Un make parallèle avec Go RPC

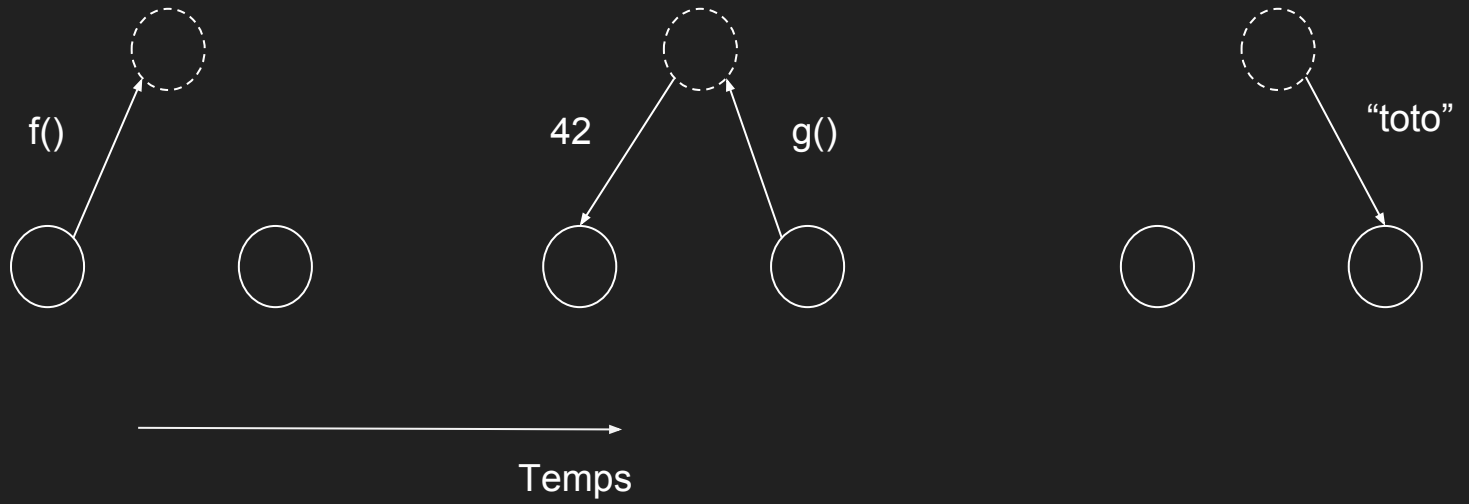
Ludovic CARRE - Maxime DELOCHE - Vincent LEFOULON - Omar SANHAJI

Le langage Go

- Open-source (Google, 2009)
- Compilé, fortement et statiquement typé
- Projets utilisant Go : Docker, Kubernetes, Dropbox...
- Orienté programmation concurrente (threads, mutexes, etc.)

Le protocole RPC

- **R**emote **P**rocedure **C**all
- Architecture client-serveur



- Pour être contacté un noeud doit faire tourner un serveur

Configuration et déploiement

- Déploiement : TakTuk (lancement parallèle commandes sur des noeuds distants)
- Connexion à un site de grid5000, réservation de noeuds

- Configuration : scripts shell (récupère hostname des noeuds avec $OAR_FILE_NODES)
- Clône d'un dépôt Git & compilation du projet sur chaque noeud ou copie du dossier sur la mémoire partagée vers /tmp/ pour éviter d'installer git.
- Beaucoup de noeuds inutilisables : 70 noeuds réservés, 15 répondent

Automatisation des mesures

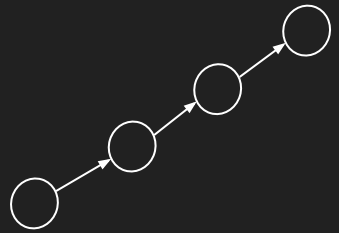
1. Configuration des machines : dépendances, sources, compilation
2. Récupération des caractéristiques des noeuds
3. master = nodes[0]; slaves = nodes[1:]
4. Pour MIN_SLAVES <= i <= MAX_SLAVES, faire N fois :
    a. Démarrer le maître
    b. Démarrer les esclaves
    c. Récupérer les temps d’exécution de chaque machine en JSON
5. Représenter graphiquement les résultats avec gnuplot

