



**HAL**  
open science

## Progression à base de tâches des communications asynchrones

Florian Reynier

### ► To cite this version:

Florian Reynier. Progression à base de tâches des communications asynchrones. COMPAS 2019 - Conférence d'informatique en Parallélisme, Architecture et Système, Jun 2019, Anglet, France. hal-02407276

**HAL Id: hal-02407276**

**<https://inria.hal.science/hal-02407276>**

Submitted on 12 Dec 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## LES COMMUNICATIONS MPI

- Standard très utilisé en calcul haute performance
  - Communication par messages
  - Programmation sur mémoire distribuée
  - Communications non bloquantes
  - L'appel rend directement la main
  - Communications en "tâches de fond"
  - Gain de temps processeur
  - Opérations collectives
  - Opérations basique sur plus de deux noeuds
  - Broadcast, reduce, gather ...

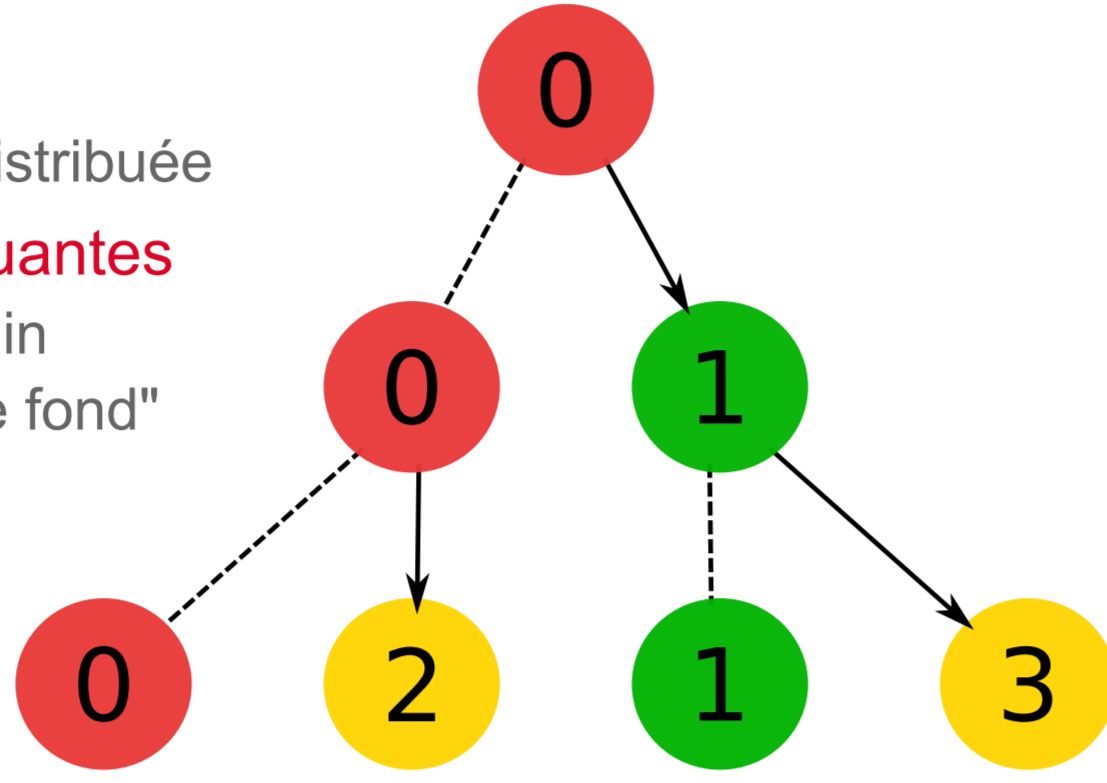


fig1: MPI\_Bcast sur 4 noeuds

## PROBLEMATIQUE

- Recouvrir les communications par du calcul
  - Paralléliser les communications et le calcul
  - Minimiser leur impact mutuel
- Comment bien gérer les ressources disponibles?
  - Répartir les communication et le calcul
  - Quelle priorité ?
  - Quelle politique de répartition ?

