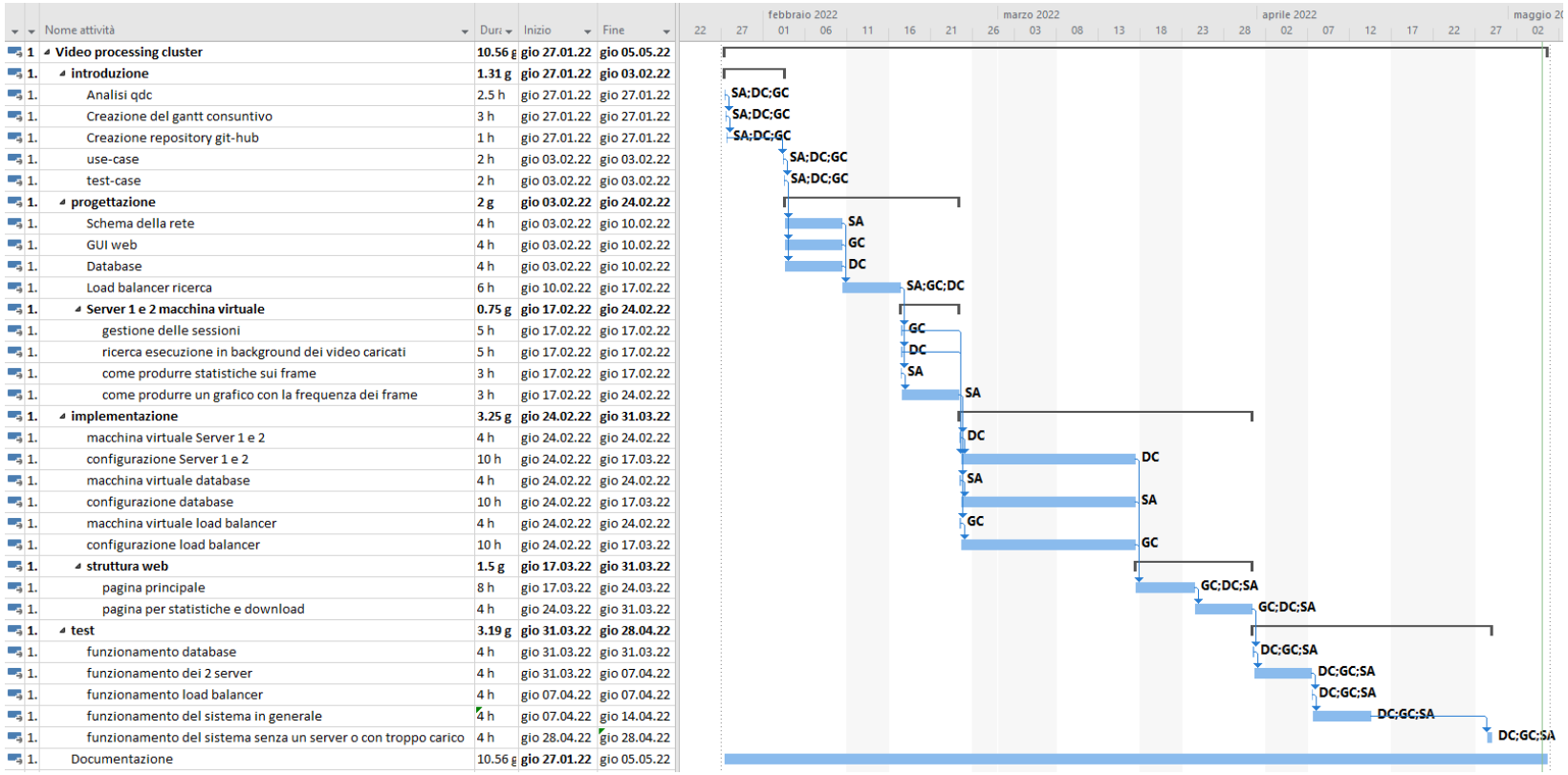


All'interno dell'applicativo abbiamo inserito 3 tipi diversi di attori, più specificatamente:

- Utente: questi attori potranno interagendo con la pagina web servita dal server: caricare un filmato e quindi vedere la rispettiva pagina web con le statistiche oppure senza caricare un filmato essi potranno inserendo un token relativo ad una specifica statistica vedere quest'ultima e in entrambi i casi si può scaricare un video composto dai motion vector, un video con i frame I/B/P e un file zip contenente tutte le immagini con i frame I/B/P.
- Load balancer: questo attore è singolo e il suo compito è quello di gestire il carico, indirizzando le richieste ad uno dei due server in base a dei criteri definiti.
- Server cluster: questi attori sono due e lavorano in cluster ovvero condividono delle risorse, come la gestione delle sessioni e l'elaborazione dei video.

2.4 Pianificazione

Di seguito c'è la pianificazione preventiva, nel Gantt preventivo è stimato il tempo necessario per completare le task e le rispettive sotto task del progetto.



	Nome attività	Durata	Inizio	Fine
1	**Video processing cluster**	**10.56 g**	**gio 27.01.22**	**gio 05.05.22**
1.	**introduzione**	**1.31 g**	**gio 27.01.22**	**gio 03.02.22**
1.	Analisi qdc	2.5 h	gio 27.01.22	gio 27.01.22
1.	Creazione del gantt consuntivo	3 h	gio 27.01.22	gio 27.01.22
1.	Creazione repository git-hub	1 h	gio 27.01.22	gio 27.01.22
1.	use-case	2 h	gio 03.02.22	gio 03.02.22
1.	test-case	2 h	gio 03.02.22	gio 03.02.22
1.	**progettazione**	**2 g**	**gio 03.02.22**	**gio 24.02.22**
1.	Schema della rete	4 h	gio 03.02.22	gio 10.02.22
1.	GUI web	4 h	gio 03.02.22	gio 10.02.22
1.	Database	4 h	gio 03.02.22	gio 10.02.22
1.	Load balancer ricerca	6 h	gio 10.02.22	gio 17.02.22
1.	**Server 1 e 2 macchina virtuale**	**0.75 g**	**gio 17.02.22**	**gio 24.02.22**
1.	gestione delle sessioni	5 h	gio 17.02.22	gio 17.02.22
1.	ricerca esecuzione in background dei video caricati	5 h	gio 17.02.22	gio 17.02.22
1.	come produrre statistiche sui frame	3 h	gio 17.02.22	gio 17.02.22
1.	come produrre un grafico con la frequenza dei frame	3 h	gio 17.02.22	gio 24.02.22
1.	**implementazione**	**3.25 g**	**gio 24.02.22**	**gio 31.03.22**
1.	macchina virtuale Server 1 e 2	4 h	gio 24.02.22	gio 24.02.22
1.	configurazione Server 1 e 2	10 h	gio 24.02.22	gio 17.03.22
1.	macchina virtuale database	4 h	gio 24.02.22	gio 24.02.22
1.	configurazione database	10 h	gio 24.02.22	gio 17.03.22
1.	macchina virtuale load balancer	4 h	gio 24.02.22	gio 24.02.22
1.	configurazione load balancer	10 h	gio 24.02.22	gio 17.03.22
1.	**struttura web**	**1.5 g**	**gio 17.03.22**	**gio 31.03.22**
1.	pagina principale	8 h	gio 17.03.22	gio 24.03.22
1.	pagina per statistiche e download	4 h	gio 24.03.22	gio 31.03.22
1.	**test**	**3.19 g**	**gio 31.03.22**	**gio 28.04.22**
1.	funzionamento database	4 h	gio 31.03.22	gio 31.03.22
1.	funzionamento dei 2 server	4 h	gio 31.03.22	gio 07.04.22
1.	funzionamento load balancer	4 h	gio 07.04.22	gio 07.04.22
1.	funzionamento del sistema in generale	4 h	gio 07.04.22	gio 14.04.22
1.	funzionamento del sistema senza un server o con troppo carico	4 h	gio 28.04.22	gio 28.04.22
1.	Documentazione	10.56 g	**gio 27.01.22**	gio 05.05.22

Figura 1 – GANTT preventivo

Le annotazioni a fianco di ogni attività sono i nomi delle persone le quali devono occuparsi di tale: GC = Gioele Cavallo, DC = Damian Campesi, SA = Samuele Abbà.

