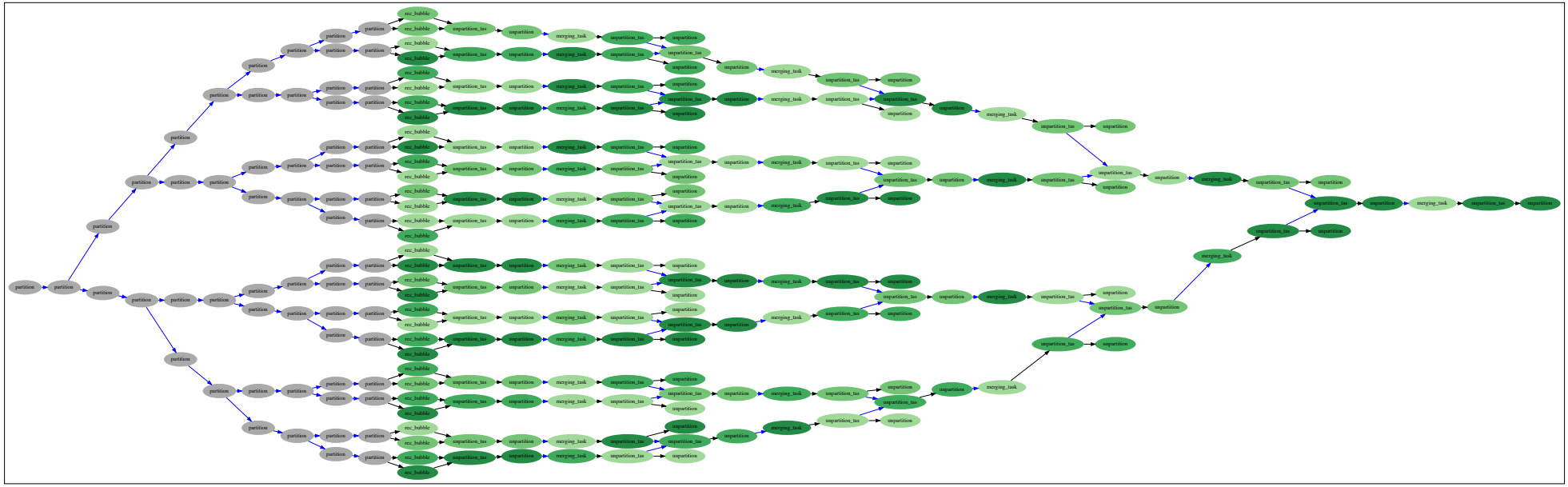
Programming with hierarchical tasks

Control the task flow

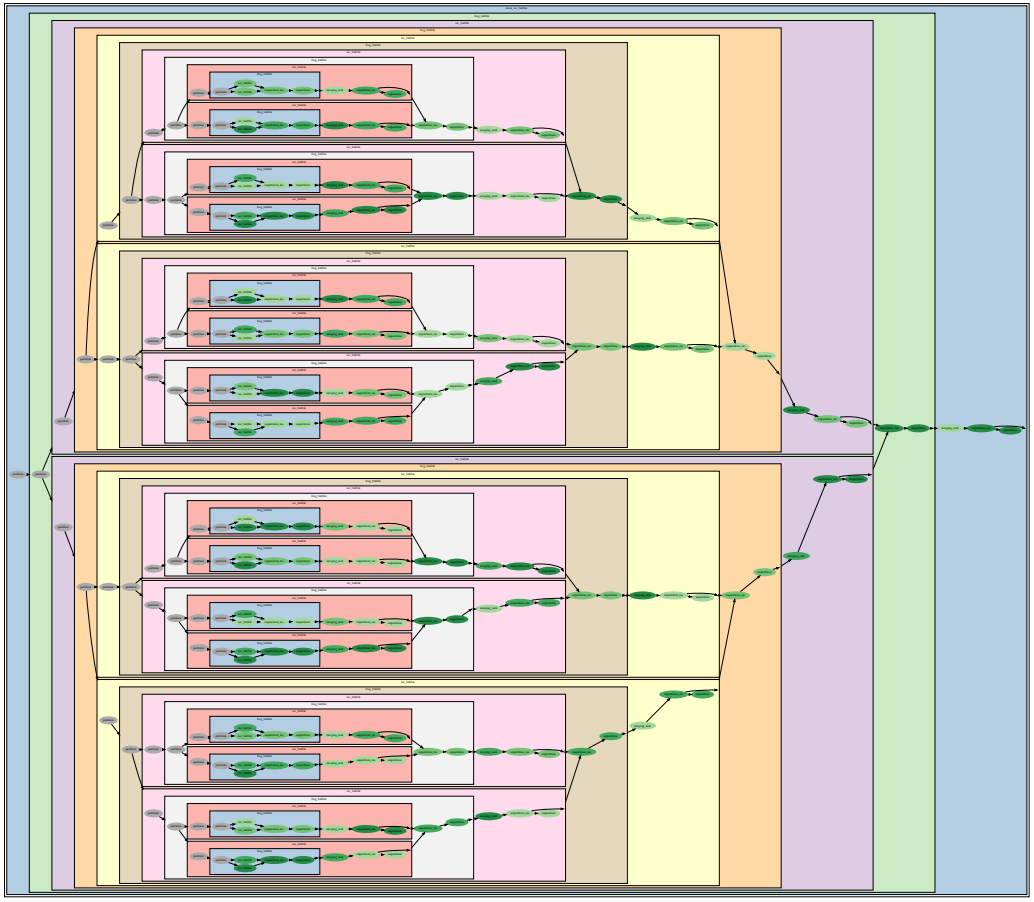
EQUIPE PROJET
STORM
INRIA
Bordeaux-Sud-Ouest

Exemple : merge sort



On sait générer récursivement un graphe de tâches

Exemple : merge sort



Comment générer ce dag à l'aide de tâches récursives ?

Introduction

- Tâches hiérarchiques
  - Définition : des tâches qui génèrent récursivement des tâches
  - Des *Bulles* dans jargon runtime / storm
- Motivations
  - Granularité [Wu & al IPDPS 2016]
    - Grosses taches sur GPU, dag de petites tâches sur CPU
  - Faciliter la programmation avec des structures de données hiérarchiques
    - h-matrix [Airbus, CEA, BSC, Valencia]
  - Abstraction du DAG pour un meilleur ordonnancement
    - Pouvoir exécuter des algorithmes coûteux sur un DAG de haut niveau
  - Contourner certaines limites du STF

Contourner certaines limites du STF

- Générer le dag de calcul juste à temps à partir d'un graphe de bulles
  - Choisir une implémentation au runtime
    - Tache : CPU, OpenMP, GPU,
    - Bulle : StarPU, Multi-GPU, MPI,…
  - Réduction de l'empreinte mémoire de StarPU
    - Objectif : les tâches en mémoire seront bientôt exécutées
- Soumission parallèle des tâches
  - Deux bulles indépendantes peuvent s'exécuter en même temps
  - Étalement de la soumission

Contourner certaines limites du STF

- Augmente le pouvoir d'expression de StarPU
  - Mode commute de StarPU
  - Deux bulles peuvent commuter contrairement aux sous dags.