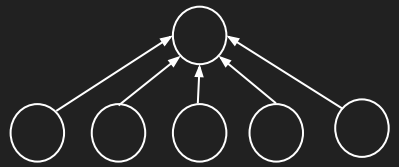
Intervalle de confiance

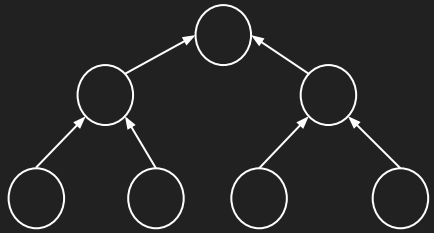
1. Mesures = x_1, x_2, …, x_n
2. Moyenne m, écart-type σ (empiriques)
3. Confiance (loi de Student) α = 95% => $Z_{\alpha/2} = Z_{0.475} = 2.92$
4. Erreur : $e = Z_{0.475} \times \sigma / \sqrt{n}$
5. Intervalle de confiance = [m - e ; m + e]

Environnement de test

- Makefiles créés avec commandes sleep (simule des traitements longs)
- Plusieurs types de Makefile :

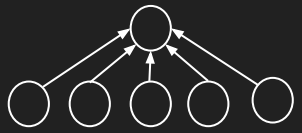




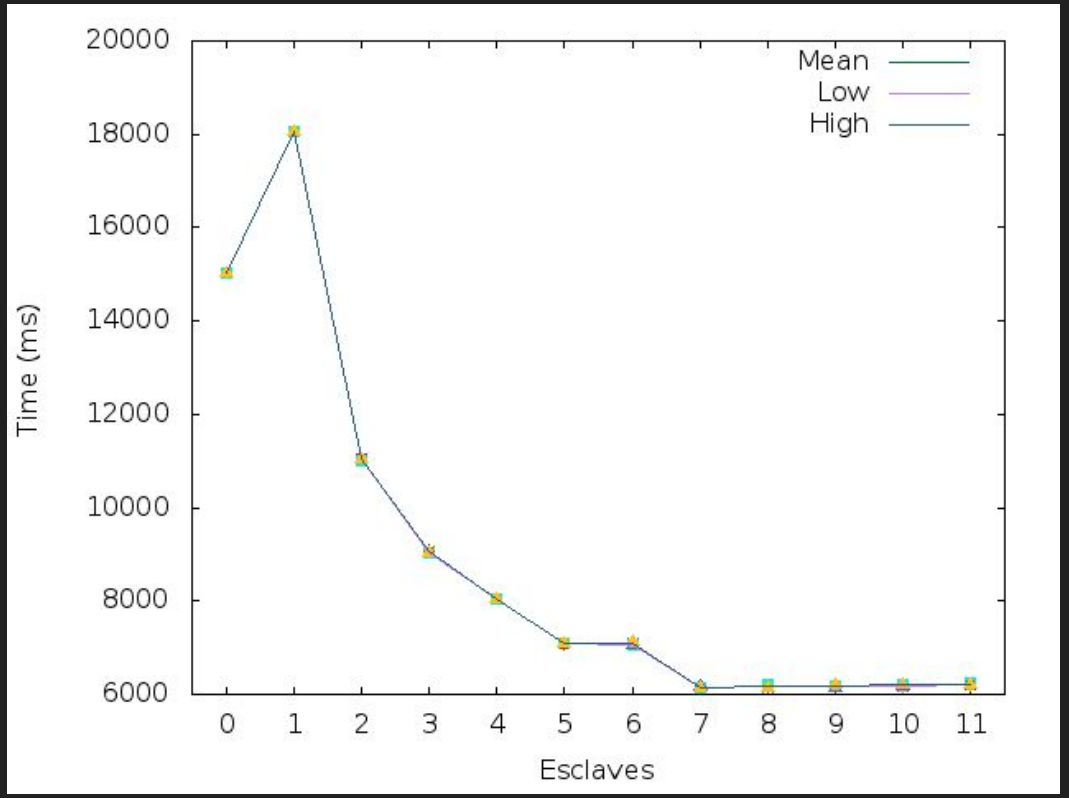


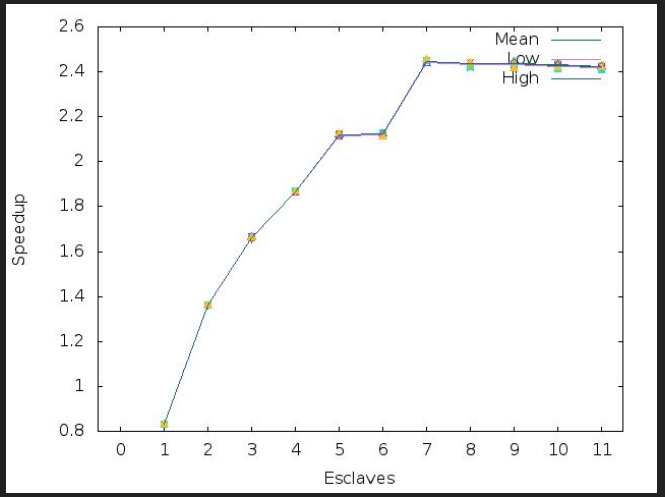
- Machines de test :
  - Intel Xeon 4 coeurs 2.53GHz
  - SMP Debian 3.16.43-2+deb8u5
  - 16Go RAM
- 3 répétitions, tracé de la moyenne des mesures de temps

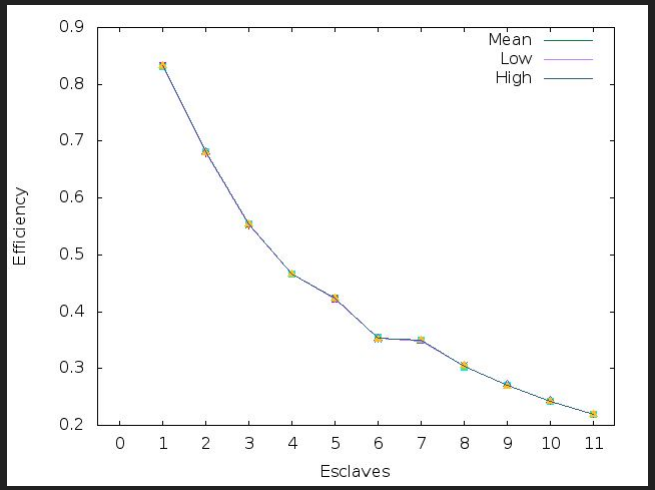
Mesures de performance



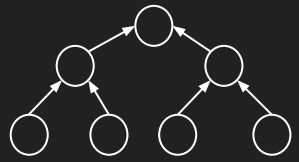
- 30 répétitions, tracé de la moyenne des mesures