2.5 Analisi dei mezzi

Per questo progetto ci sarà fornito 1 PC della scuola ad ognuno, sul quale ci sarà un sistema operativo Windows 10 Pro.

2.5.1 Software

I software utilizzati per questo progetto sono Virtualbox per la creazione delle macchine virtuali Linux, HTML, CSS, PHP, JS e apache per il sito applicativo, Memcached per mettere i due server in cluster e Nginx per il load balancer. Poi per processare i file utilizziamo ffmpeg e ffprobe, per creare i file con le informazioni sul video utilizziamo invece Python per eseguire ffstats_converter.py.

2.5.2 Hardware

- HDD e SSD per memorizzare le VM ed un Computer con:
- CPU Intel Core i7-7700
- RAM 16 GB

3 Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all'esecutore di evitare fraintendimenti e imprecisioni nell'implementazione del prodotto.

3.1 Design dell'architettura del sito web

Il sito web è strutturato nel modo seguente:

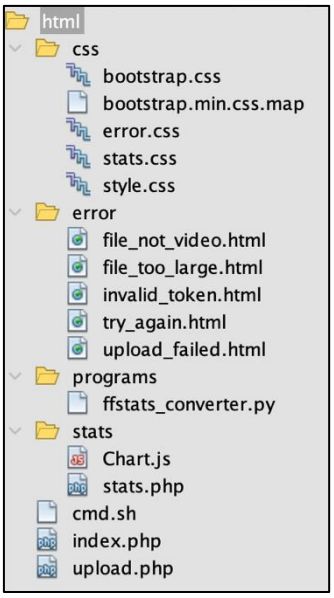


Figura 2 – Design dell'architettura

- **html**: è la root del sito web.
- **index.php**: è la pagina principale del sito che viene mostrata appena ci si connette al load balancer.
- **upload.php**: è la pagina che si occupa di far partire lo script, essa viene richiamata da index.php.
- **cmd.sh**: è il file bash, che permette di:
  - Crea un file .txt con tutte le informazioni del video.
  - Dal file .txt viene generato un file csv contenente tutte le informazioni del video formattate.
  - Crea un video con i motion vector del video.
  - Crea i 3 video con solo i frame I,B,P.
  - Crea la zip contenente tutti i frame del video.
- **css**: questa cartella contiene i file di stile delle varie pagine web.
- **error**: questa cartella contiene le pagine di errore che vengono mostrati nel caso di errore nelle varie pagine web.
- **programs**: questa cartella contiene il file python che permette tramite il file txt generato contenente le informazioni sul video, di creare il csv contenente le informazioni del video formattate.
- **stats**: questa cartella contiene la pagina web che serve a visualizzare le statistiche relative al file mp4 e il file Chart.js che è la libreria per creare i grafici in HTML5, che viene utilizzata nella pagina delle statistiche.Design delle interfacce.

3.1.1 Progettazione interfacce home page

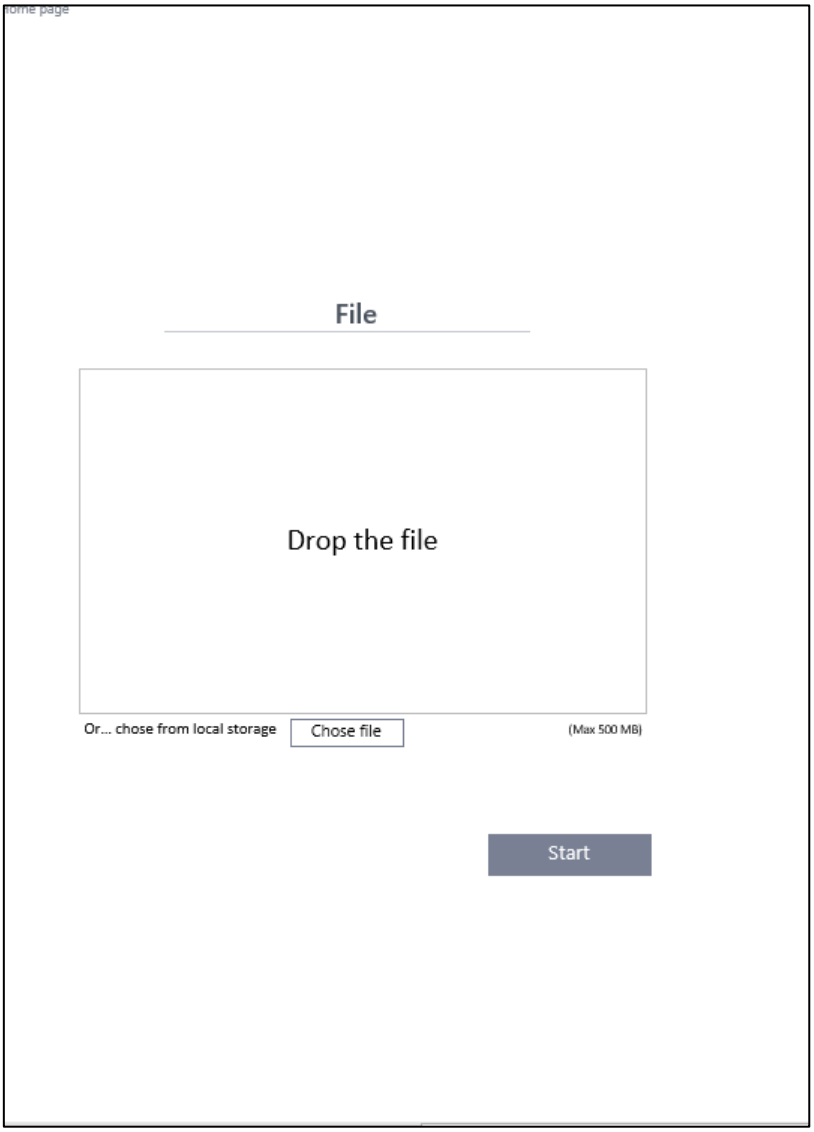


Figura 3 – Progettazione interfacce home page

La home page, è l'interfaccia che si visualizza non appena ci si connette al server, tramite il load balancer. In questa pagina si potrà caricare un video mp4 da massimo 500 MB, tramite il bottone "chose file" o trascinando il file nell'area demarcata, infine premendo sul bottone start, i dati verranno validati e processati.

