Consommation adaptative par négociation continue

Ellie Beauprez, Anne-Cécile Caron, Maxime Morge, Jean-Christophe Routier

Équipe SMAC, CRIStAL, Univ. Lille

5 Juillet 2023

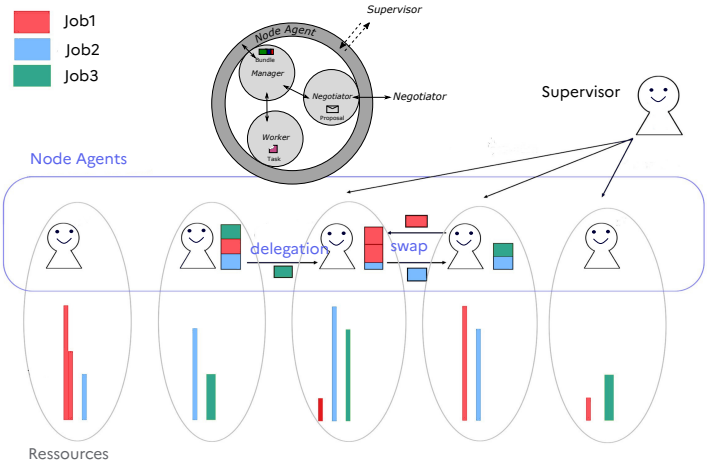
Consommation adaptative par négociation continue

Université de Lille

- **Objectif.**
  Optimisation du réordonnancement de tâches pour des jobs multiples qui doivent être exécutés le plus tôt possible
- **Application pratique.**
  Traitement de données massives avec le patron de conception MapReduce
- **Proposition.**
  Une architecture multi-agent modulaire qui permet la concurrence de la négociation et de la consommation des tâches
- **Contributions.**
  - Formalisation de la consommation de tâche et des opérations de réallocation
  - **3 agents composants:**
    - **un *worker* qui exécute les tâches**
    - **un *negotiator* qui négocie les réallocations**
    - **un *manager* qui coordonne localement ces opérations**
  - Expériences approfondies

Équilibrage de charge de jobs concurrents

Chaque agent-nœud représente un nœud de calcul qui gère son propre lot de tâches



Plan

Université de Lille

Répondre aux exigeances de l'application pratique

Université de Lille

- Limites des solutions centralisées [Jiang, 2016] :
  1. impossibilité de traiter des problèmes à grande échelle
  2. faible réactivité aux changements
- Beaucoup de travaux sont basés sur des enchères (e.g. Koenig et al. [2006], Choi et al. [2009]) mais :
  - nous considérons des **agents coopératifs** visant à minimiser le *flowtime* moyen, i.e. la durée moyenne entre la date de libération des jobs et leur date de réalisation
  - nous préférons un **protocole bilatéral** qui permet plusieurs négociations concurrentes
- Contrairement à [Chen et al., 2016], nos agents peuvent avoir une connaissance imparfaite de l'environnement
- Contrairement à [Creech et al., 2021], notre solution ne requiert aucun modèle préalable sur les données ou l'environnement, ni phase d'exploration

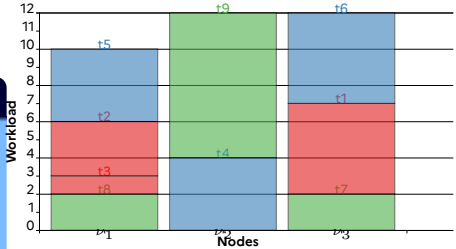
Formalisation

Université de Lille

Définition : STAP

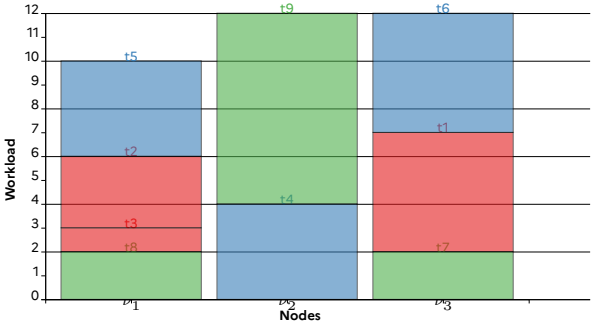
Un **problème d'allocation de tâches situées** (STAP - *Situated Task Allocation Problem*) est un quadruplet $\text{STAP} = \langle \mathcal{D}, \mathcal{T}, \mathcal{J}, c \rangle$ où :