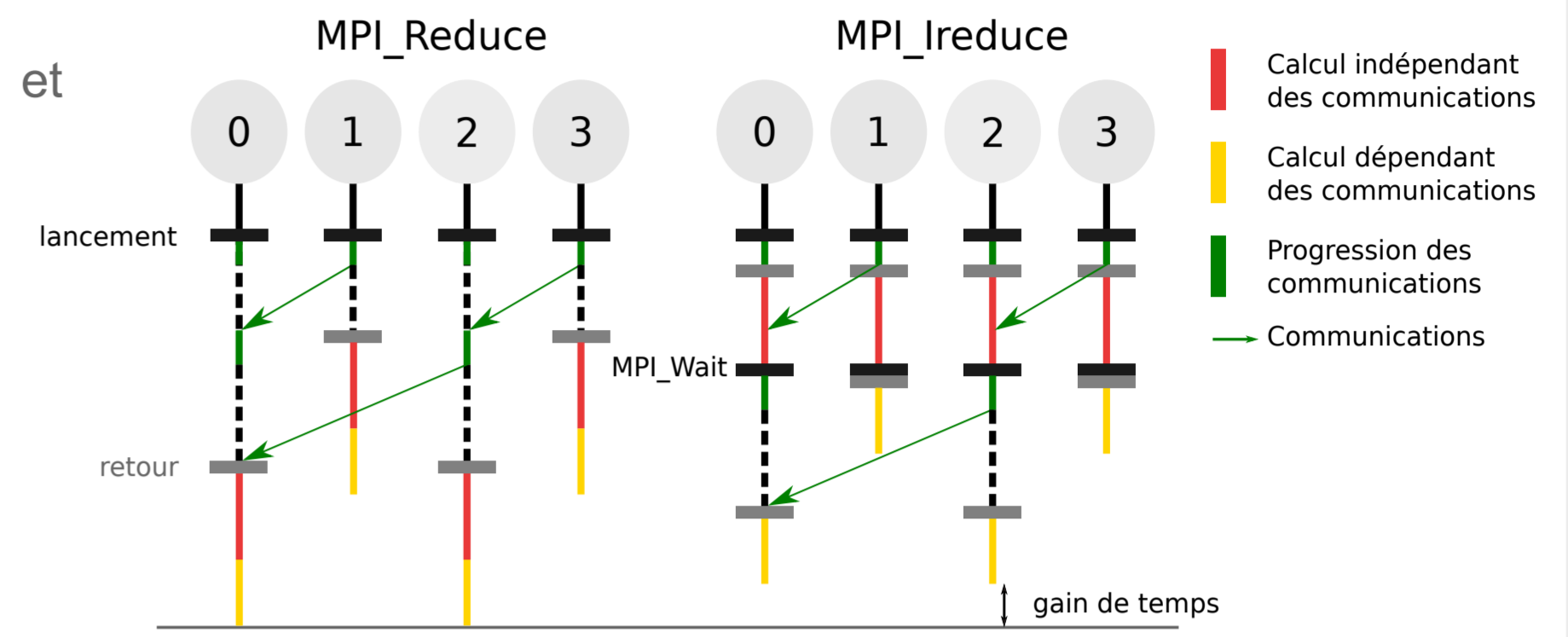