3.1.2 Progettazione interfacce stats page

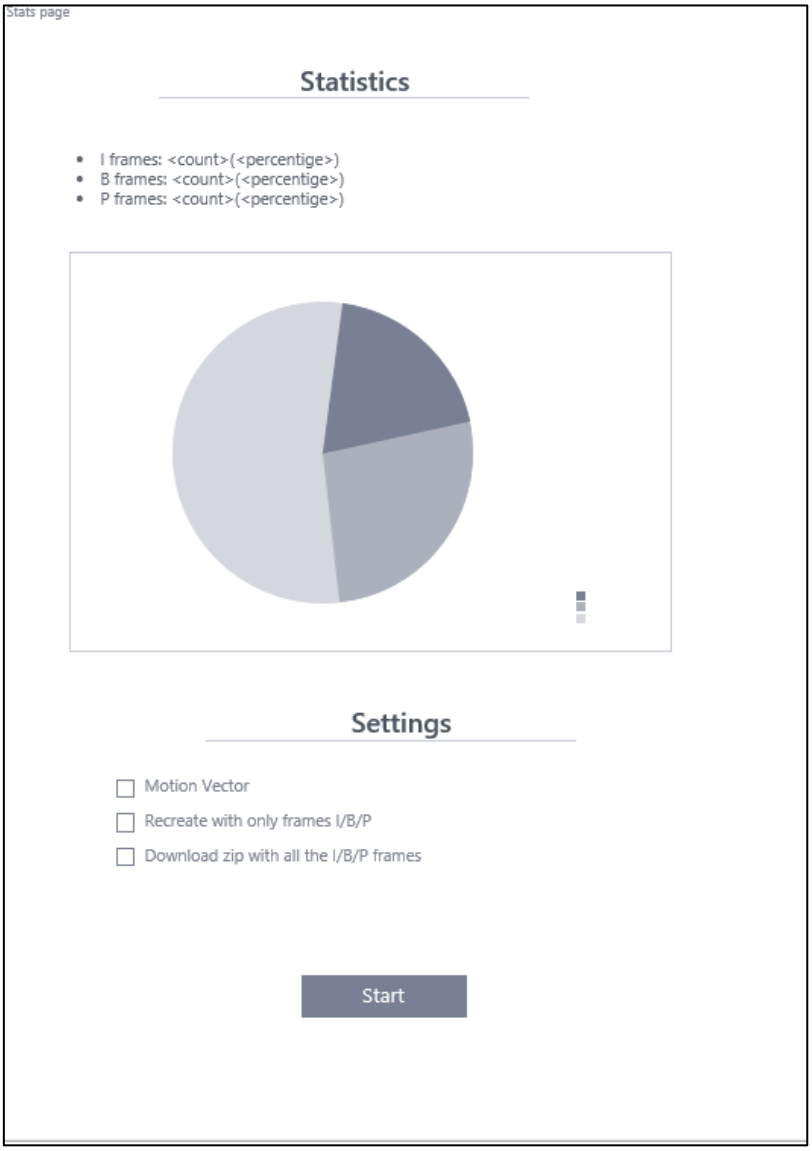


Figura 4 – Progettazione interfacce stats.php page

La stats page, è l'interfaccia che si visualizza dopo che si ha caricato e trattato i dati. In questa pagina verranno visualizzate tutte le statistiche del file caricato nella schermata precedente tramite un grafico. Infine è disponibile selezionare e successivamente scaricare i motion vector, i frame I/B/P o scaricare una zip contenente tutti i frame.

3.2 Design procedurale

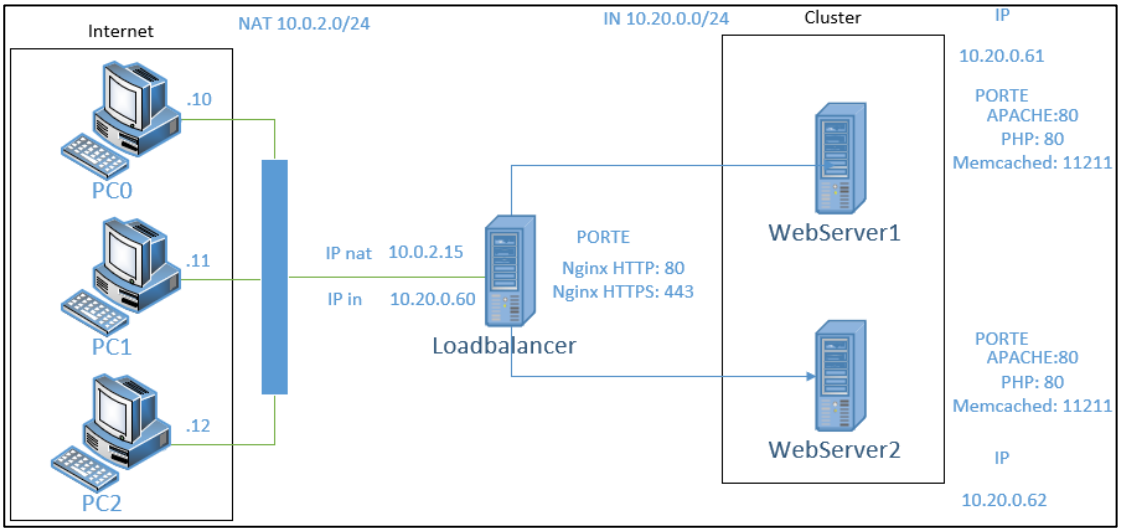


Figura 5 – Design dello schema di rete

Questo è lo schema di rete di base, su di esso si basa tutto il progetto.
Partendo da sinistra c'è la rete nat 10.0.2.0/24 nella quale ci sono i client, da essa tutti i client si possono connettere al load balancer.
Grazie al load balancer si possono reindirizzare le chiamate a uno dei due server dalla rete nat alla rete in 10.20.0.0/24.
Il load balancer ha due schede di rete una 10.0.2.15 che si interfaccia sulla rete nat ovvero la 10.0.2.0/24 e l'altra 10.20.0.60 che si interfaccia sulla rete in ovvero la 10.20.0.0/24, inoltre sul load balancer è installato Nginx (web server che permette di servire pagine web statiche o dinamiche in base al carico e al tre funzionalità), questo servizio utilizza la porta 80 per le richieste http mentre la porta 443 per le richieste https.
I due server sono pressoché uguali le uniche differenze sono nell'ip che nel caso del primo è 10.20.0.61 mentre nel secondo è 10.20.0.62.
Questi due server hanno un servizio apache e php che permette di fornire le pagine web dinamiche, sulla porta 80.
Infine i due server lavorano in cluster e quindi condividono la stessa memoria, per fare questo bisogna installare Memcached che permette di collegare le due memorie, lavorando sulla porta 11211.
Per questi server sono state utilizzate macchine virtuali.

4 Implementazione

4.1 Configurazioni

4.1.1 php.ini

Percorso file:
- "/etc/php/8.1/apache2/php.ini"

Modificare:
- upload_max_filesize=500M
- max_file_uploads=1
- session.save_handler= memcached.e
- session.save_path="10.20.0.62:11211,10.20.0.61:11211"
- post_max_size=500M

Spiegazione:
1. Nel file localizzato in "/etc/php/8.1/apache2/php.ini" bisogna modificare nella sezione "File Uploads" alla riga 580 la variabile upload_max_filesize settandola a 500M. Alla riga 853 si deve modificare max_file_uploads ad 1, così da poter fare l'upload di un solo file alla volta. Alla riga 1326 è necessario cambiare session.save_handler e settarla a memcached.e inoltre alla riga 1334 occorre settare session.save_path a "10.20.0.62:11211,10.20.0.61:11211". Infine nella sezione "Data Handling" è necessario cambiare alla riga 698 il post_max_size a 500M.

4.1.2 memcached.conf

Percorso file:
- /etc/memcached.conf

Modificare:
- ~m 1024
- ~p 11211

Spiegazione:
1. Abbiamo aumentato la memoria RAM a disposizione grazie al parametro ~m a 1024. Abbiamo modificato la porta di ascolto grazie al parametro ~p alla 11211.