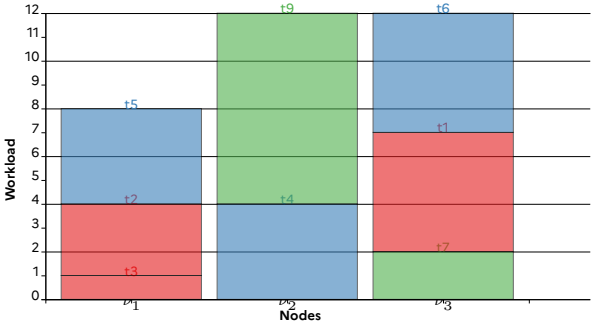
- $\mathcal{D} = \langle \mathcal{N}, \mathcal{E}, \mathcal{R} \rangle$ est un système distribué avec m nœuds ;
- $\mathcal{T} = \{\tau_1, \dots, \tau_n\}$ est un ensemble de n tâches ;
- $\mathcal{J} = \{\mathsf{J}_1, \dots, \mathsf{J}_\ell\}$ est un partionnement des tâches en ℓ jobs ;
- $c : \mathcal{T} \times \mathcal{N} \mapsto \mathbb{R}_+^*$ est la fonction de coût.



- ***Flowtime* moyen** = durée moyenne de réalisation des jobs pour une allocation
- $\mathsf{C}_{\mathsf{J}_{rouge}}(\overrightarrow{\mathsf{A}_t}) = 7$
- $\mathsf{C}_{\mathsf{J}_{bleu}}(\overrightarrow{\mathsf{A}_t}) = 12$
- $\mathsf{C}_{\mathsf{J}_{vert}}(\overrightarrow{\mathsf{A}_t}) = 12$
- Soit $\mathsf{C}_{mean}(\overrightarrow{\mathsf{A}_t}) \simeq 10,33$

Opérations : consommation





Le nœud ν_1 consomme la tâche τ_8

Opérations : consommation

Université de Lille

Définition : consommation

La **consommation à l'instant t par le nœud** ν_i conduit à l'allocation $\overrightarrow{\mathsf{A}_t}' = \lambda(\nu_i, \overrightarrow{\mathsf{B}_{i,t}})$ pour le problème $\mathsf{STAP}' = \langle \mathcal{D}, \mathcal{T}', \mathcal{J}', c \rangle$ où $\mathcal{T}' = \mathcal{T} \setminus \{\min_{\prec_i} \mathsf{B}_{i,t}\}$ et :

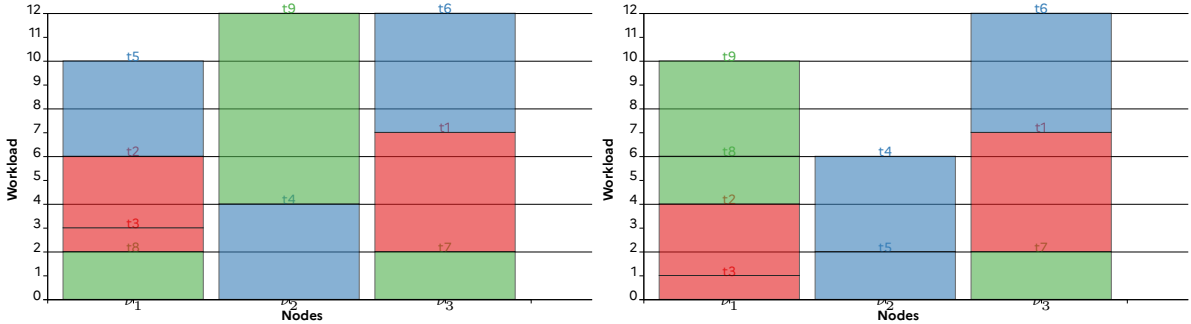
$$\mathcal{J}' = \begin{cases} \mathcal{J} \setminus \{\mathsf{job}(\min_{\prec_i} \mathsf{B}_{i,t})\} & \text{si } \mathsf{job}(\min_{\prec_i} \mathsf{B}_{i,t}) = \{\min_{\prec_i} \mathsf{B}_{i,t}\} \\ \mathcal{J} & \text{sinon} \end{cases}$$