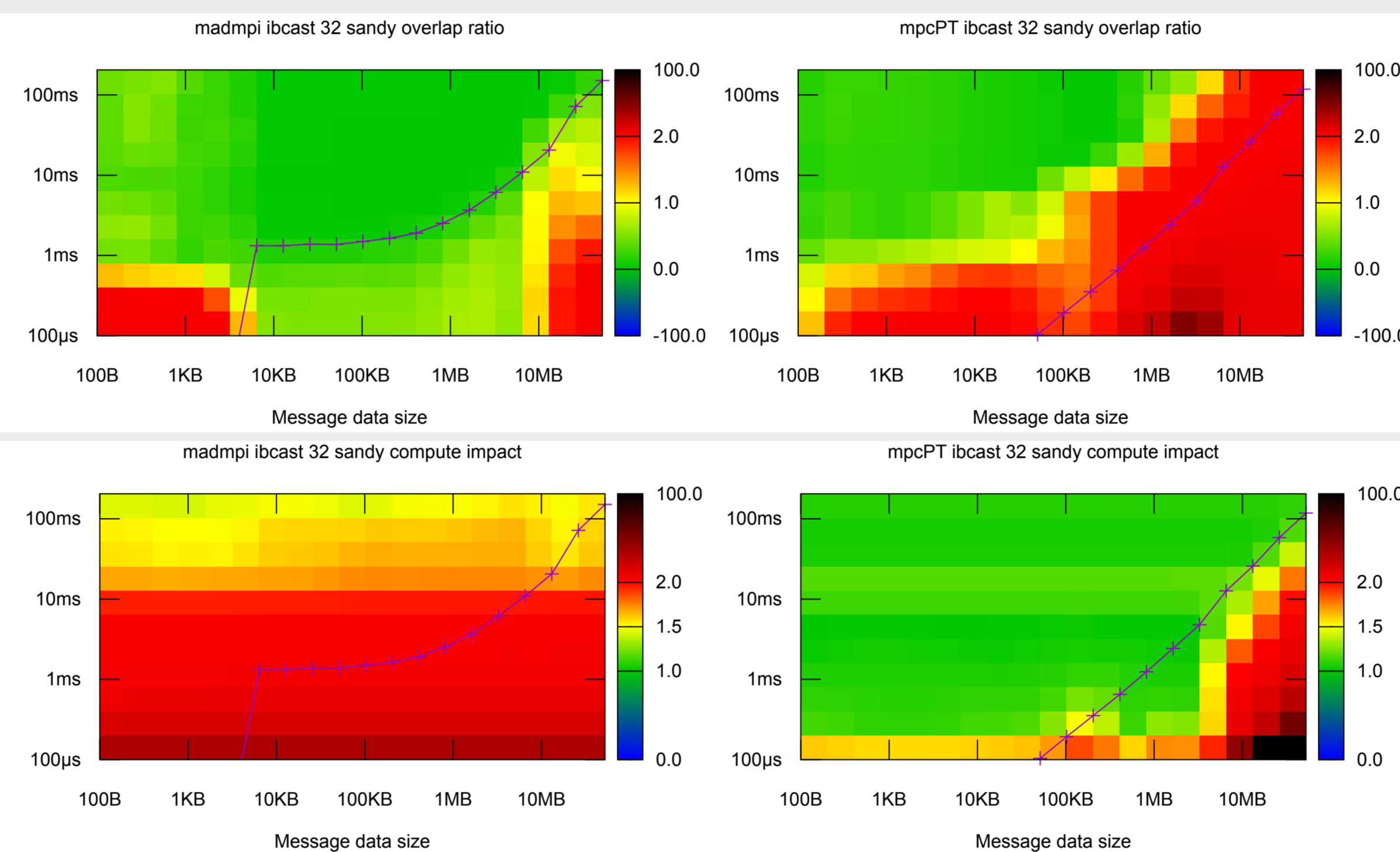