4.1.3 load-balancer.conf

Nel file presente in "/etc/nginx/conf.d/load-balancing.conf "bisogna inserire il codice seguente per configurare il load balancer utilizzando nginx.

```
upstream backend {
    least_conn;
    server 192.168.10.11;
    server 192.168.10.12;
}

server {
    listen      80;
    server_name loadbalancing.com;
    location / {
        proxy_redirect      off;
        proxy_set_header    X-Real-IP $remote_addr;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    Host $http_host;
        proxy_pass http://backend;
    }
}
```

Figura 6 – load-balancer.conf

4.1.4 Capitolo configurazione cartella condivisa

Per poter avere uno spazio di archiviazione condiviso tra i due server abbiamo creato una cartella condivisa tra loro. Per farlo abbiamo utilizzato l'NFS.
Per prima cosa abbiamo installato nfs-kernel-server sul server che ospiterà la cartella condivisa. Poi abbiamo creato la cartella da rendere condivisa, nel nostro caso "/etc/var/www/html/upload". Quindi abbiamo impostato il proprietario con "sudo chown nobody:nogroup /etc/var/www/html/upload" e dopo abbiamo modificato i permessi con "sudo chmod 777 /etc/var/www/html/upload".

Il file "/etc/exports", consente di modificare la lista di controllo degli accessi, per le macchine che si connettono come client NFS. Abbiamo aggiunto le righe qui sotto evidenziate in modo che la cartella condivisa sia disponibile a tutti e due i server.

```
1  #/etc/exports: the access control list for filesystems which may be exported
2  #               to NFS clients.  See exports(5).
3  #
4  # Example for NFSv2 and NFSv3:
5  # /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
6  #
7  # Example for NFSv4:
8  # /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
9  # /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
10 #
11 /var/www/html/upload 10.20.0.61(rw,sync,no_subtree_check)
12 /var/www/html/upload 10.20.0.60(rw,sync,no_subtree_check)
```

Figura 7 – configurazione cartella condivisa

4.2 Applicativo Web

4.2.1 File index.php

Il file index.php è la pagina web principale che entrambi i server forniscono appena vengono interpellati. Questa pagina permette di caricare un file video mp4 tramite un input file e successivamente inviarlo tramite il bottone submit (“submit”) ad un’altra pagina (upload.php) che si occuperà di salvare il file e di creare i relativi dati.
Nella pagina è presente un secondo bottone submit (“submitToken”), che permette di inserire il token generato dall’upload del file mp4 e di vedere le statistiche relative ad esso.

```
<!doctype html>
<html>
    <head>
        <link href="css/style.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div class="main">
            <div>
                <h1 class="title">UPLOAD</h1><hr class="title_divisor"><br>
                <form action="upload.php" method="POST" class="subContainer">
                    <input type="number" name="id" placeholder="Inserire un token:" min="0" max="9999999999" class="other">
                    <input type="submit" name="submitToken" value="Mostra" id="show">
                </form>
            </div>
            <br>
            <div>
                <form method="POST" action="upload.php" class="subContainer">
                    <input type="file" name="file" accept="video/mp4" class="other">
                    <input type="submit" name="submit" value="Invia" id="send">
                </form>
            </div>
        </div>
    </body>
</html>
```

Figura 8 – codice index.php

4.2.2 File upload.php

Il file upload.php è la pagina che viene richiamata dall'index dopo un submit, essa si occupa di fare i controlli opportuni sul file caricato o sul token inserito.
Innanzi tutto setta la variabile globale *token*, poi prende dal file di configurazione di php (php.ini) i server Memcached in ascolto, aggiungendoli all'oggetto Memcached. Viene istanziata la variabile per il percorso delle pagine di errore e istanziati gli handler per i file di log.

```php
global $token;

// gets all the memcached servers listening from the php.ini .
$servers = explode(",", ini_get("session.save_path"));
$c = count($servers);
for ($i = 0; $i < $c; ++$i) {
    $servers[$i] = explode(":", $servers[$i]);
}

$mem = new Memcached();
$mem->addServers($servers);

$error_dir = "error/";

$handlerInfo = fopen("/var/www/log/info.log", "a");
$handlerError = fopen("/var/www/log/error.log", "a");
```

Figura 9 – dichiarazione variabili upload.php 1

Dopodiché viene controllato se è stato premuto il bottone submit per il caricamento del file. Se così fosse vengono istanziate le variabili utili per l'upload del file e per il controllo.

```php
// this is the token value and the name of the directory.
$time = time();

// where files are saved.
$target_dir = "upload/";

$filename = str_replace(" ", "_", $time . '_' . basename($_FILES["file"]["name"]));

// the path of the uploaded file.
$target_file = $target_dir . $time . "/" . $filename;

// this variable is used in case the loading should not be successful.It will be setted to 0.
$uploadOk = 1;

// the max file size in Byte.
$maxSize = 500000000;
```

Figura 10 – dichiarazione variabili upload.php 2

Dopo come primo controllo viene verificato che il file caricato sia un video di tipo mp4, nel caso in cui non lo fosse mostra la pagina di errore e setta la variabile *uploadOK* a 0.

```
// open error page if file is not a mp4.
if ((strtolower(pathinfo($target_file, PATHINFO_EXTENSION))) != "mp4") {
    require $error_dir . "file_not_video.html";
    $uploadOk = 0;
}
```

Figura 11 – controllo tipo del file

Dopodiché come secondo controllo viene verificato che la dimensione del file sia minore di 500MB, in caso contrario mostra la pagina di errore e setta la variabile *uploadOK* a 0.

```
// open error page if file size is greater than the maxSize.
if ($_FILES['file']['size'] > $maxSize) {
    require $error_dir . "file_too_large.html";
    $uploadOk = 0;
}
```

Figura 12 – controllo dimensioni del file

Poi come quarto controllo viene verificato che la cartella *upload* esista, nel caso in cui così non fosse viene creata e le vengono dati tutti i permessi.

```
// if the dir not exists, it creates it.
if (!file_exists($target_dir)) {
    mkdir($target_dir, 0777, true);
}
```

Figura 13 – controllo cartella generale esistente e aggiunta permessi

Come quinto controllo viene verificato che l'upload non ci siano stati problemi e che quindi possa avvenire. Poi viene controllato che la cartella specifica per l'upload del file esista, in caso contrario verrà creata e le verranno dati tutti i permessi.

```php
// give permissions to the folder to make it readable and writable.
if (!file_exists("$target_dir$time/")) {
    mkdir("$target_dir$time/", 0777, true);
    chmod("$target_dir$time/", 0777);
}
```

Figura 14 – controllo cartella specifica esistente e aggiunta permessi

Come ultimo punto viene verificato che l'upload sia avvenuto con successo. Dopo viene eseguito il file cmd.sh, ed in seguito viene eseguito il comando per dare i permessi ricorsivi alla cartella del file caricato. Inoltre viene settate le variabili globale *token* che è il percorso univoco della cartella contenente i file. Successivamente il token viene salvato all'interno di Memcached.
Infine viene scritto nel logo l'operazione e si richiama la pagina delle statistiche.
In caso l'upload non dovesse andare a buon fine viene scritto nell' errorLog l'informazione e aperta la pagina web di errore.

```php
if (move_uploaded_file($_FILES["file"]["tmp_name"], $target_file)) {

    // this is the path to the txt file that ontains all the information needed to create the statistics
    $target_txt = "$target_dir$time/testo.txt";
    $shell_exec_video = "sh cmd.sh '$target_file' '$target_txt' '$target_dir$time/'";

    // the cmd.sh file is executed, which creates statistics on the file and the various mp4s.
    echo shell_exec($shell_exec_video);
    $comandoPerPermessiCartella = "chmod 777 -R $target_dir$time/";

    echo shell_exec("$comandoPerPermessiCartella");

    // setted globals variable token.
    $GLOBALS["token"] = $time;

    //add token to memcached.
    $mem->add($GLOBALS["token"], $time);

    // write info upload on log.
    $txt = date("d.m.y H:i:s") . " : INFO : file " . $time . " uploaded from " . $_SERVER['REMOTE_ADDR'] . PHP_EOL;
    fwrite($handlerInfo, $txt);

    require "stats/stats.php";
} else {

    // write error failed upload on log.
    $txt = date("d.m.y H:i:s") . " : ERROR : failed upload of " . $time . " from " . $_SERVER['REMOTE_ADDR'] . PHP_EOL;
    fwrite($handlerError, $txt);
    require $error_dir . "upload_failed.html";
}
```

Figura 15 – viene spostato il file e creata la cartella con i relativi file

4.2.3 File cmd.sh

Il file cmd.sh è uno script che viene richiamato dall'upload per creare i file necessari per le statistiche e per i download, esso riceve due parametri uno è il percorso del video dal quale prendere le informazioni e l'altro è il file nel quale andrà a salvare le informazioni.