Dans ce dernier cas :

$$\forall \mathsf{J}_j \in \mathcal{J}\ \exists \mathsf{J}'_j \in \mathcal{J}' \text{ s.t. } \mathsf{J}'_j = \begin{cases} \mathsf{J}_j \setminus \{\min_{\prec_i} \mathsf{B}_{i,t}\} \text{ si } \mathsf{job}(\min_{\prec_i} \mathsf{B}_{i,t}) = \mathsf{J}_j \\ \mathsf{J}_j \text{ sinon} \end{cases}$$

$$\text{et } \overrightarrow{\mathsf{A}_t}' = (\overrightarrow{\mathsf{B}_{1,t}}', ..., \overrightarrow{\mathsf{B}_{m,t}}') \text{ avec } \overrightarrow{\mathsf{B}_{j,t}}' = \begin{cases} \overrightarrow{\mathsf{B}_{i,t} \ominus \min_{\prec_i} \mathsf{B}_{i,t}} & \text{si } j = i \\ \overrightarrow{\mathsf{B}_{j,t}} & \text{sinon} \end{cases}$$

Opérations : réallocation bilatérale



Les nœuds ν_1 et ν_2 échangent les tâches τ_5 et τ_9

Opérations : réallocation bilatérale

Université de Lille

Définition : réallocation bilatérale

La **réallocation bilatérale** de la liste non vide de tâches T_1 affectées au proposant ν_i en échange de la liste de tâches T_2 affectées au répondant ν_j in $\overrightarrow{\mathsf{A}_t}$ conduit à l'allocation $\gamma(\mathrm{T}_1, \mathrm{T}_2, \nu_i, \nu_j, \overrightarrow{\mathsf{A}_t})$ avec m lots de tâches tels que :

$$\gamma(\mathrm{T}_1, \mathrm{T}_2, \nu_i, \nu_j, \overrightarrow{\mathsf{B}_{k,t}}) = \begin{cases} \overrightarrow{\mathsf{B}_{i,t} \ominus \mathrm{T}_1 \oplus \mathrm{T}_2} & \text{si } k = i, \\ \overrightarrow{\mathsf{B}_{j,t} \ominus \mathrm{T}_2 \oplus \mathrm{T}_1} & \text{si } k = j, \\ \overrightarrow{\mathsf{B}_{k,t}} & \text{sinon} \end{cases}$$

Si T_2 est vide, la réallocation est une délégation. Sinon, c'est un échange.

Stratégie de négociation en deux phases

Université de Lille

détaillée dans Beauprez, Caron, et al. [2022a]

Les agents disposent de :

- Un modèle des pairs construit à partir de sa base de croyances
- Une stratégie d'offre
- Une règle d'acceptabilité
- Une stratégie de contre-offre

Phase 1

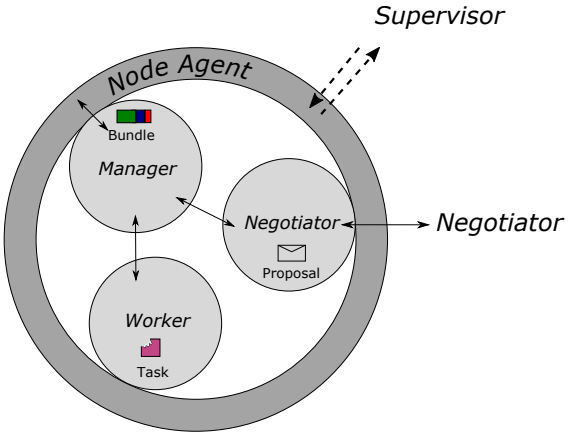
Stratégie d'offre conservatrice où le donneur propose des délégations socialement rationnelles

Phase 2

Stratégie d'offre libérale où le donneur propose des délégations même si elles ne sont pas rationnelles, et où le receveur cherche une contre-offre pertinente