- Éviter de bloquer le thread de soumission pour des raisons techniques
  - Encombrement mémoire, tag MPI, …
  - Acquire / Release,
  - gestion mémoire,

- Ordre de soumission moins crucial

Premier modèle

- Jérôme Clet-Ortega, Arthur Chevalier, Léo Villeveygoux
- Validation Chameleon QR-MUMPS
- Surcouche de StarPU
  - Très peu de modification dans starPU
  - 👎 Un niveau d'imbrication
  - 👎 Nécessite la réécriture des sources (macros)

```
for (k = 0; k < nblocks; k++)
{
    k_val[k] = k;
    starpu_bubble_insert(CHOLESKY_BUBBLE,
                         STARPU_PRIORITY, STARPU_MAX_PRIO,
                         STARPU_RW, dataA,
                         STARPU_VALUE, &(k_val[k]), sizeof(k),
                         STARPU_VALUE, &nblocks, sizeof(nblocks),
                         0);
}
```

Modèle intégré à StarPU

- Gwénolé Lucas
- Objectifs
  - Tâches récursives & gestion des données
    - Faciliter l'utilisation des structures de données hiérarchiques
  - Intégrer le modèle au sein de StarPU
    - Compatibilité avec l'existant
    - Tout en restant *« Sam compliant »*
    - Tactique : on code avec StarPU avant de coder dans StarPU.

Modèle interne à StarPU

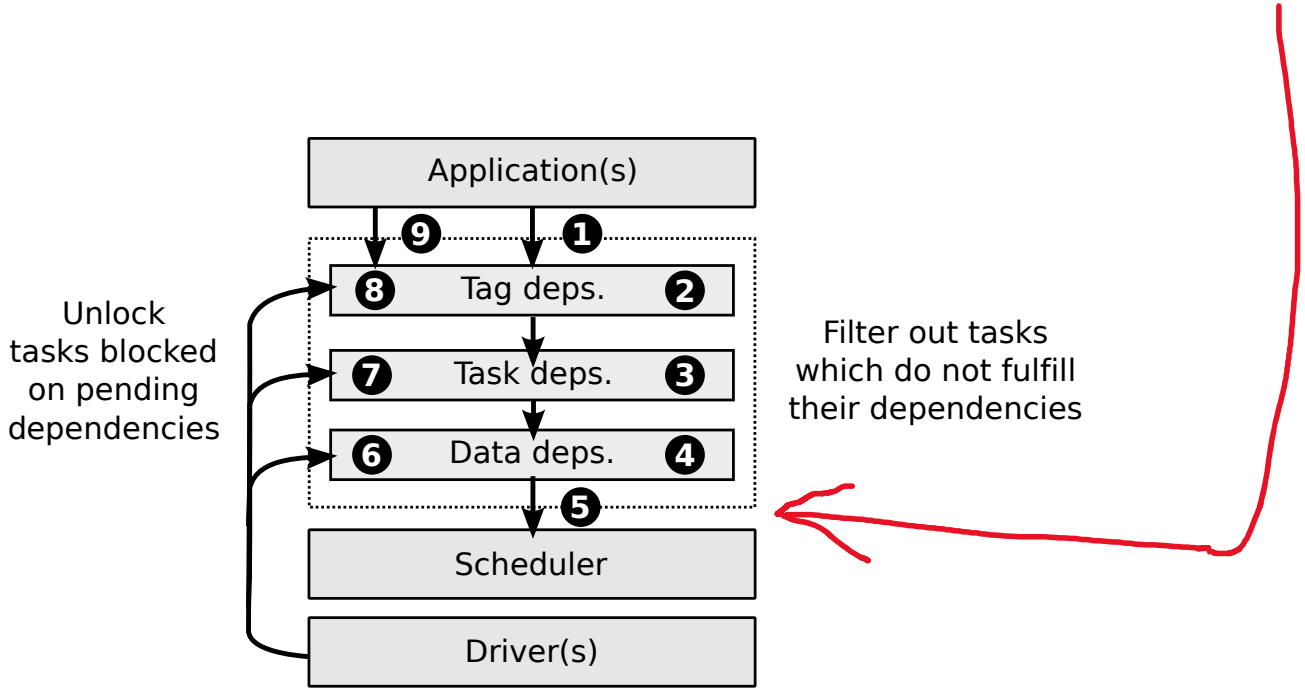
- Insertion d'une tâche qui peut devenir une bulle

```
starpu_task_insert(&merge_sort_codelet,
        STARPU_RW, subdata[i],
        STARPU_BUBBLE_FUNC, &is_bubble,
        STARPU_BUBBLE_FUNC_ARG, subdata[i],
        STARPU_BUBBLE_FUNC_GEN_DAG, &merge_sort_rec_bubble,
        STARPU_BUBBLE_FUNC_GEN_DAG_ARG, subdata[i],
        STARPU_BUBBLE_PARENT, me,
        STARPU_NAME, "rec_bubble", 0);
```

- Une fonction de décision is_bubble()
- Un codelet pour la version tâche
- Une fonction de génération pour la version bulle

Modification de StarPU : Tâche ou Bulle ?

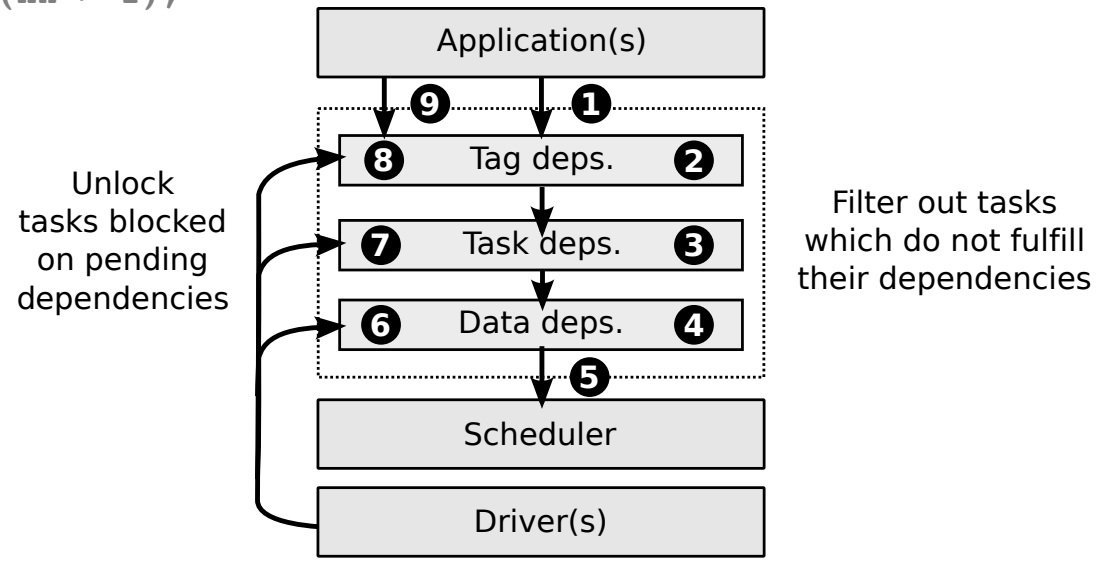
- Introduction du mécanisme de décision tâche ou bulle :
  - Appel de la fonction de décision dès que les dépendances sont satisfaites



Exemples de fonction de décision

Pour le moment : fonctions basées sur des données contenues dans le handle.