fig2: Comparaison d'un reduce bloquant / non bloquant

CONTEXTE

## BENCHMARKING DES COLLECTIVES NON BLOQUANTES

- Synchronisation des horloges
  - Paralléliser les communications et le calcul
  - Minimiser leur impact mutuel
  - Mesurer une collective
  - Paralléliser les communications et le calcul
  - Minimiser leur impact mutuel



- Ratio d'overhead
  - Mesure la parallélisation du calcul et des communication
  - $overhead = \frac{t_{measured} - \max(t_{comm}, t_{col})}{\min(t_{comm}, t_{col})}$
  - ratio = 0: recouvrement parfait
  - ratio = 1: comm et calcul sérialisé
  - ratio > 1: perte de temps
- Ratio d'impact sur le calcul
  - Mesure l'impact de la parallélisation sur le temps de calcul
  - $impact = \frac{t_{calc}}{t_{calc-pure}}$
  - ratio = 1: aucun impact sur le calcul
  - ratio > 1: le calcul est impacté

CONTRIBUTIONS

## NEW MADELEINE[2]

- Une bibliothèque de communication
    - Conçue sur le principe des communications asynchrones
    - Optimisation reposant sur la gestion dynamique de paquets
    - Modulable: permet d'expérimenter diverses stratégies d'ordonnancement
  - Une bibliothèque de communication
    - Implante le modèle à base de tâches
    - Les tâches intermédiaires sont créées en fonction des évènements réseaux
  - MadMPI: une implémentation MPI basée sur New Madeleine
    - Permet le fonctionnement d'application MPI avec NewMadeleine
    - Basé sur quatre opérations basiques du standard, commune avec NewMadeleine:
      - MPI\_Isend, MPI\_Irecv
      - MPI\_Wait, MPI\_Test
    - Implémentation de collectives non bloquantes: MPI\_Ibcast, MPI\_Ireduce

## MPC[1]

- Un environnement d'exécution unifié
    - Implémentation OpenMP, MPI, Pthread
    - Implémentation MPI basée sur des threads
    - Threads utilisateurs: gérés par un ordonnanceur propre à MPC
    - Optimisé pour le fonctionnement simultanés des modèles de programmation

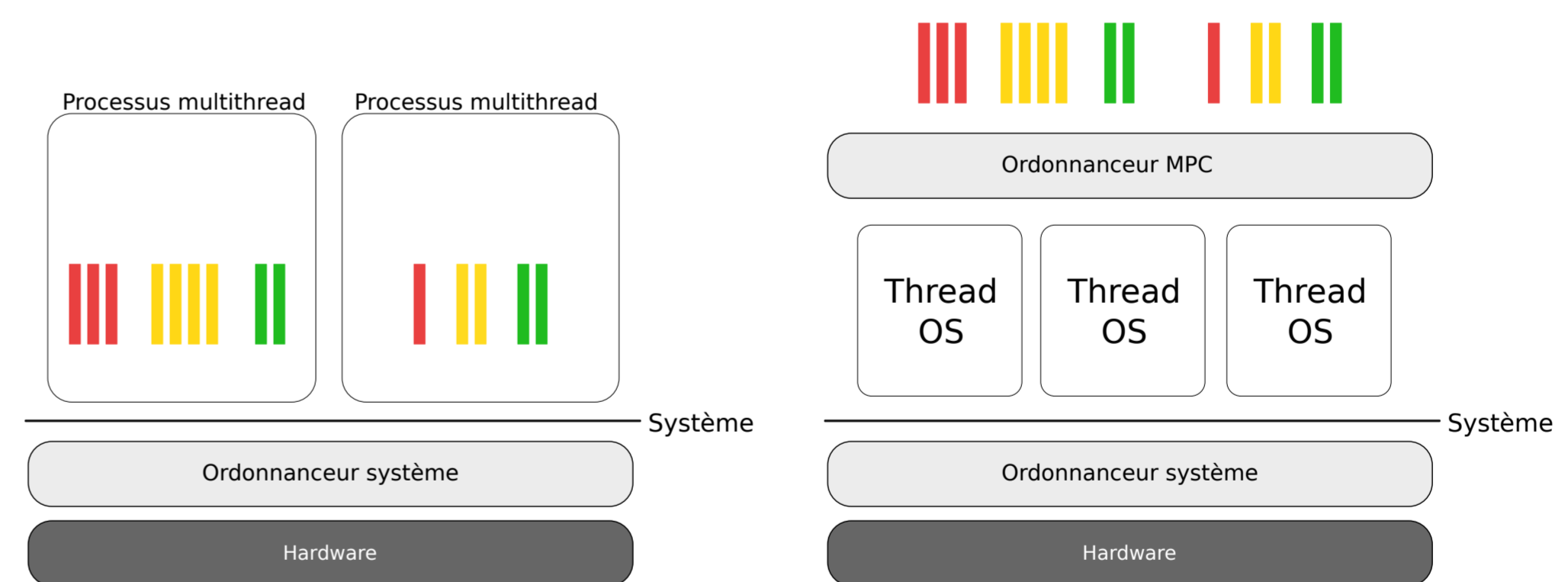


fig3: fonctionnement multimodèle classique / MPC

- Ajout d'un moteur de tâches au sein de MPC
  - Basé sur des tâches légères
  - Gestion de dépendances pour les collectives
  - Moteur basé sur les threads utilisateurs MPC