1. Il comando ffprobe serve per creare un file txt contenente tutte le info dal video
2. Il comando python3 programs/ffstats_converter.py serve per creare un file csv dal file txt
3. Il primo comando ffmpeg serve per creare un video con i motion vector
4. Il secondo ffmpeg serve per creare un video con i soli frame I
5. Il terzo ffmpeg serve per creare un video con i soli frame B
6. Il quarto ffmpeg serve per creare un video con i soli frame P
7. Poi viene creata la cartella nella quale verranno successivamente inseriti tutti i frame del video
8. L'ultimo comando ffmpeg serve per mettere nella cartella creata in precedenza tutti i frame del video
9. Infine l'ultimo comando creerà la zip della cartella che contiene tutti i frame del video

```
#!/bin/bash

MOTION_VECTOR="$3motion.mp4"
ffprobe -v error -select_streams v:0 -show_frames $1 > $2
python3 programs/ffstats_converter.py $2
ffmpeg -flags2 +export_mvs -i $1 -vf codecview=mv=pf+bf+bb $MOTION_VECTOR &

ffmpeg -hide_banner -loglevel panic -i $1 -vf "select='eq(pict_type\,I)',showinfo" "$3I_frames.mp4" &
ffmpeg -hide_banner -loglevel panic -i $1 -vf "select='eq(pict_type\,B)',showinfo" "$3B_frames.mp4" &
ffmpeg -hide_banner -loglevel panic -i $1 -vf "select='eq(pict_type\,P)',showinfo" "$3P_frames.mp4" &
mkdir "$3zip"
ffmpeg -hide_banner -loglevel panic -i $1 "$3zip/frame_%d.jpg"
zip -q -r "$3zip.zip" "$3zip"
```

Figura 16 – cmd.sh

4.2.4 File stats.php

Il file stats.php è uno script che viene richiamato dall'upload. Esso permette di leggere i vari file creati e di mostrarli tramite grafici nella pagina web, permettendo il download del file zip e degli altri mp4 creati precedentemente.
Il file contiene il metodo *getCsv* che ritorna i dati del csv.

```
// return the data of the CSV.
function getCsv($filePath)
{
    $nomeFile = $filePath . "testo.txt.csv";
    $data = array();

    if (($handle = fopen($nomeFile, "r")) !== false) {
        while (($subData = fgetcsv($handle, 4096, ",")) !== false) {
            $data[] = $subData;
        }
            fclose($handle);
    }

    return $data;
}
$data = getCsv($filePath);
```

Figura 17 – il metodo getCsv del file stats

Questo file contiene diversi loop per ottenere le statistiche dai file e successivamente per calcolarli.

Le variabili successivamente conterranno i dati relativi alla qualità di frame inoltre ci sono gli array per i colori e per le dimensioni delle barre dell'istogramma.

```
// get the info about the data.
// get count of frame I, B, P and set a different color for each type of frame.
$quantitaB = 0;
$quantitaI = 0;
$quantitaP = 0;
$colorsBar = array();
$sizeBar = array();
```

Figura 18 – variabili del file stats

Questo loop serve per ottenere il colore della barra in funzione al tipo di frame, ed il conteggio dei vari frames.

```
for ($i = 0; $i < COUNT($data); $i++) {
    for ($j = 0; $j < COUNT($data[$i]); $j++) {
        if ($j == 17 && $i != 0) {
            if ($data[$i][$j] == 'B') {
                $quantitaB++;
                $colorsBar[] = "red";
            } elseif ($data[$i][$j] == 'I') {
                $quantitaI++;
                $colorsBar[] = "green";
            } else {
                $quantitaP++;
                $colorsBar[] = "blue";
            }
        }
    }
}
```

Figura 19 – logica per ottenere il numero dei tipi di frames e il colore

Il secondo loop serve per ottenere le informazioni del time stamp e la grandezza delle barre per i frame.

```
// get the timestamp information about each frame.
$timeStamp = array();
for ($i = 2; $i < COUNT($data); $i++) {
    $timeStamp[] = $data[$i][4];
    $sizeBar[] = $data[$i][12];
}
```

Figura 20 – loop per ottenere il time stamp e il size bar

Infine vengono create le variabili con le percentuali da visualizzare.

```
$totale = $quantitaB + $quantitaI + $quantitaP;
$B = round($quantitaB / $totale * 100);
$I = round($quantitaI / $totale * 100);
$P = round($quantitaP / $totale * 100);
```

Figura 21 – loop per ottenere il time stamp e il size bar

All'interno del file c'è il codice html della pagina che permette di visualizzare due grafici: uno a torta per le percentuali statistiche e uno per vedere la frequenza dei frame.

```
<div id="containerPie">
    <canvas id="pieChart"></canvas>
</div>
<br>
<br>
<br>
<br>
<div id="containerPie" >
    <canvas id="frequencyChart"></canvas>
</div>
```

Figura 22 – i div contenenti i canvas per la visualizzazione dei grafici

In fine c'è l'html con i percorsi dov'è possibile scaricare i file creati precedentemente.

```
<div id="drop_not_file_zone">
    <ul>
        <li><label for="change1"><a href='<?php echo $filePath . "motion.mp4" ?>'>Motion vector</a></label></li>
        <li><label for="change2"><a href='<?php echo $filePath . "I_frames.mp4" ?>'>Recreate with only frames I</a></label></li>
        <li><label for="change2"><a href='<?php echo $filePath . "B_frames.mp4" ?>'>Recreate with only frames B</a></label></li>
        <li><label for="change2"><a href='<?php echo $filePath . "P_frames.mp4" ?>'>Recreate with only frames P</a></label></li>
        <li><label for="change3"><a href='<?php echo $filePath . "zip.zip" ?>'>Download Zip with all the frames</a></label></li>
    </ul>
</div>
```

Figura 23 – loop per ottenere il time stamp e il size bar

4.2.5 File chart.js

Il file chart.js è uno script che viene utilizzato da stats.php per creare i grafici delle statistiche, questo file js lo abbiamo scaricato da (https://www.chartjs.org/docs/latest/), esso è una libreria JavaScript open source per creare grafici interattivi e animati su pagine web. Chart.js sfrutta l'elemento HTML5 canvas. Chart.js permette a grafici, designer e programmatori di visualizzare i dataset tramite diversi tipi di grafici, tra cui grafici a linee, a barre (orizzontali e verticali), radar, polari, ad area, a torta e ad anello. Inoltre, con Chart.js è possibile modificare ed estendere queste tipologie di grafici e scriverne di nuovi.

```
/*!
 * Chart.js v3.7.1
 * https://www.chartjs.org
 * (c) 2022 Chart.js Contributors
 * Released under the MIT License
 */
!function(t,e){"object"==typeof exports&&"
/*!
 * @kurkle/color v0.1.9
 * https://github.com/kurkle/color#readme
 * (c) 2020 Jukka Kurkela
 * Released under the MIT License
 */const r={0:0,1:1,2:2,3:3,4:4,5:5,6:6,7:
```

Figura 24 – prime righe del file Chart.js

4.2.6 File ffstats_converter.py

Il file ffstats_converter.py è un file python che viene utilizzato da cmd.php per convertire il file txt contenente tutte le informazioni del video in un file csv contenente tutte le informazioni più facilmente consultabili.