Une architecture modulaire

Négociation et consommation concurrentes



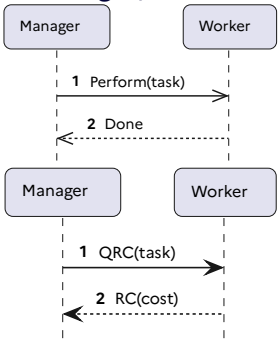
- Le ***worker*** consomme les tâches
- Le ***negotiator*** met à jour une base de croyances pour négocier les tâches avec ses pairs
- Le ***manager*** gère le lot de tâches du nœud, ordonnance les tâches à donner au *worker*, et retire ou ajoute des tâches selon les négociations effectuées

Une architecture modulaire

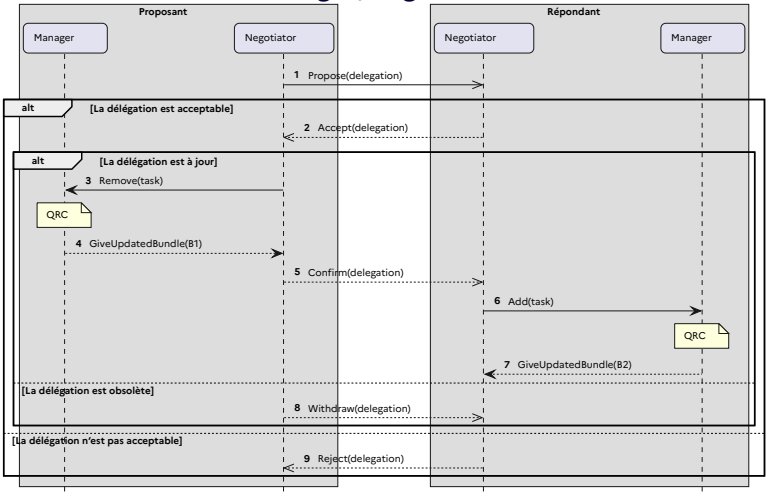
Interactions entre le *manager*, le *worker* et le *negotiator*

Université de Lille

Manager/Worker

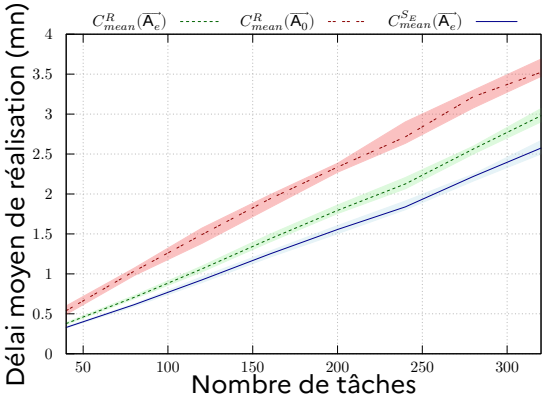


Manager/Negotiator



Expérimentations

La stratégie de réallocation améliore le *flowtime*

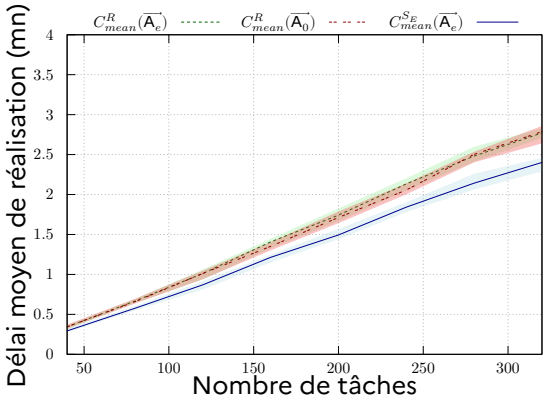


Expérimentations

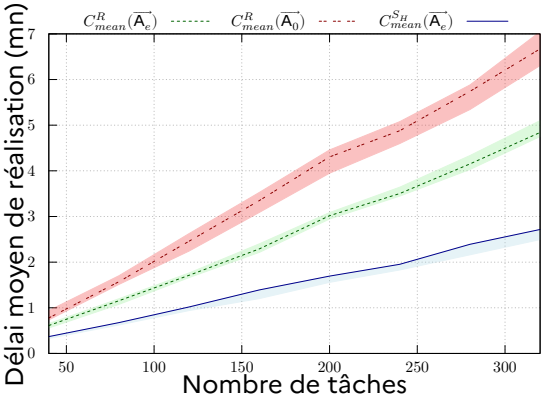
Université de Lille

La stratégie de réallocation ne pénalise pas la consommation et est robuste face aux aléas d'exécution