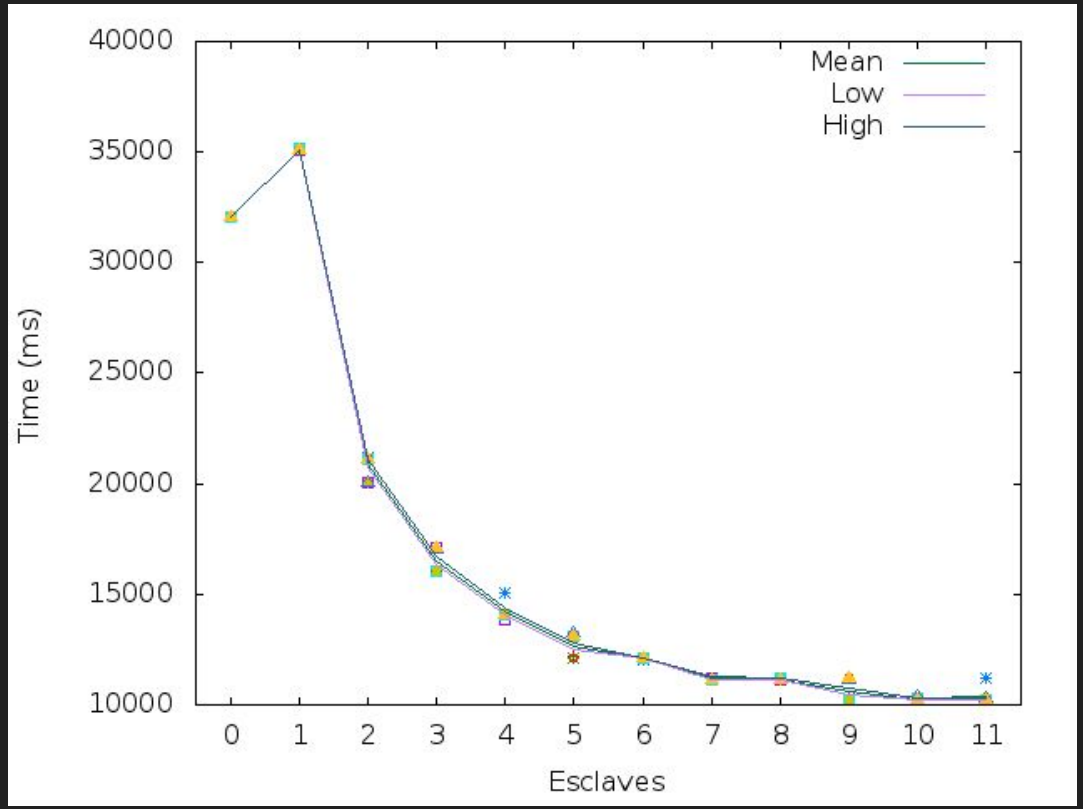


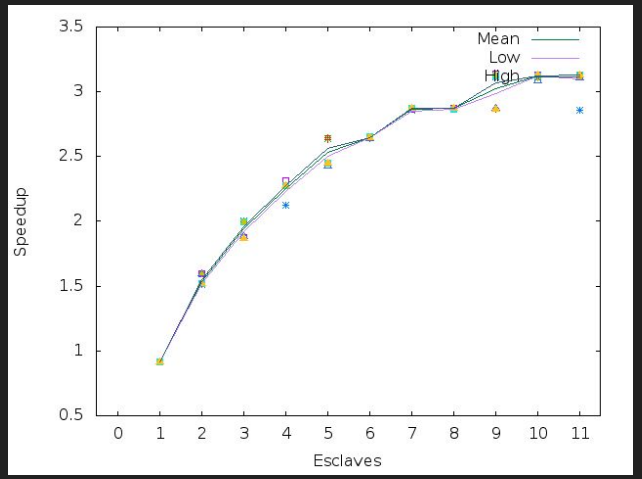


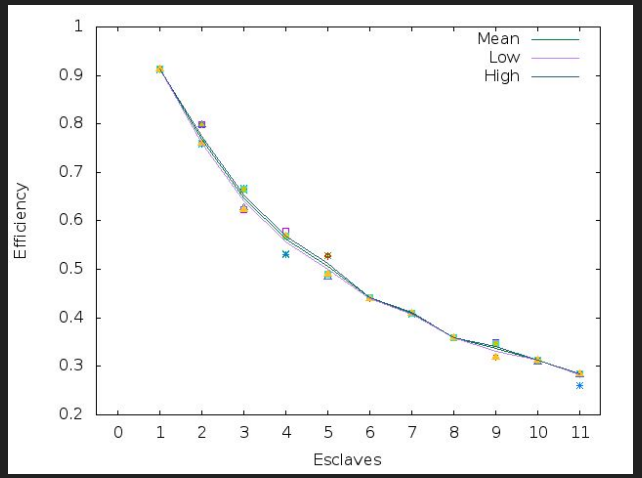
Mesures de performance



- 30 répétitions, tracé de la moyenne des mesures







Retour critique

- Go très pratique pour gérer la concurrence (création de threads, exclusion mutuelle, etc.)
- Go très bien documenté
- Concurrence difficile à déboguer
- RPC : communication bi-directionnelle compliquée (besoin de démarrer un serveur à chaque fois) -> combiner RPC et PubSub