4.2.7 File *.css

Questi file, sono i file relativi allo stile delle pagine web, essi permettono alle pagine web di migliorare abbellendo l'estetica.

4.3 Eliminazione Files nel server

Abbiamo creato questo script per poter eliminare dopo un'ora tutti i files che vengono creati una volta caricato il file video.

4.3.1 Script

Per prima cosa, lo script entra nella cartella upload, dopo memorizza la data corrente meno un'ora. Dopodiché svolge un'iterazione per ogni elemento della cartella, ed elimina quest'ultima nel caso in qui la data di creazione sia più vecchia della data memorizzata all'inizio.

```
#!/bin/bash

cd /var/www/html/upload

current_date=$(date --date="-1 hours" +'%s')

for i in $(ls)
do
        creation=$(stat -c %x $i | cut -c1-19)
        creation_date=$(date -d "$creation" "+%s")
        if [ "$current_date" -ge "$creation_date" ];
        then
                rm -r $i
                echo "removed ${i}"
        fi
done
```

Figura 25 – Script per l'eliminazione dei file

4.3.2 Crontab

Per continuare a eliminare i contenuti della cartella di upload con lo script visto in precedenza, abbiamo utilizzato il Crontab. Grazie alla ultima riga contenuta in questo file possiamo ripetere l'esecuzione dello script nel tempo, ed in questo caso abbiamo deciso di ripeterlo sempre ad ogni ora.

```
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 */1 * * * /var/www/script/script.sh
```

Figura 26 – Crontab

5 Test

5.1 Protocollo di test

Test Case: **Riferimento**:	TC-01 REQ-01	**Nome:**	Funzionamento load balancer basato sul carico
Descrizione:	Verificare il corretto funzionamento del load balancer		
Prerequisiti:	Avere il cluster di server Avere il load balancer configurato		
Procedura:	1. Saturare la potenza di calcolo del primo server 2. Collegarsi al load balancer 3. Verificare che si è collegati al secondo server 4. Ricollegarsi al load balancer 5. Verificare che si è collegati nuovamente al secondo server		
Risultati attesi:	Si aspetta che la procedura venga soddisfatta e che alla fine si è collegati al secondo server ovvero quello con meno carico.		

Test Case: **Riferimento**:	TC-02 REQ-02	**Nome:**	Funzionamento delle sessioni
Descrizione:	Verificare il corretto funzionamento della gestione delle sessioni		
Prerequisiti:	Avere il cluster di server		
Procedura:	1. Collegarsi al primo server 2. Iniziare un'interazione 3. Collegarsi al secondo server 4. Verificare che siano apparse le statistiche sul file caricato in precedenza		
Risultati attesi:	Si aspetta che le statistiche visualizzate siano riguardanti quelle del file caricato.		

Test Case: **Riferimento**:	TC-03 REQ-03	**Nome:**	Funzionamento elaborazione in background dei video caricati
Descrizione:	Verificare il funzionamento dell'elaborazione in background		
Prerequisiti:	Avere un server funzionante		
Procedura:	1. Collegarsi al server 2. Caricare un file nella GUI 3. Far avviare i processi 4. Chiudere la GUI 5. Riaprire la GUI dopo un po' di tempo 6. Verificare che i processi siano continuati		
Risultati attesi:	Si aspetta che vengano visualizzate le statistiche una volta riaperto il browser.		

Test Case: **Riferimento**:	TC-04 REQ-04	**Nome:**	Funzionamento upload di un filmato
Descrizione:	Verificare il funzionamento dell'upload di un filmato di massimo 500MB		

Prerequisiti:	Avere un server funzionante
Procedura:	1. Collegarsi al server 2. Caricare un file nella GUI di dimensioni maggiori di 500MB 3. Verificare che venga mostrato un errore concernente la dimensione troppo grande del filmato caricato. 4. Ricaricare un file nella GUI di dimensioni minori di 500MB 5. Verificare che non vengano dati errori
Risultati attesi:	Si aspetta che inizialmente le procedure del server non vengano eseguite, mentre dopo il secondo upload esse vengano eseguite e appaiano le statistiche.

Test Case: **Riferimento**:	TC-05 REQ-05	**Nome:**	Produrre statistiche sul video
Descrizione:	Verificare che vengano create le statistiche sul video		
Prerequisiti:	Avere un server funzionante		
Procedura:	1. Collegarsi al server 2. Caricare un file nella GUI 3. Verificare che vengano mostrate le statistiche inerenti ad esso		
Risultati attesi:	Si aspetta che vengano visualizzate le statistiche.		

Test Case: **Riferimento**:	TC-06 REQ-06	**Nome:**	Possibilità di scaricare differenti contenuti
Descrizione:	Verificare che ci sia la possibilità di scaricare i file nelle diverse modalità		
Prerequisiti:	Avere un server funzionante		
Procedura:	1. Collegarsi al server 2. Caricare un file nella GUI 3. Verificare che vengano creati i file seguenti: a. Video con motion vector b. Video con frame I/B/P c. Immagini con tutti i frame I/B/P		
Risultati attesi:	Si aspetta che vengano creati correttamente i file sopra citati.		

Test Case: **Riferimento**:	TC-07 REQ-07	**Nome:**	Cancellazione dati sul server
Descrizione:	Verificare che i dati sul server vengano cancellati dopo 1 ora		
Prerequisiti:	Avere un server funzionante		
Procedura:	1. Collegarsi al server 2. Caricare un file 3. Attendere 1 ora 4. Verificare che si sia svuotata la cache del server e che la cartella locale sia cancellata 5. Verificare che si sia svuotato		
Risultati attesi:	Si aspetta che i dati più vecchi di 1 ora non siano più memorizzati.		