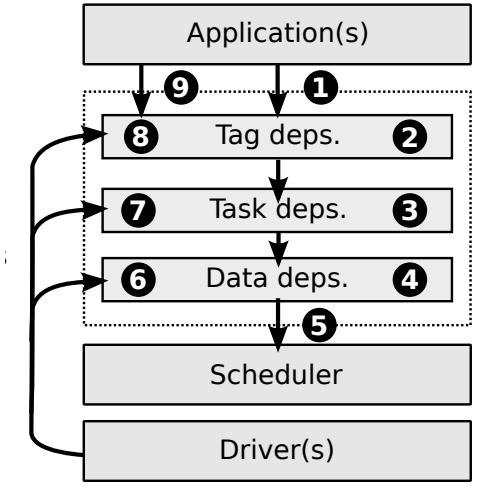
```
int is_bubble(struct starpu_task *t, void *arg)
{
  starpu_data_handle_t data = (starpu_data_handle_t)arg;
  int nx = starpu_vector_get_nx(data);
  return (nx > 1);
}
```



- **Comment intégrer / automatiser la prise de décision ?**

Modification de StarPU
Génération des sous tâches

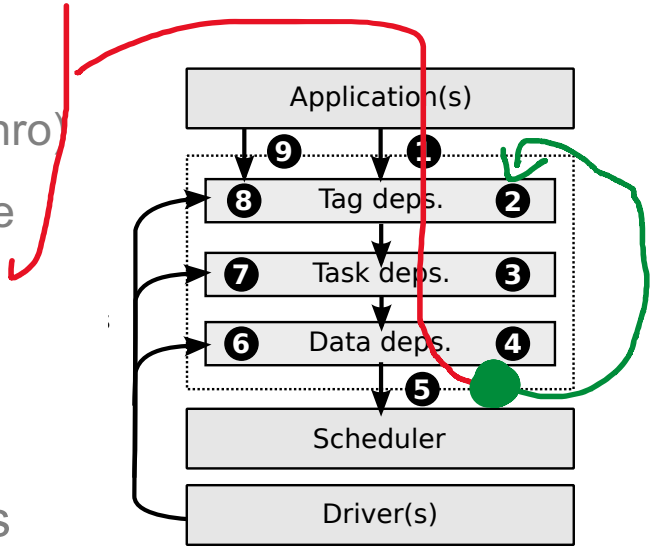
1. La bulle travaille uniquement au niveau des handles
   - Le transfert des données est inutile
   - Suppression du codelet (= tâche de synchro)
   - Génération des sous-tâches dans la foulée



2. La bulle utilise les handles et les données
   - Substitution du codelet par un autre générant le sous graphe
   - La bulle doit alors passer par l'ordonnanceur pour être exécutée

Modification de StarPU : Génération des sous tâches

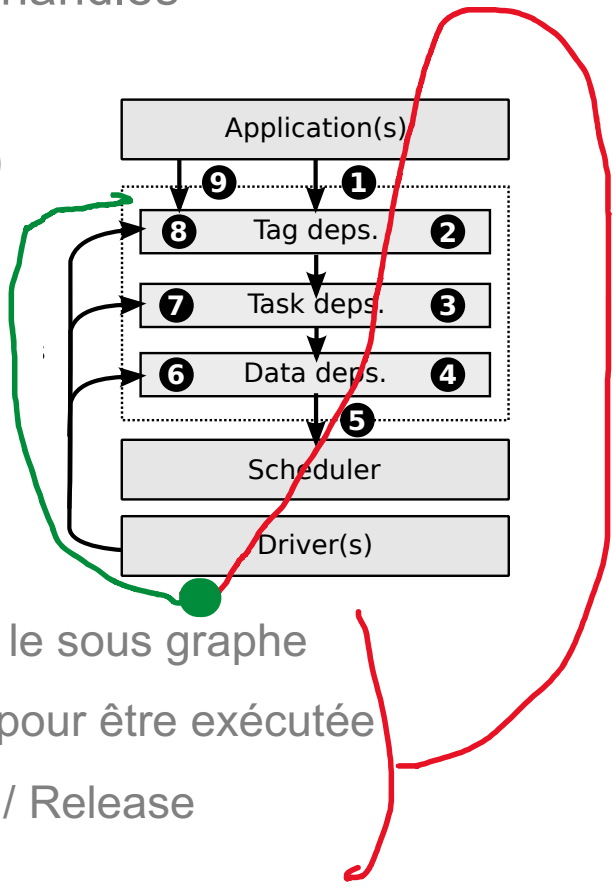
- La bulle travaille uniquement au niveau des handles
  - Le transfert des données est inutile
  - Suppression du codelet (= tâche de synchro)
  - Génération des sous-tâches dans la foulée



- La bulle utilise les handles et les données
  - Substitution du codelet par un autre générant le sous graphe
  - La bulle doit alors passer par l'ordonnanceur pour être exécutée

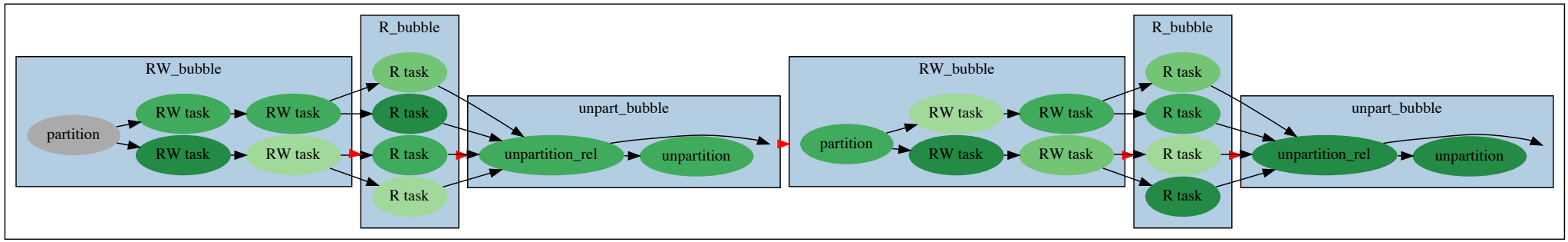
Modification de StarPU
Génération des sous tâches

