

Etude de la scalabilité des serveurs de visio-conférence

Roméo La Spina

Stage encadré par Gilles Muller (INRIA)
Tuteur : Alain Tchana
ENS de Lyon

1er juin - 31 juillet 2020



Lignes directrices

- 1 Introduction
- 2 Présentation de la plateforme Jitsi Meet
 - Les différents modes de distribution des flux
 - Fonctionnement de la plateforme
- 3 Notre installation en pratique
 - Côté serveur
 - Côté client
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

Introduction

- Contexte sanitaire → visio-conférences !
- Scalabilité : capacité d'un système à conserver ses performances lors de son utilisation à très grande échelle
- Objectif : améliorer l'implantation de ces systèmes afin qu'ils utilisent efficacement les ressources

Présentation du stage

- Utilisation de la plateforme Jitsi Meet, open source
- Tests à grande échelle + monitoring de l'utilisation des ressources côté client et côté serveur
- But : connaître les principaux obstacles à la scalabilité de Jitsi Meet

Lignes directrices

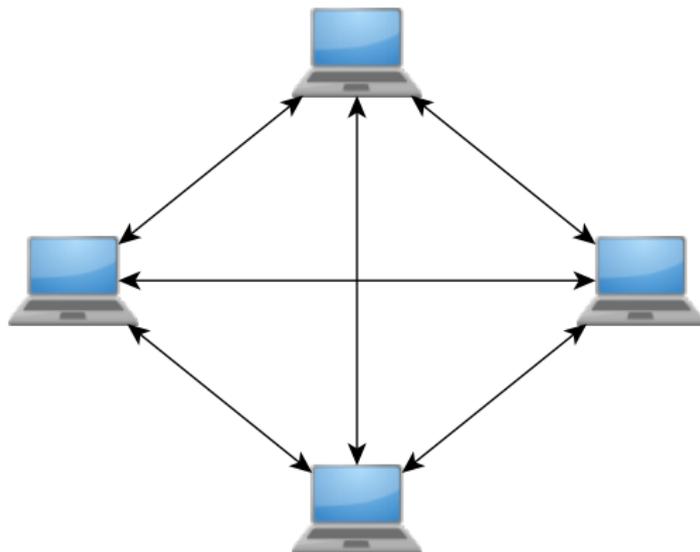
- 1 Introduction
- 2 **Présentation de la plateforme Jitsi Meet**
 - Les différents modes de distribution des flux
 - Fonctionnement de la plateforme
- 3 Notre installation en pratique
 - Côté serveur
 - Côté client
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

Lignes directrices

- 1 Introduction
- 2 **Présentation de la plateforme Jitsi Meet**
 - **Les différents modes de distribution des flux**
 - Fonctionnement de la plateforme
- 3 Notre installation en pratique
 - Côté serveur
 - Côté client
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

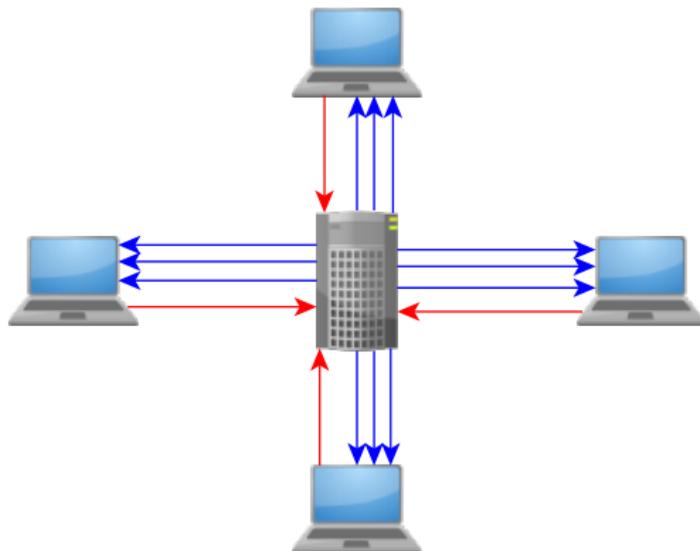
Mesh/Peer to Peer

Pas de serveur, bien pour de petites sessions.