5.2 Risultati test

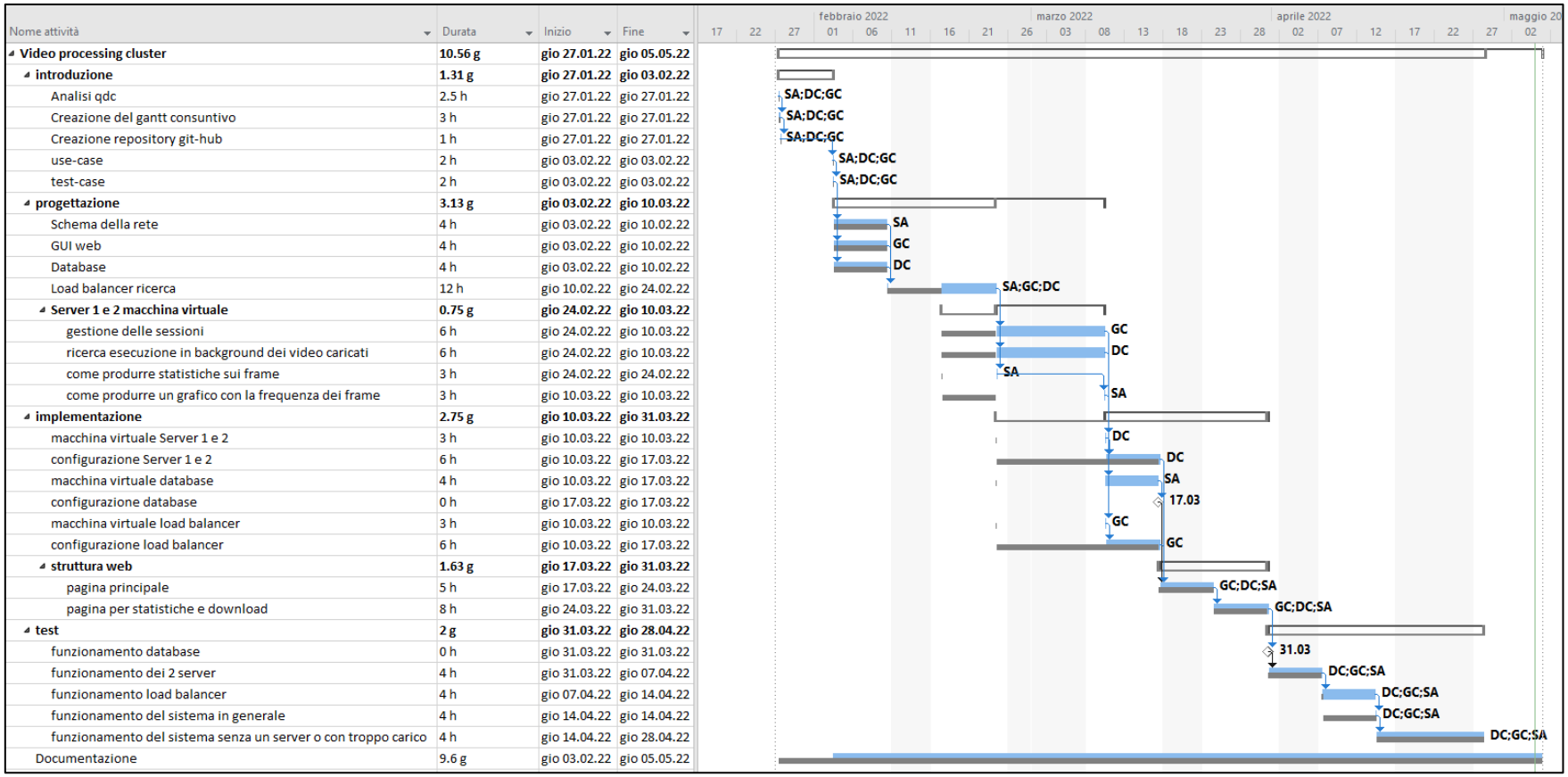
Test Case	Funzionamento	Commento	Data
TC-01	NON OK	Il client quando si connette viene reindirizzato al load balancer con il numero di connessioni minore e non con il carico minore.	07.04.2022
TC-02	OK	Il client in caso dovesse ricollegarsi e viene reindirizzato su un server diverso dal precedente, deve poter continuare la sua esperienza senza interruzioni.	14.04.2022
TC-03	OK	Quando viene chiusa la GUI e si riapre, il processo dei file non si deve fermare.	14.04.2022
TC-04	OK	L'utente può caricare solo un file mp4 con dimensioni inferiori a 500 MB.	14.04.2022
TC-05	OK	Il sever crea le statistiche inerenti al file mp4 caricato e vengono visualizzate nel sito.	14.04.2022
TC-06	OK	È possibile scaricare i vari file creati dal server: motion vector, video con solo frames I/B/P, zip con i singoli frames.	28.04.2022
TC-07	OK	Le cartelle create da più di 1 ora vengono cancellate.	28.04.2022

5.3 Mancanze/limitazioni conosciute

A livello estetico manca un po' di parte grafica che renderebbe il sito più accattivante. Inoltre un'aggiunta di un input per i file, senza il tag ma con un'apposita area come dropzone sarebbe esteticamente più bello e funzionale (*user friendly*).
Successivamente si potrebbe mettere questo servizio online e quindi mancherebbe uno spazio fisico dove salvare l'applicativo e farlo raggiungere da chiunque tramite l'ausilio di un browser con accesso ad Internet e con un determinato dominio.
A causa della nostra progettazione, non siamo riusciti a rendere il processing dei video asincrono. Quando si carica un files, infatti, tutte le versioni di esso vengono create (motion vector, zip con ogni frames e video con solo frame I/B/P), invece di creare solo le versioni richieste.

6 Consuntivo

Rispetto alla pianificazione preventiva, nel Gantt consuntivo sono cambiati diversi tempi di diverse attività, questo era dovuto dal fatto che sono stati calcolati male alcuni tempi e l'importanza di alcune attività. Non sono state prese in considerazione ore di margine per eventuali problemi riscontati durante l'implementazione. Nonostante ciò, siamo comunque riusciti a stare nei tempi e a finire il progetto base, ma mancherebbero ancora alcune funzioni.



Nome attività	Durata	Inizio	Fine
Video processing cluster	**10.56 g**	**gio 27.01.22**	**gio 05.05.22**
introduzione	**1.31 g**	**gio 27.01.22**	**gio 03.02.22**
Analisi qdc	2.5 h	gio 27.01.22	gio 27.01.22
Creazione del gantt consuntivo	3 h	gio 27.01.22	gio 27.01.22
Creazione repository git-hub	1 h	gio 27.01.22	gio 27.01.22
use-case	2 h	gio 03.02.22	gio 03.02.22
test-case	2 h	gio 03.02.22	gio 03.02.22
progettazione	**3.13 g**	**gio 03.02.22**	**gio 10.03.22**
Schema della rete	4 h	gio 03.02.22	gio 10.02.22
GUI web	4 h	gio 03.02.22	gio 10.02.22
Database	4 h	gio 03.02.22	gio 10.02.22
Load balancer ricerca	12 h	gio 10.02.22	gio 24.02.22
Server 1 e 2 macchina virtuale	**0.75 g**	**gio 24.02.22**	**gio 10.03.22**
gestione delle sessioni	6 h	gio 24.02.22	gio 10.03.22
ricerca esecuzione in background dei video caricati	6 h	gio 24.02.22	gio 10.03.22
come produrre statistiche sui frame	3 h	gio 24.02.22	gio 24.02.22
come produrre un grafico con la frequenza dei frame	3 h	gio 10.03.22	gio 10.03.22
implementazione	**2.75 g**	**gio 10.03.22**	**gio 31.03.22**
macchina virtuale Server 1 e 2	3 h	gio 10.03.22	gio 10.03.22
configurazione Server 1 e 2	6 h	gio 10.03.22	gio 17.03.22
macchina virtuale database	4 h	gio 10.03.22	gio 17.03.22
configurazione database	0 h	gio 17.03.22	gio 17.03.22
macchina virtuale load balancer	3 h	gio 10.03.22	gio 10.03.22
configurazione load balancer	6 h	gio 10.03.22	gio 17.03.22
struttura web	**1.63 g**	**gio 17.03.22**	**gio 31.03.22**
pagina principale	5 h	gio 17.03.22	gio 24.03.22
pagina per statistiche e download	8 h	gio 24.03.22	gio 31.03.22
test	**2 g**	**gio 31.03.22**	**gio 28.04.22**
funzionamento database	0 h	gio 31.03.22	gio 31.03.22
funzionamento dei 2 server	4 h	gio 31.03.22	gio 07.04.22
funzionamento load balancer	4 h	gio 07.04.22	gio 14.04.22
funzionamento del sistema in generale	4 h	gio 14.04.22	gio 14.04.22
funzionamento del sistema senza un server o con troppo carico	4 h	gio 14.04.22	gio 28.04.22
Documentazione	9.6 g	gio 03.02.22	gio 05.05.22

Figura 27 – GANTT consuntivo

7 Conclusioni

7.1 Sviluppi futuri

Come sviluppo futuro si potrebbe migliorare la parte dei server rendo il load balancer basato sul carico e non sulle connessioni ed anche implementando un sistema per rendere più veloci i processi mettendo le azioni in parallelo (background). Poi si può implementare una grafica più bella e user friendly con la pagina principale che contenga una dropzone per il caricamento del file.