- La bulle travaille uniquement au niveau des handles
  - Le transfert des données est inutile
  - Suppression du codelet (= tâche de synchro)
  - Génération des sous-tâches dans la foulée

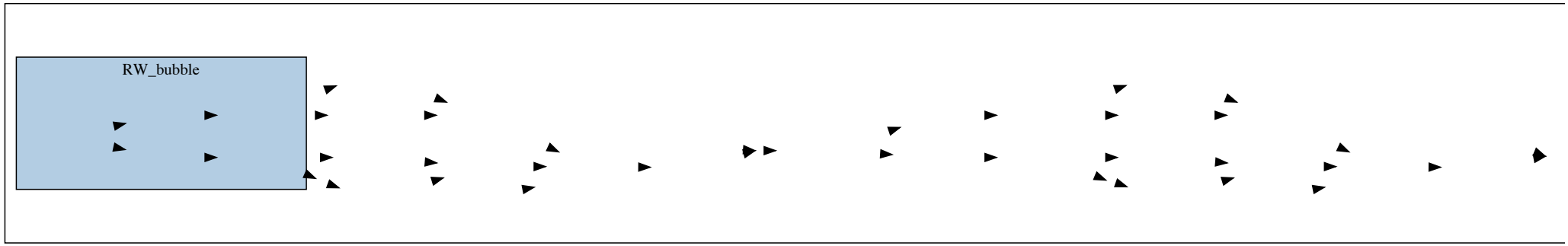


- La bulle utilise les handles et les données
  - Substitution du codelet par un autre générant le sous graphe
  - La bulle doit alors passer par l'ordonnanceur pour être exécutée
  - ➢ Utile pour automatiser le mécanisme Acquire / Release

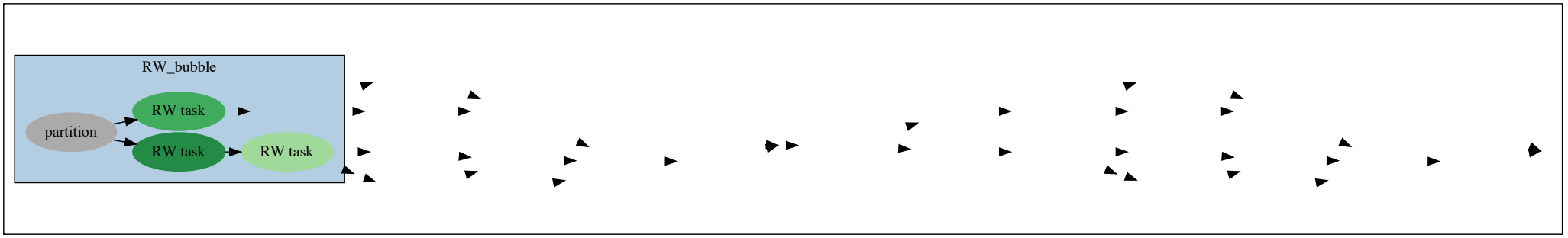
Exemple de génération



1 Première soumission

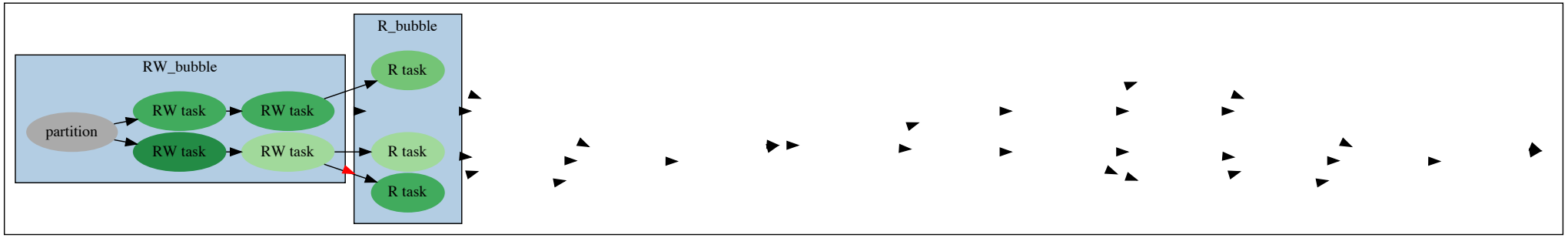


5



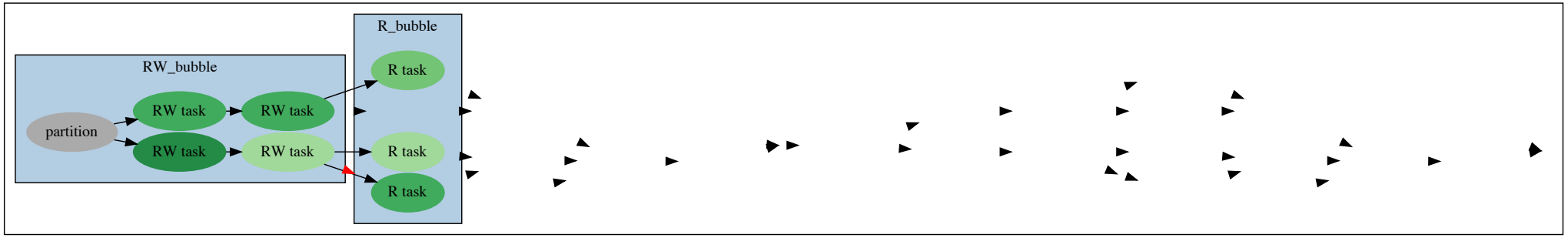
Le thread de soumission exécute immédiatement la première bulle

10



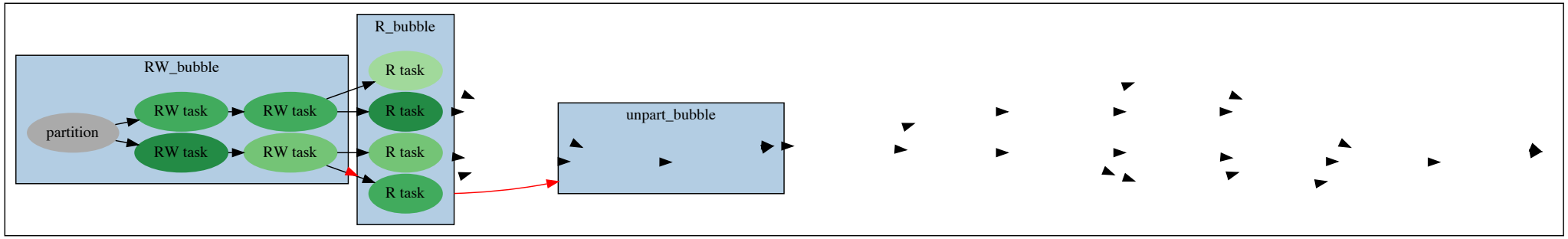
Le thread de soumission exécute directement la seconde