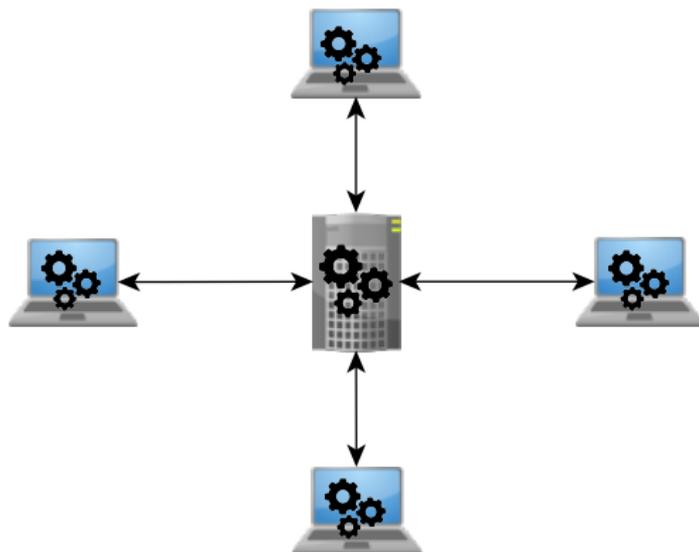
SFU (Selective Forwarding Unit)

Le plus utilisé, limite la bande passante sortante côté client.



MCU (Multipoint Control Unit)

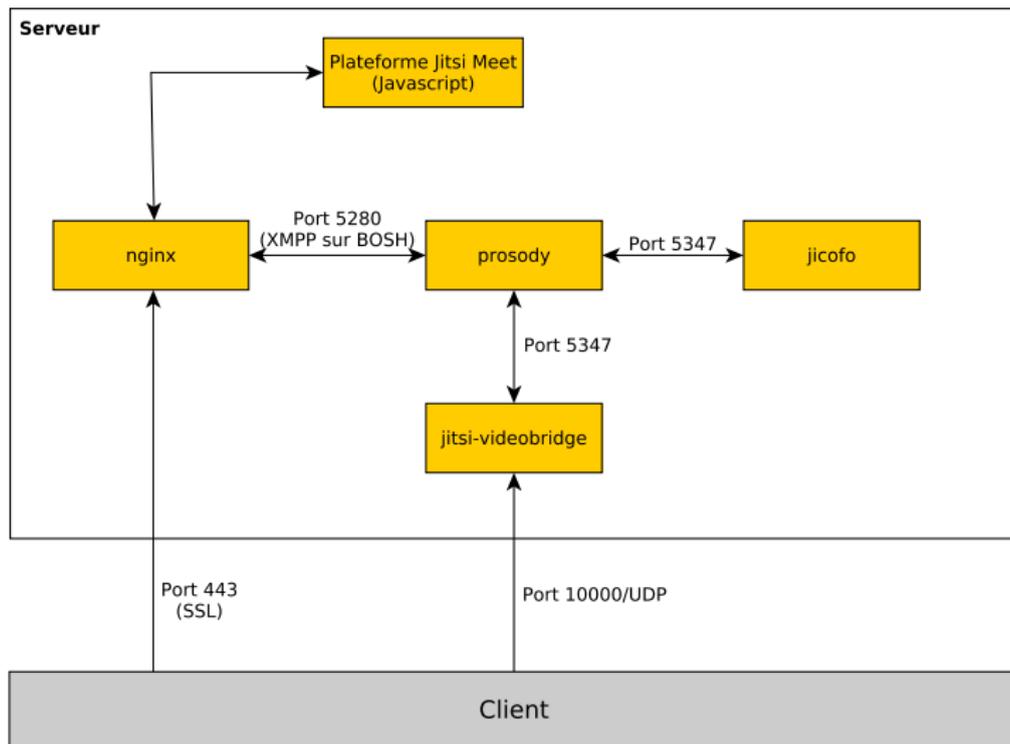
Plus de calculs mais moins de bande passante. De plus en plus utilisé.



Lignes directrices

- 1 Introduction
- 2 **Présentation de la plateforme Jitsi Meet**
 - Les différents modes de distribution des flux
 - **Fonctionnement de la plateforme**
- 3 Notre installation en pratique
 - Côté serveur
 - Côté client
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

Un petit schéma vaut mieux qu'un long discours...



Lignes directrices

- 1 Introduction
- 2 Présentation de la plateforme Jitsi Meet
 - Les différents modes de distribution des flux
 - Fonctionnement de la plateforme
- 3 Notre installation en pratique**
 - Côté serveur
 - Côté client
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

Lignes directrices

- 1 Introduction
- 2 Présentation de la plateforme Jitsi Meet
 - Les différents modes de distribution des flux
 - Fonctionnement de la plateforme
- 3 Notre installation en pratique
 - **Côté serveur**
 - Côté client
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

Présentation de l'installation

- Serveur situé à l'IRISA configuré par SSH
- Héberge les composants de la plateforme (nginx, prosody,...)
- `netdata` : deuxième plateforme pour collecter des métriques

Collecte des métriques

- Statistiques fournies par `jitsi-videobridge` + extension Jitsi pour `netdata`
- Pour visualiser les métriques : page Javascript (graphiques avec le nombre de participants en abscisse)

