

Projet HagiMule

Mohammed Amine Benkia Marwane Karaoui

January 2025

1 Introduction et objectifs du projet

1.1 Présentation du projet

HagiMule est une infrastructure distribuée conçue pour faciliter le téléchargement parallèle de fichiers volumineux en s'appuyant sur plusieurs sources simultanément.

Objectifs :

- **Optimisation des téléchargements** : Exploiter le parallélisme au niveau du CPU et du réseau pour accélérer le processus.
- **Gestion des pannes** : Assurer la continuité des téléchargements grâce à une gestion efficace des échecs et des relances.
- **Amélioration des performances** : Réduire la taille des données à télécharger en appliquant une compression sur les fragments de fichiers.

1.2 Architecture principale

L'infrastructure est composée de trois composants principaux :

- **Annuaire** : Gère les fichiers des clients connectés via RMI.
- **Daemon** : Répond aux demandes de fragments via des sockets TCP.
- **Downloader** : Télécharge les fragments et les assemble pour reconstituer le fichier.

L'architecture est bien mise en place, et les tests montrent que tous les composants fonctionnent correctement ensemble. Les captures d'écran ci-dessous illustrent le succès de l'implémentation.

```
PS C:\Users\moskia\Downloads\HagiMule-main\HagiMule-main\HagiMule-main> java AnnuaireImpl
get server is running
```

Figure 1: Lancement de l'annuaire (AnnuaireImpl) avec succès.

```
PS C:\Users\moskia\Downloads\HagiMule-main\HagiMule-main\HagiMule-main> java Client.Client farok 8080 0
jeune en écoute sur le port : 8080
```

Figure 2: Lancement du client (Daemon) avec succès.

2 Réalisations techniques (1/2)

2.1 Connexion et communication, Fragmentation des fichiers

La communication entre Daemon et Downloader est assurée par des sockets TCP, permettant le transfert des fragments de fichiers de manière fiable et rapide. Ces sockets garantissent ainsi un transfert fluide des données. Par ailleurs, la gestion des fichiers dans l'annuaire est facilitée par l'utilisation de RMI (Remote Method Invocation), qui permet de gérer à distance les informations sur les fichiers partagés par les clients. Cela garantit que chaque client puisse accéder à l'annuaire pour rechercher et mettre à jour les fichiers disponibles.

Les fichiers lors du téléchargement sont divisés en fragments égaux, à l'exception du dernier fragment qui peut être ajusté pour correspondre à la taille exacte du fichier. Cette approche permet une gestion efficace du téléchargement et de l'assemblage des fichiers.

```
PS C:\Users\moskia\Downloads\HagiMule-main\HagiMule-main\HagiMule-main> java Client.Client abas 8082 1 trip.mp4
les sources disponible : [farok:8080, nawras:8081]
Téléchargement du fichier : trip.mp4
fragment size :228649811
fragment size :228649812
Total file size telecharge: 457299623
Temps total d'execution : 4153 ms
```

Figure 3: Téléchargement d'un fichier par fragmentation.

2.2 Gestion des clients

Le système gère deux types de clients :

- **Clients non enregistrés** : Ces clients téléchargent des fichiers sans être présents dans l'annuaire. Dans ce cas, l'annuaire n'est pas mis à jour.
- **Clients enregistrés** : Ces clients sont déjà inscrits dans l'annuaire. Lorsqu'ils ajoutent des fichiers, l'annuaire est automatiquement mis à jour pour refléter ces changements.

Chaque client possède un dossier local pour stocker ses fichiers téléchargés et partagés. Lorsqu'un client enregistré effectue une modification, l'annuaire est mis à jour dynamiquement pour garantir que les informations des fichiers partagés sont toujours à jour.

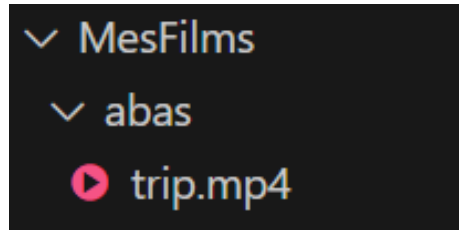


Figure 4: Illustration de la gestion dynamique des clients et de l'annuaire.

3 Réalisations techniques (2/2)

3.1 Relance des fragments perdus

Lorsque le téléchargement d'un fragment échoue, le système tente automatiquement de le récupérer depuis une autre source. Cependant, pour éviter les boucles infinies ou une surcharge du système, la relance est limitée à trois tentatives par fragment. Cela permet de garantir que les téléchargements se poursuivent de manière efficace sans provoquer de ralentissements importants.

```
PS C:\Users\moskia\Downloads\HagiMule-main\HagiMule-main\HagiMule-main> java Client.Client abas 8082 1 trip.mp4 c
les sources disponible : [farok:8080, nawras:8081]
Téléchargement du fichier : trip.mp4
fragment size :228649811
fragment size :228649812
Erreur lors du téléchargement du fragment 1 depuis nawras:8081 : Connection reset
Relancement du téléchargement du fragment 1 depuis farok:8080 : Connection reset
fragment size :228649812
Total file size telecharge: 456294709
Temps total d'execution : 15474 ms
```

Figure 5: Relancement du téléchargement du fragment perdu.