10

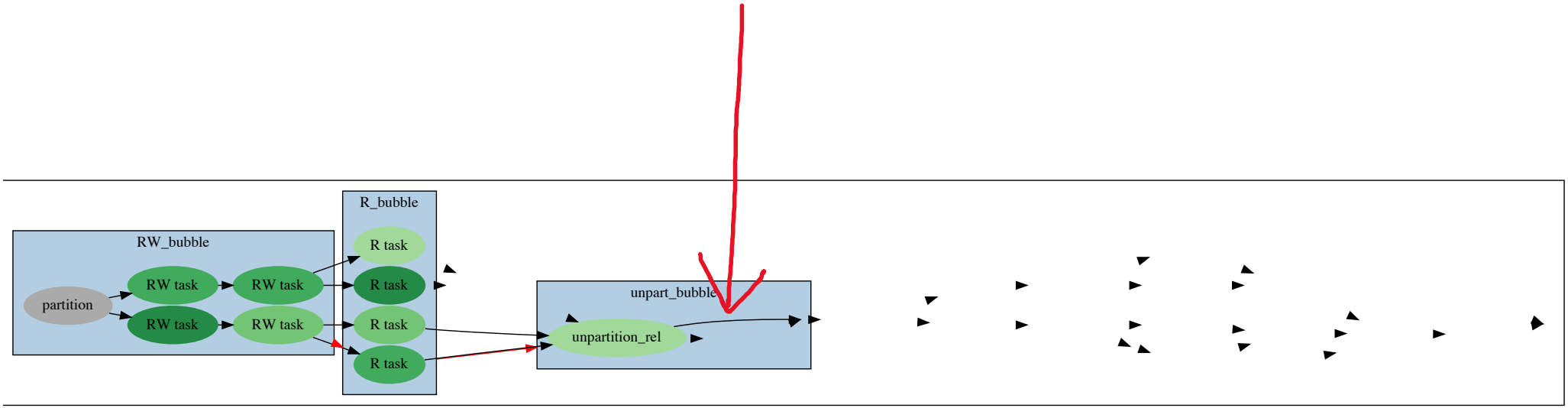


Le thread de soumission exécute directement la seconde
La soumission n'est pas liée au calcul…

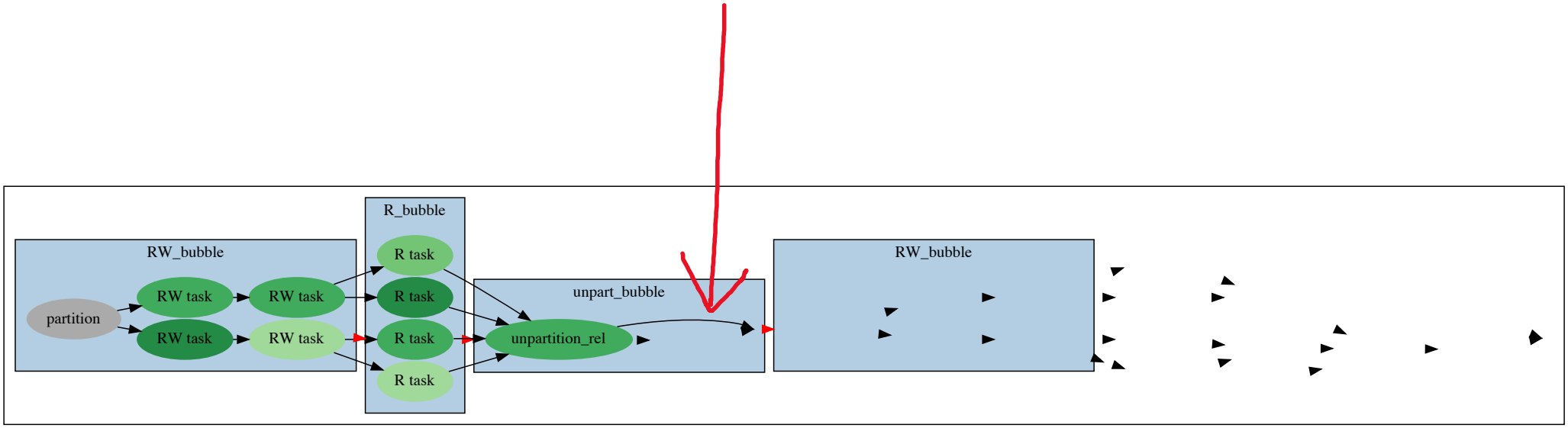
12


R_bubble
RW_bubble
R task
RW task
RW task
R task
partition
RW task
RW task
R task
unpart_bubble
R task

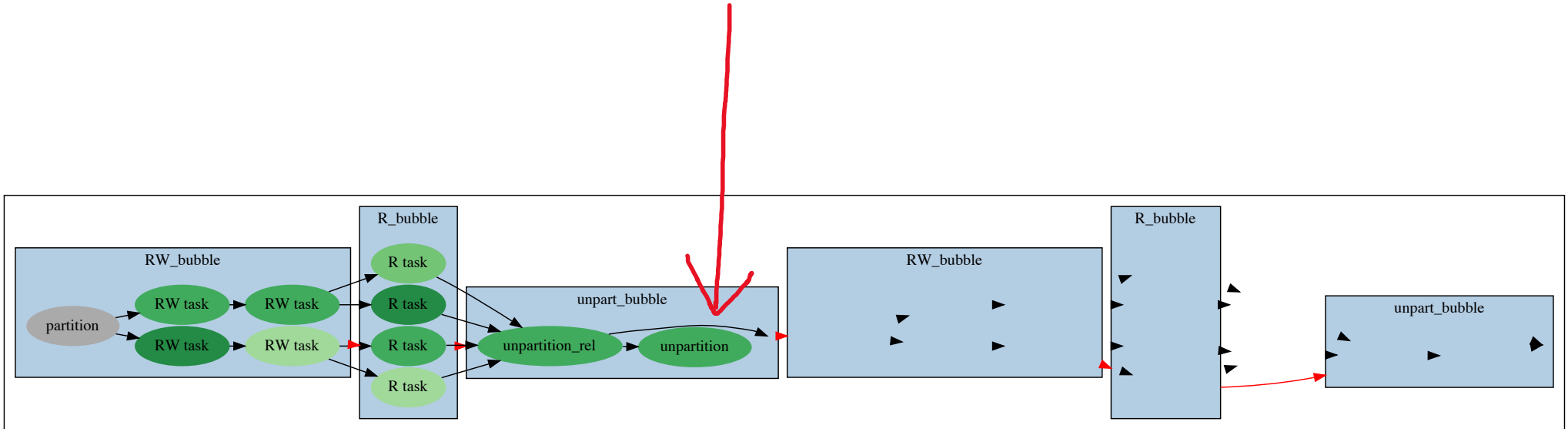
14


R_bubble
RW_bubble
R task
RW task
RW task
partition
R task
RW task
RW task
R task
unpart_bubble
R task
unpartition_rel