Graphes suivis côté serveur

- L'utilisation du CPU liée à l'utilisateur (*user*) et de la RAM
- Le trafic réseau entrant et sortant
- Le nombre d'instructions par seconde
- Le nombre d'accès cache et de *cache misses*

Lignes directrices

- 1 Introduction
- 2 Présentation de la plateforme Jitsi Meet
 - Les différents modes de distribution des flux
 - Fonctionnement de la plateforme
- 3 Notre installation en pratique**
 - Côté serveur
 - Côté client**
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

Présentation de l'installation

- Java + Selenium + ChromeDriver pour simuler des connexions
- Flux vidéo factice
- Mesure de la qualité audio : enregistrement de flux sortants/entrants + analyse avec MOSNet
- Deux machines virtuelles exécutent le programme Java pour avoir une bande passante suffisante (environ 20 connexions)
- `netdata` sur les machines virtuelles

Scénarios d'expérimentation

Trois scénarios :

- Scénario 1 : tous les participants sont dans la même session
- Scénario 2 : les participants sont répartis dans différentes sessions de 2 participants (gérées en mode P2P).
- Scénario 3 : les participants sont répartis dans différentes sessions de 4 participants (gérées en mode SFU)

Lignes directrices

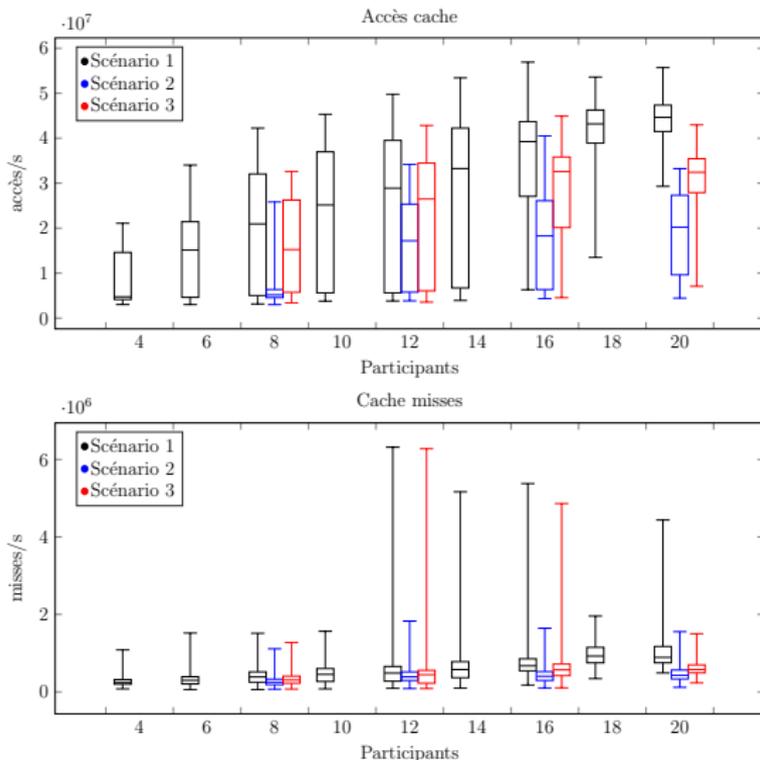
- 1 Introduction
- 2 Présentation de la plateforme Jitsi Meet
 - Les différents modes de distribution des flux
 - Fonctionnement de la plateforme
- 3 Notre installation en pratique
 - Côté serveur
 - Côté client
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

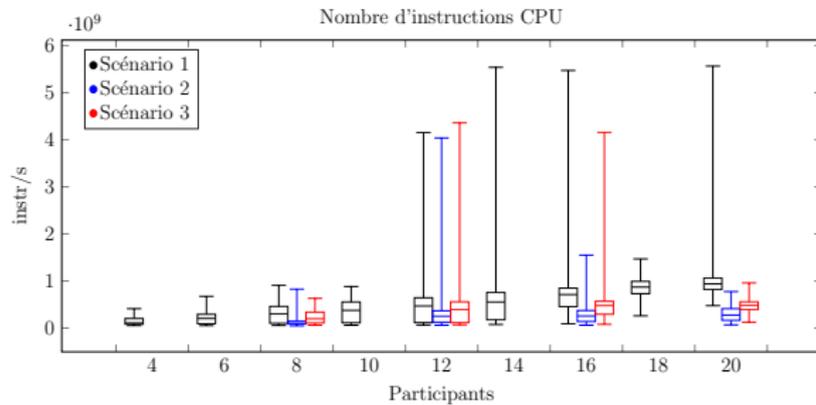
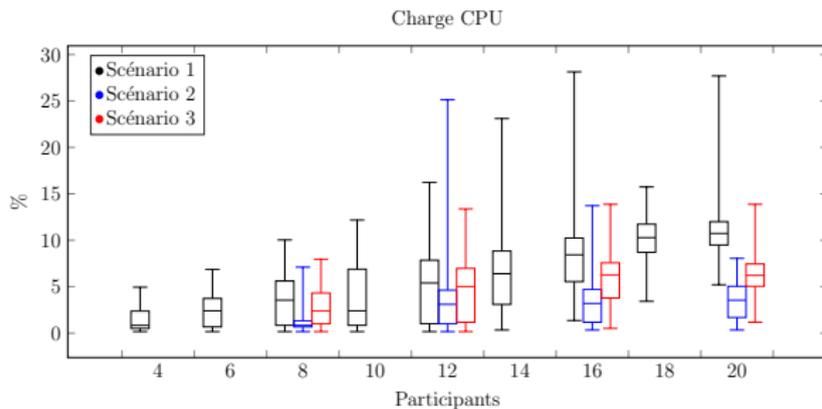
Principales tendances