3.2 Compression/Décompression

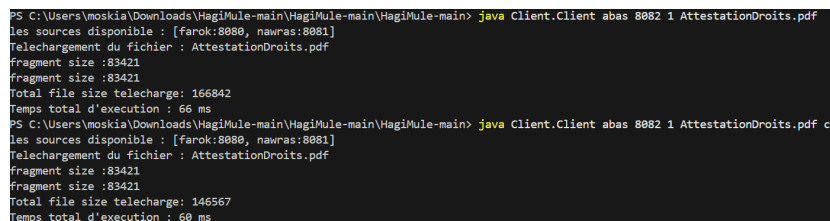
Les fragments sont compressés avant leur envoi afin d'optimiser la bande passante, réduisant ainsi la quantité de données transférées. Une fois reçus, les fragments sont décompressés pour reconstituer le fichier complet. La compression est particulièrement avantageuse pour les fichiers texte, car elle réduit leur taille de manière significative. Cependant, pour les petits fichiers, la compression peut introduire une latence supplémentaire, ce qui peut ralentir le processus global de téléchargement.

```
PS C:\Users\moskia\Downloads\HagiMule-main\HagiMule-main\HagiMule-main> java Client.Client abas 8082 1 trip.mp4 c
les sources disponible : [farok:8080, nawras:8081]
Téléchargement du fichier : trip.mp4
fragment size :228649812
fragment size :228649811
Total file size telecharge: 456294709
Temps total d'execution : 12822 ms
```

Figure 6: Logs montrant les tailles avant et après compression côté Daemon.

3.3 Suivi des performances

Le système calcule le temps total de téléchargement en utilisant la méthode ‘System.nanoTime()’ pour obtenir une mesure précise du temps écoulé. Cela permet de suivre les performances du téléchargement et d’identifier d’éventuels goulots d’étranglement. Les informations sur le temps total sont essentielles pour évaluer l’efficacité du système et apporter des améliorations si nécessaire.



```
PS C:\Users\moskia\Downloads\HagiMule-main\HagiMule-main\HagiMule-main> java Client.Client abas 8082 1 AttestationDroits.pdf
les sources disponible : [farok:8080, nawras:8081]
téléchargement du fichier : AttestationDroits.pdf
fragment size :83421
fragment size :83421
Total file size telecharge: 166842
Temps total d'execution : 86 ms
PS C:\Users\moskia\Downloads\HagiMule-main\HagiMule-main\HagiMule-main> java Client.Client abas 8082 1 AttestationDroits.pdf c
les sources disponible : [farok:8080, nawras:8081]
téléchargement du fichier : AttestationDroits.pdf
fragment size :83421
fragment size :83421
Total file size telecharge: 146567
Temps total d'execution : 60 ms
```

Figure 7: La durée totale du téléchargement sans/avec compression.

4 Limitations et points non implémentés

4.1 Relance avec fragmentation

Actuellement, lorsque les relances échouent, il n’est pas possible de fragmenter davantage le fragment problématique. Cette limitation empêche une nouvelle tentative de téléchargement en utilisant une autre méthode de fragmentation pour récupérer le fragment manquant.

4.2 Reprise partielle des fragments

Si un téléchargement échoue partiellement, les données déjà reçues ne sont pas conservées. En d’autres termes, le système redémarre le téléchargement du fragment depuis le début, sans profiter des données déjà téléchargées.

4.3 Gestion des sources multiples

Le système n’implémente pas de limitation sur l’utilisation simultanée d’une même source par plusieurs clients. Cela peut entraîner une surcharge de la source et affecter les performances globales. De plus, il n’y a pas de mécanisme de priorisation ou de file d’attente pour gérer la demande, ce qui pourrait améliorer la répartition des ressources et éviter la congestion.

4.4 Compression dynamique

La compression est appliquée à tous les fichiers, même si elle est inefficace, comme dans le cas des fichiers déjà compressés. Cela peut introduire une latence

inutile et augmenter le temps de traitement, notamment pour les fichiers qui ne bénéficient pas de la compression.

4.5 Sécurité

Le système ne prévoit pas de chiffrement pour protéger les données échangées entre les clients et les serveurs. Cela laisse les données vulnérables aux attaques et aux interceptions pendant le transfert.

5 Conclusion

5.1 Résumé des réalisations

Les principales fonctionnalités du projet ont été implémentées avec succès, telles que le téléchargement parallèle, la compression/décompression des fragments et la gestion des pannes. L'infrastructure est maintenant opérationnelle et peut facilement être étendue pour répondre à de nouveaux besoins.

5.2 Réflexion sur les limitations

Cependant, certaines fonctionnalités non implémentées restent cruciales pour une utilisation en production. Cela inclut la sécurité des données, la gestion avancée des sources et la reprise partielle des fragments, qui seraient nécessaires pour améliorer la robustesse et la performance du système.

5.3 Perspectives

Dans l'avenir, il serait pertinent d'intégrer les fonctionnalités suivantes :

- Sauvegarde partielle des fragments échoués pour éviter de redémarrer un téléchargement complet.
- Limitation des connexions simultanées à une même source pour mieux gérer les ressources.
- Compression dynamique qui s'adapte en fonction du type de fichier pour optimiser les performances.