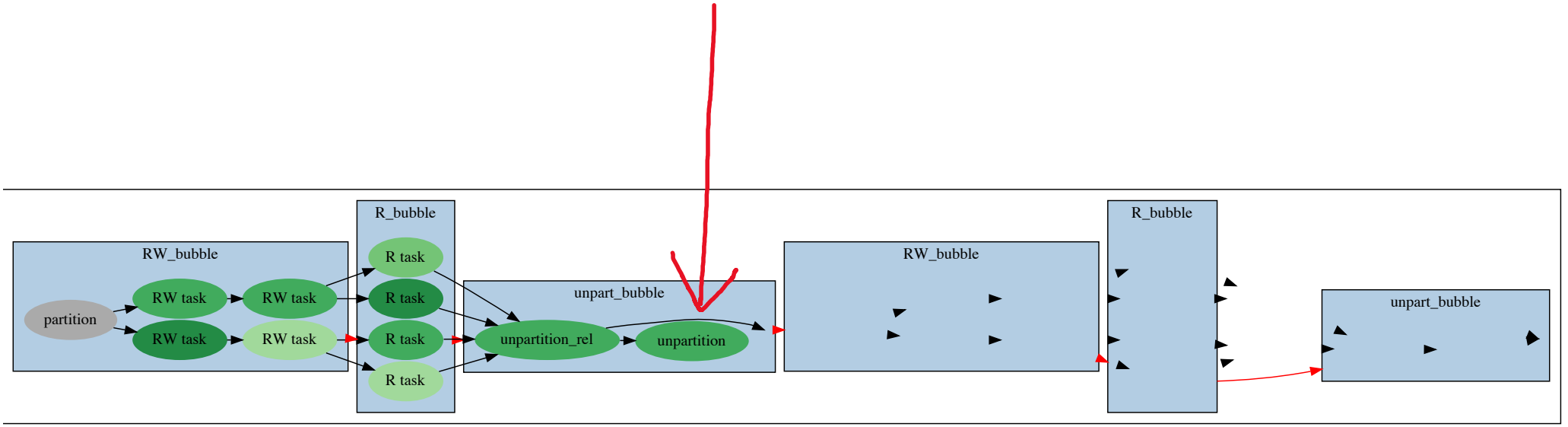
16


R_bubble
RW_bubble
R task
RW task
RW task
R task
partition
RW task
RW task
R task
unpart_bubble
R task
unpartition_rel
RW_bubble

20

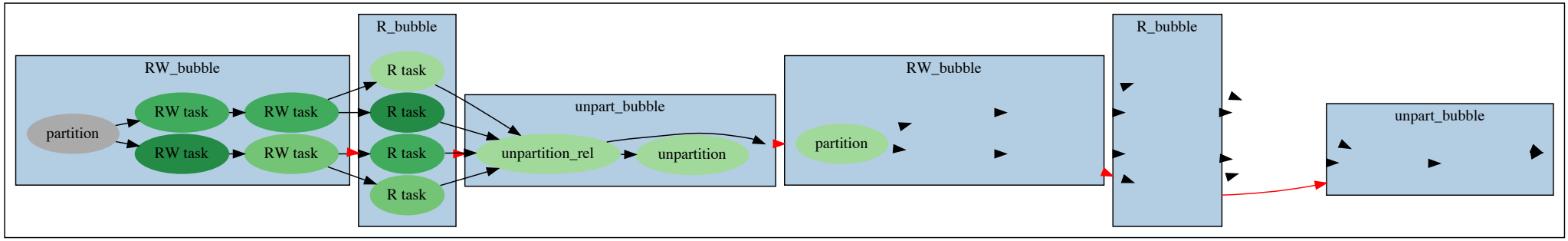

R_bubble
RW_bubble
partition
RW task
RW task
RW task
RW task
R task
R task
R task
R task
unpart_bubble
unpartition_rel
unpartition
RW_bubble
R_bubble
unpart_bubble

25

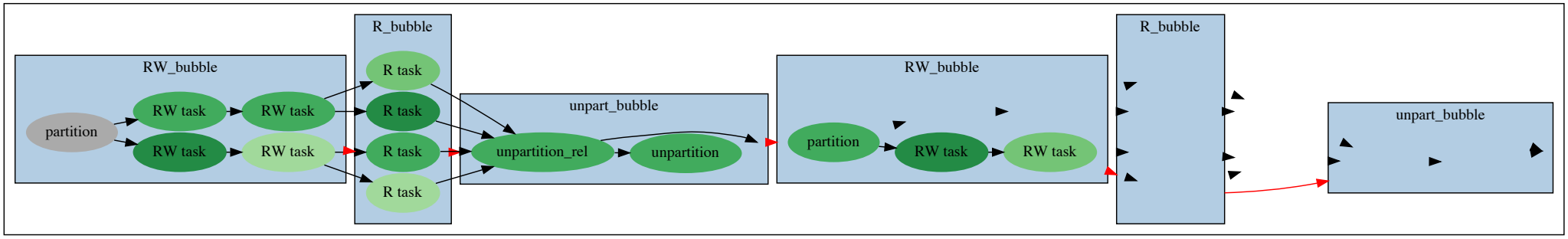


Ici la terminaison de unpart_bubble dépend de unpartition_rel.
Les bulles suivantes sont générées sans être exécutées.
Elles attendent la terminaison de unpart_bubble.

28

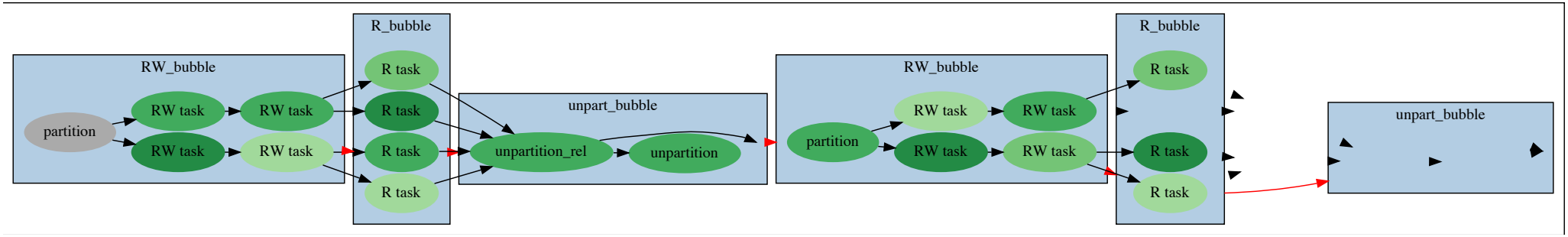

RW_bubble
partition
RW task
RW task
RW task
RW task
R_bubble
R task
R task
R task
R task
unpart_bubble
unpartition_rel
unpartition
RW_bubble
partition
R_bubble
unpart_bubble