7.2 Considerazioni personali

7.2.1 Samuele Abbà

In conclusione, posso dire che il progetto mi ha insegnato moltissimo, sia a livello personale che a livello lavorativo. Essendo un progetto a gruppi è stato interessante lavorare con i miei compagni, non è stato facile soprattutto all'inizio ma conoscendoci meglio siamo riusciti ad andare d'accordo e a lavorare bene assieme. Essendo un progetto a gruppi è stato più difficile tenere il tutto ordinato, sia lato di codice che lato dei diari. Infatti, durante questo progetto a mio parere non abbiamo tenuto molta cura ai dettagli nei diari e in quello che andavamo a svolgere durante ogni giornata di lavoro, infatti ci siamo concentrati più sul funzionamento rispetto all'estetica del sito e del codice. Nonostante ciò ho migliorato di molto le mie competenze in php, lavorare con dei sistemi come il load balancer (con Nginx) e i due server in cluster (con Memcached) e nella gestione di un progetto con altre persone. Il risultato finale del progetto mi ha abbastanza soddisfatto, anche perché inizialmente non eravamo minimamente in grado di farlo e alla fine abbiamo comunque raggiunto l'obbiettivo. Grazie a questo ho aumentato la mia autostima e la mia voglia di mettermi in gioco in qualcosa che non conosco bene con anche qualcuno che non conosco bene. Come ultimo ma non meno importante a livello di tempistiche posso dire che sono soddisfatto del progetto siccome siamo riusciti a gestire al meglio tutto il tempo, a nostra disposizione.

7.2.2 Damian Campesi

Come commento finale posso dire che sono contento di aver scelto questo progetto, prima d'ora non avevo mai fatto una cosa del genere e grazie a questo progetto ho imparato cose nuove. Ho utilizzato dei servizi molto interessanti, come Memcached che prima non conoscevo però ho anche avuto modo di mettere alla prova le mie abilità in cose che già conoscevo, come l'utilizzo del crontab o la programmazione in PHP. Riguardo al teamwork credo che abbiamo fatto un buon lavoro, siamo riusciti a dividerci in modo efficiente tutte le attività ed in caso di problemi ci siamo sempre confrontati ed aiutati a vicenda. Nonostante ciò durante il progetto abbiamo riscontrato dei problemi che ci hanno portato al rallentamento della progettazione e alla modifica della struttura da sviluppare.

7.2.3 Gioele Cavallo

Personalmente ho trovato il progetto molto interessante e formativo. Ho imparato a conoscere più profondamente il sistema Linux server acquisendo più dimistichezza.
Il progetto lo ho trovato differente da quelli precendentemente fatti perché era in grande parte imparare qualcosa che non sapevo, mi ha portato a crescere professionalmente e a mettermi in gioco. Oltre ad essere differente mi è piaciuto che non fosse tanto di programmazione (infatti abbiamo scritto poco codice), ma fosse molto da sistemista, mettendo all'opera dei server che riuscissero a comunicare tra di loro.
Globalmente sono soddisfatto perché siamo risuciti a rispettare tutte le specifiche (tranne una che per avere un LB basato sul carico bisognava pagare). Se dovessi rifare il progetto userei qualche framework in js che

permette una comunicazione più facile e veloce tra il server ed il client, creando una pagina dinamica con i dati che si visualizzano mentre il server continua ad elaborare il file in background.
Al livello del team sono felice che siamo riusciti a dividerci i compiti bene. Quando qualcuno aveva bisogno di una mano c'era sempre chi era disposto a smettere di svolgere il proprio compito per aiutare, e questa cosa mi rende molto felice perché ho capito di essere stato affianco ad un buon team composto da persone competenti ed altruiste.
Inoltre le idee di tutti era messe in discussione in modo costruttivo senza ridicolizzarle, anche se molto diverse da ciò che si pensava fosse la strada migliore, erano accolte senza pregiudizzi e ciò rendeneva l'ambiente lavorativo molto leggero rendendo le persone del team propense a collaborare perché si sentivano ascoltate.

8 Bibliografia

8.1 Sitografia

https://www.php.net/manual/en/index.php , *Sito di Php*, abbiamo consultato questo sito durante tutto il progetto.

https://www.w3schools.com/php/default.asp , *Sito W3schools per php*, abbiamo consultato questo sito durante tutto il progetto.

https://help.clouding.io/hc/en-us/articles/360019908839-How-to-set-up-Nginx-load-balancing-server-on-Ubuntu-20-04, *How to set up Nginx load balancing server on Ubuntu 20.04,* 03.02.2022.

https://turbolab.it/webserver-2954/guida-definitiva-nginx-php-8-ubuntu-centos-come-attivare-installare-configurare-php-fpm-nginx-linux-1957, *come attivare, installare, configurare PHP-FPM con Nginx su Linux,* 10.02.2022.

https://www.digitalocean.com/community/tutorials/how-to-share-php-sessions-on-multiple-memcached-servers-on-ubuntu-14-04, *How To Share PHP Sessions on Multiple Memcached Servers on Ubuntu 14.04,* 17.02.2022.

https://cloud.netapp.com/blog/azure-anf-blg-linux-nfs-server-how-to-set-up-server-and-client, *Linux NFS Server: How to Set Up Server and Client,* 24.02.2022.

https://www.chartjs.org/docs/latest/, *Chart.js*, 24.02.2022

8.2 Glossario

Cluster: Un cluster di computer è un gruppo di computer collegati che funzionano insieme in modo da essere considerati come un unico computer.

Frame: Singola immagine visualizzabile.

Client: Indica genericamente un qualunque componente software, presente tipicamente su una macchina host, che accede ai servizi o alle risorse di un'altra componente.

Server: Indica genericamente un componente o sottosistema informatico di elaborazione e gestione del traffico di informazioni che fornisce, a livello logico e fisico, un qualunque tipo di servizio ad altre componenti.

Host: Un host di rete è un computer o un altro dispositivo connesso a una rete di computer

Load balancer: È una tecnica informatica che consiste nel distribuire il carico di uno specifico servizio, ad esempio la fornitura di un sito web.

Memcached: È un programma che mette in cache i dati richiesti permettendone la condivisione tramite RAM tra diversi server.

Motion Vector: Nella compressione video è un elemento chiave nel processo di stima del movimento. Viene usato per rappresentare il movimento di un macro blocco in un frame, in base alla posizione del medesimo in un altro frame (chiamato anche frame di riferimento).

Macro blocco: Unità di misura utilizzata per separare le immagini in una griglia, dove ogni cella è un *macro blocco (macroblock)*.

Variabile globale: Le variabili globali sono le variabili definite al di fuori delle funzioni. Le quali sono accessibili da qualsiasi parte dello script, all’interno ed all’esterno delle funzioni.

8.3 Indice delle immagini

9 Allegati

- Mandato e/o QdC
- Prodotto
- Diari di lavoro
- Codici sorgente/documentazione macchine virtuali
- Istruzioni di installazione del prodotto
- Use-case
- Gantt preventivo
- Gantt consultivo