## BENCHMARKING DES COLLECTIVES NON BLOQUANTES

- Définition d'une tâche
    - Une tâche peut être définie comme une portion de code à exécuter
    - Contrairement au thread, elle n'est pas liée à un contexte d'exécution
    - Dans notre cas: des tâches légères
  - Un modèle léger et souple
    - Les tâches ne sont liées à aucun coeur, elles peuvent être plus finement réparties dynamiquement à l'exécution.
    - Utile dans le cas où le code n'est pas très lourd pour le processeur, ce qui est le cas pour des communications

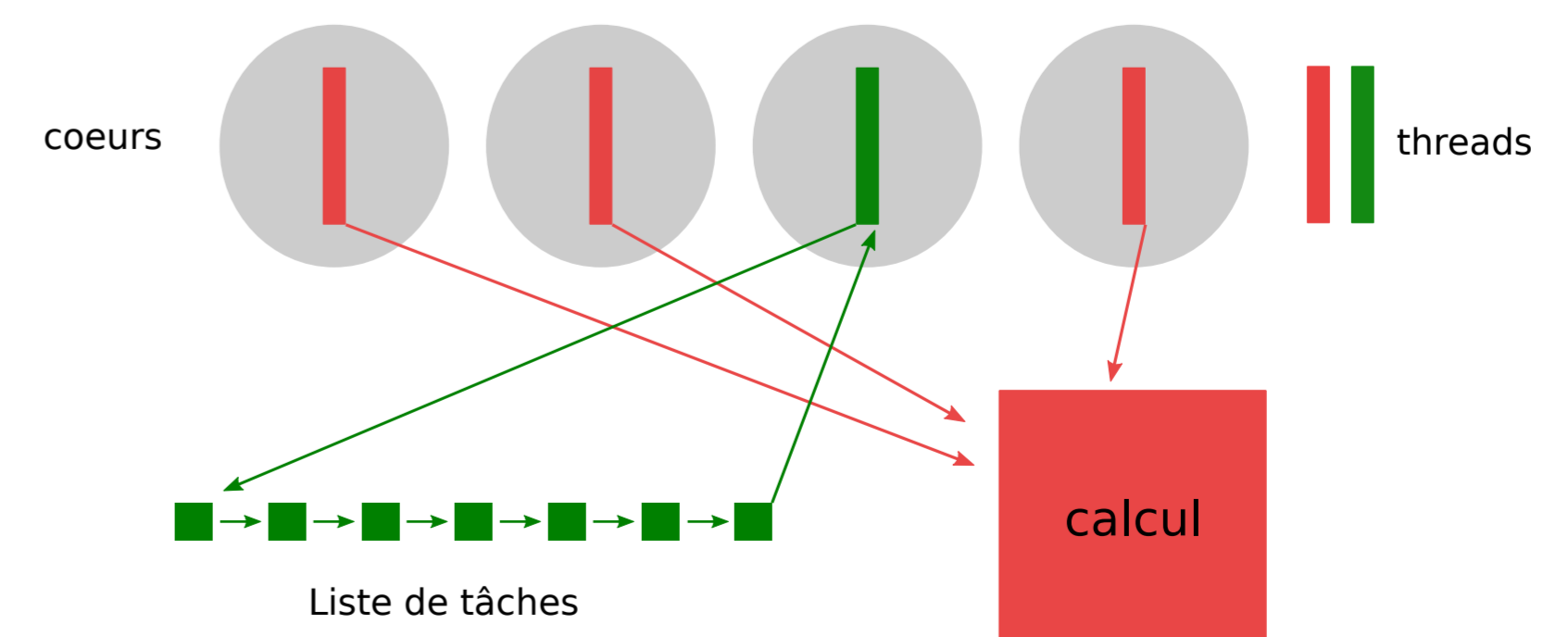


fig4: Modèle à base au sein d'un noeud