30

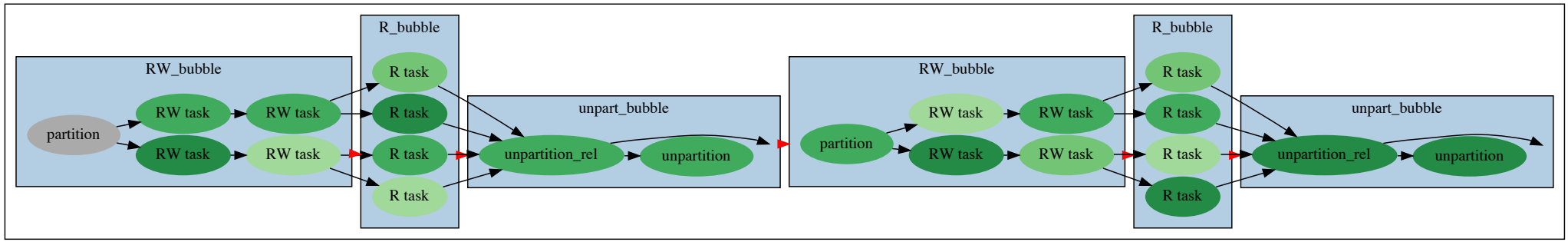


unpart_bubble est terminée.
Les bulles suivantes éclatent en série.

35

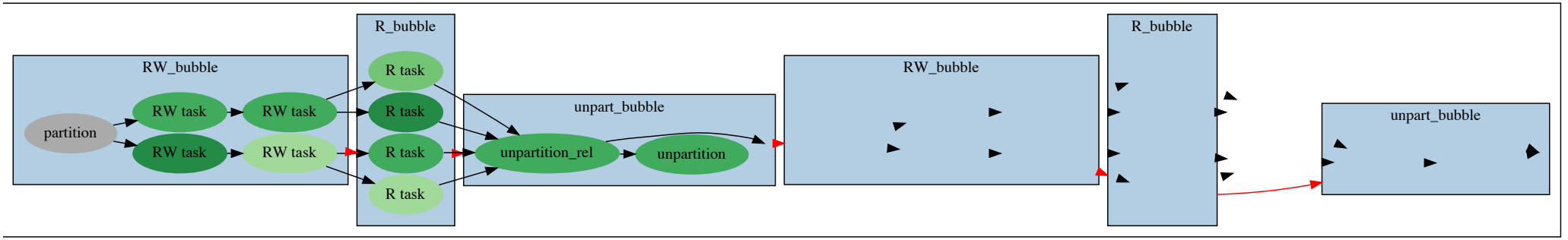


unpart_bubble est terminée.
Les bulles suivantes éclatent en série.


R_bubble
RW_bubble
partition
RW task
RW task
RW task
RW task
R task
R task
R task
R task
unpart_bubble
unpartition_rel
unpartition
R_bubble
RW_bubble
partition
RW task
RW task
RW task
RW task
R task
R task
R task
R task
unpart_bubble
unpartition_rel
unpartition

Modification de StarPU :

- Mécanisme de retardement de la terminaison d'une tâche :
  - Mise en place d'un compteur de références
  - Mise en place par le programmeur d'un callback sur certaines sous-tâches pour décrémenter le compteur



- Comment automatiser cela ?
  - Après l'exécution d'une des premières tâches ?
  - Dès qu'une sous donnée est disponible ?

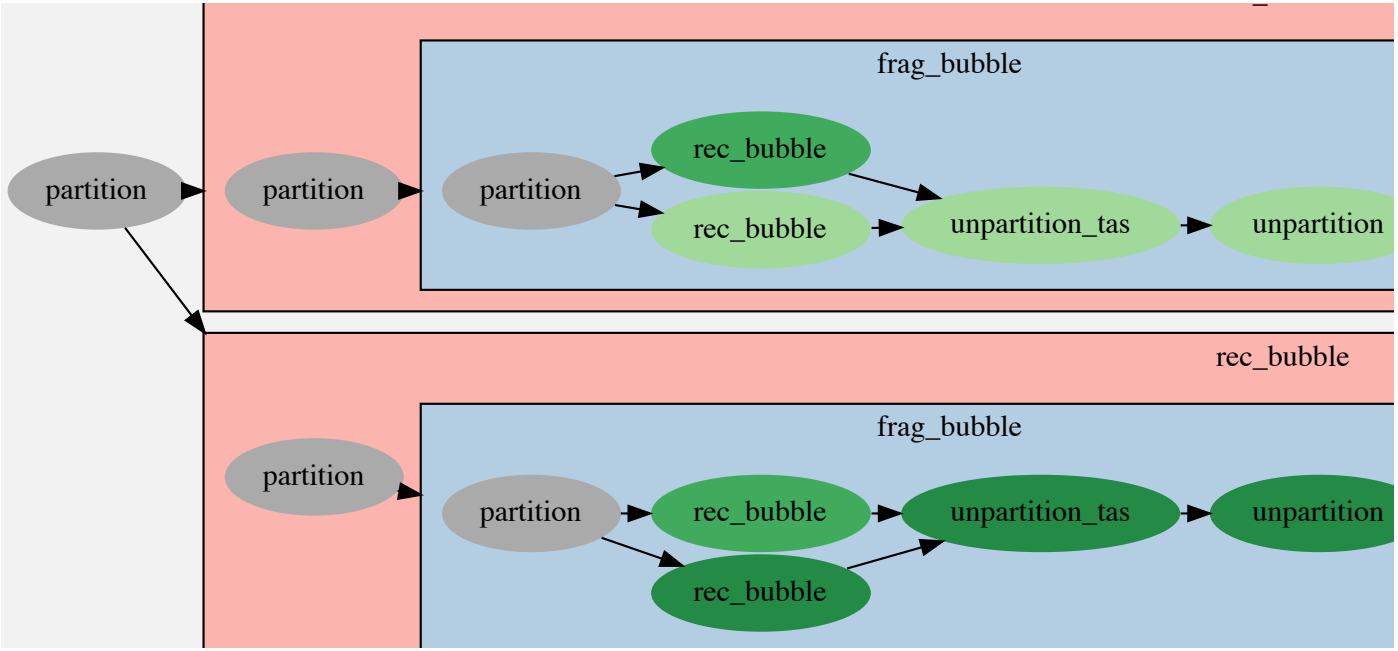
Gestion des données

- Données découpées récursivement grâce au mécanisme de partitionnement de StarPU
- Génération concurrente de tâches à différents niveaux de découpage
  - Distinction entre *utilisation des handles* vs *utilisation des données*
  - *Bonne pratique : les sous-tâches générées portent sur les sous-handles.*
- Mécanisme pour vérifier qu'un handle est découpé ou non.
  - Exemple : tester si les données sont / seront découpées

```
int is_bubble(void *arg)
{
  starpu_data_handle_t data = (starpu_data_handle_t)arg;
  if (starpu_data_get_nb_children_async(data) == 0)
    return 1;
}
```

Gestion des données

- Comment créer un sous-niveau ?