À partir d'une allocation stable



Avec des aléas d'exécution



Conclusion

Université de Lille

- **Approche.**
  Système multi-agent pour la réallocation de tâches sur des nœuds réalisée simultanément à la consommation afin de réduire la durée moyenne de réalisation de jobs concurrents
- **Contributions.**
  - Formalisation de la consommation de tâche et des opérations de réallocation
  - Architecture multi-agent modulaire qui permet la concurrence de la négociation et de la consommation des tâches avec **3 agents composants :**
    - un *worker* qui exécute les tâches
    - un *negotiator* qui négocie les réallocations
    - un *manager* qui coordonne localement ces opérations
- **Travaux futurs.**
  Un processus d'approvisionnement qui permet d'ajouter ou de retirer des nœuds de calcul au cours de l'exécution

Bibliographie I

Baert, Q., Caron, A.-C., Morge, M., Routier, J.-C., & Stathis, K. (2021). An adaptive multi-agent system for task reallocation in a MapReduce job. Journal of Parallel and Distributed Computing, 153, 75–88.

Beauprez, E., Caron, A.-C., Morge, M., & Routier, J.-C. (2022a). Échange de tâches pour la réduction de la durée moyenne de réalisation (V. CAMPS, Ed.). In V. CAMPS (Ed.), Trentièmes journées francophones sur les systèmes multi-agents (JFSMA), Saint-Étienne, France, Cépaduès.

Beauprez, E., Caron, A.-C., Morge, M., & Routier, J.-C. (2022b). Task Bundle Delegation for Reducing the Flowtime. In ICAART 2021, Revised Selected Papers (pp. 22–45). Springer International Publishing.

Beauprez, E., & Morge, M. (2020). Scala implementation of the Extended Multi-agents Situated Task Allocation.

Chen, Y., Mao, X., Hou, F., Wang, Q., & Yang, S. (2016). Combining re-allocating and re-scheduling for dynamic multi-robot task allocation. In Proc. of SMC.

Bibliographie II

Choi, H.-L., Brunet, L., & How, J. P. (2009). Consensus-based decentralized auctions for robust task allocation. IEEE transactions on robotics, 25(4), 912–926.

Creech, N., Pacheco, N. C., & Miles, S. (2021). Resource allocation in dynamic multiagent systems. CoRR, abs/2102.08317.

Damamme, A., Beynier, A., Chevaleyre, Y., & Maudet, N. (2015). The Power of Swap Deals in Distributed Resource Allocation. In Proc. of AAMAS.

Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. In Proc. of OSDI.

Jiang, Y. (2016). A survey of task allocation and load balancing in distributed systems. IEEE Transactions on Parallel and Distributed Systems, 27(2), 585–599.

Koenig, S., Tovey, C., Lagoudakis, M., Markakis, V., Kempe, D., Keskinocak, P., Kleywegt, A., Meyerson, A., & Jain, S. (2006). The power of sequential single-item auctions for agent coordination. In AAAI.

Bibliographie III

Lerman, K., Jones, C., Galstyan, A., & Matarić, M. J. (2006). Analysis of dynamic task allocation in multi-robot systems. The International Journal of Robotics Research, 25(3), 225–241.

Lightbend. (2020). Akka is the implementation of the Actor Model on the JVM.

Mayya, S., D'antonio, D. S., Saldaña, D., & Kumar, V. (2021). Resilient task allocation in heterogeneous multi-robot systems. IEEE Robotics and Automation Letters, 6(2), 1327–1334.

Mills-Tettey, G. A., Stentz, A., & Dias, M. B. (2007). The dynamic hungarian algorithm for the assignment problem with changing costs. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27.

Nam, C., & Shell, D. A. (2015). When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time. In Proc. of ICRA.

Turner, J., Meng, Q., Schaefer, G., & Soltoggio, A. (2018). Distributed Strategy Adaptation with a Prediction Function in Multi-Agent Task Allocation. In Proc. of AAMAS.

Bibliographie IV

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M. J., Shenker, S., & Stoica, I. (2012). Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In Proc. of the 9th USENIX Symposium on NSDI; San Jose, CA, USA.