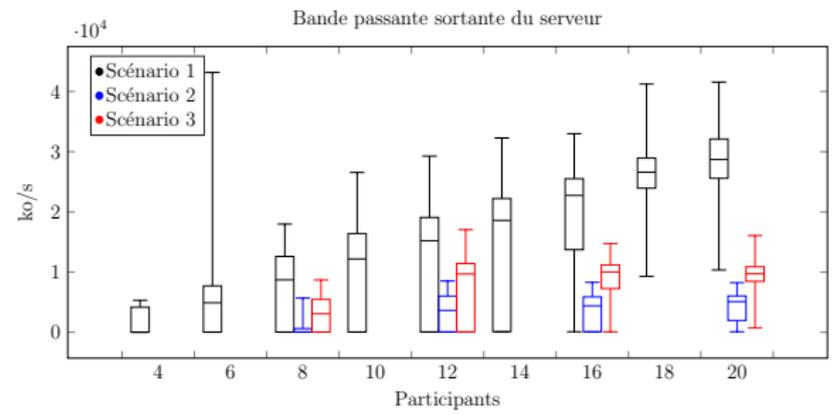
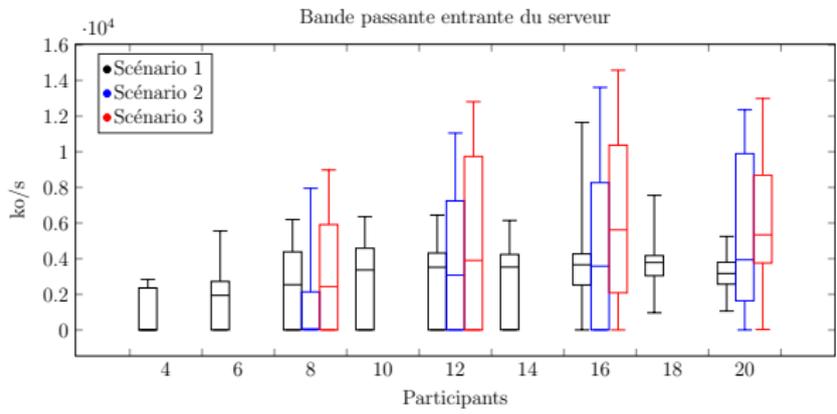
- Gourmand côté client ! (70% d'utilisation CPU pour 6 clients sur une VM)
- Au niveau du serveur, très économe en RAM : presque pas de différence entre 4 et 20 participants !
→ aucun flux stocké, seulement du transit
- Tendence globale assez intuitive : Scénario 1 > Scénario 3 > Scénario 2
- Pratiquement aucune dégradation audio à ce niveau de charge du serveur

Graphes

1 à 2% de cache misses : très bon taux







Lignes directrices

- 1 Introduction
- 2 Présentation de la plateforme Jitsi Meet
 - Les différents modes de distribution des flux
 - Fonctionnement de la plateforme
- 3 Notre installation en pratique
 - Côté serveur
 - Côté client
- 4 Résultats obtenus
- 5 Conclusion et améliorations possibles

Conclusion

- Pour améliorer la scalabilité de Jitsi Meet, il faut réduire l'usage de la bande passante... → MCU serait un choix judicieux, car on a de la marge en termes de puissance de calcul
- Problématique qui s'inscrit dans un contexte historique de croissance exponentielle du volume de données échangées

Autres améliorations possibles

- Tests à plus grande échelle pour la qualité audio, de préférence avec le scénario 3 pour ne pas saturer la bande passante
- Choisir le codec avec le meilleur rapport bande passante/qualité
→ Réalisation d'un benchmark de différents échantillons