Gestion des données

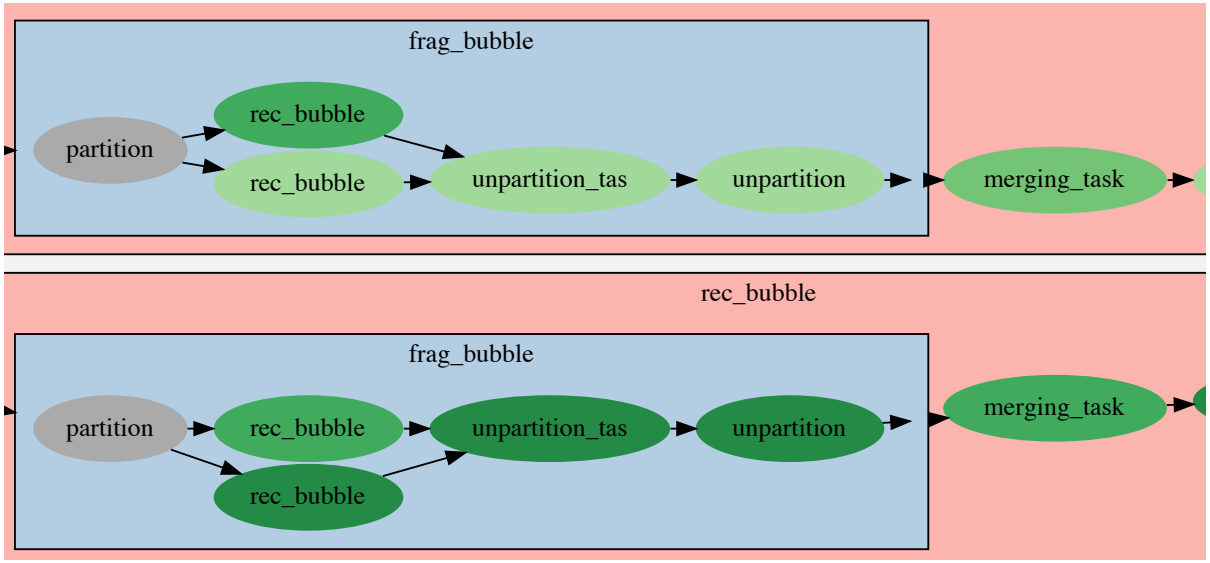
- Comment créer un sous-niveau ?
  - On désactive le pilotage automatique de StarPU

```
void merge_sort_frag_bubble(struct starpu_task *t, void *arg)
{
  starpu_data_handle_t data = (starpu_data_handle_t)arg;
  starpu_data_handle_t *subdata = malloc(PARTS*sizeof(starpu_data_handle_t));

  starpu_data_partition_plan(data, &my_vector_filter, subdata);

  for(int i=0 ; i<PARTS ; i++)
    starpu_data_partition_not_automatic(subdata[i]);

  starpu_data_partition_submit_sequential_consistency_task(data, PARTS,
     subdata, 0, t);
…
```

Formules magiques à automatiser !

Gestion des données

- Comment supprimer un sous-niveau ?
  - Insérer une tâche de recollement des sous handles,
  - La bulle ne doit se terminer qu'après le recollement
    - on a désactivé le pilotage automatique de StarPU



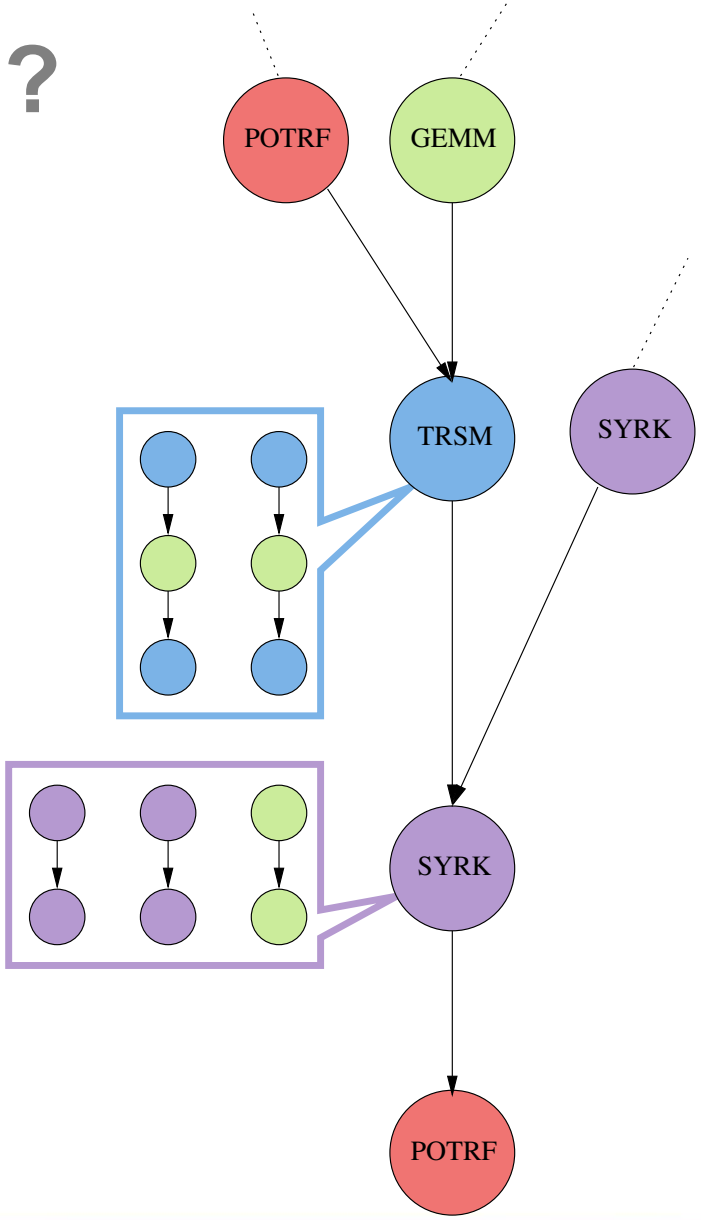
Gestion des données

- Comment supprimer un sous-niveau ?
  - Insérer une tâche de recollement des sous handles,
  - La bulle ne doit se terminer qu'après le recollement
    - on a désactivé le pilotage automatique de StarPU

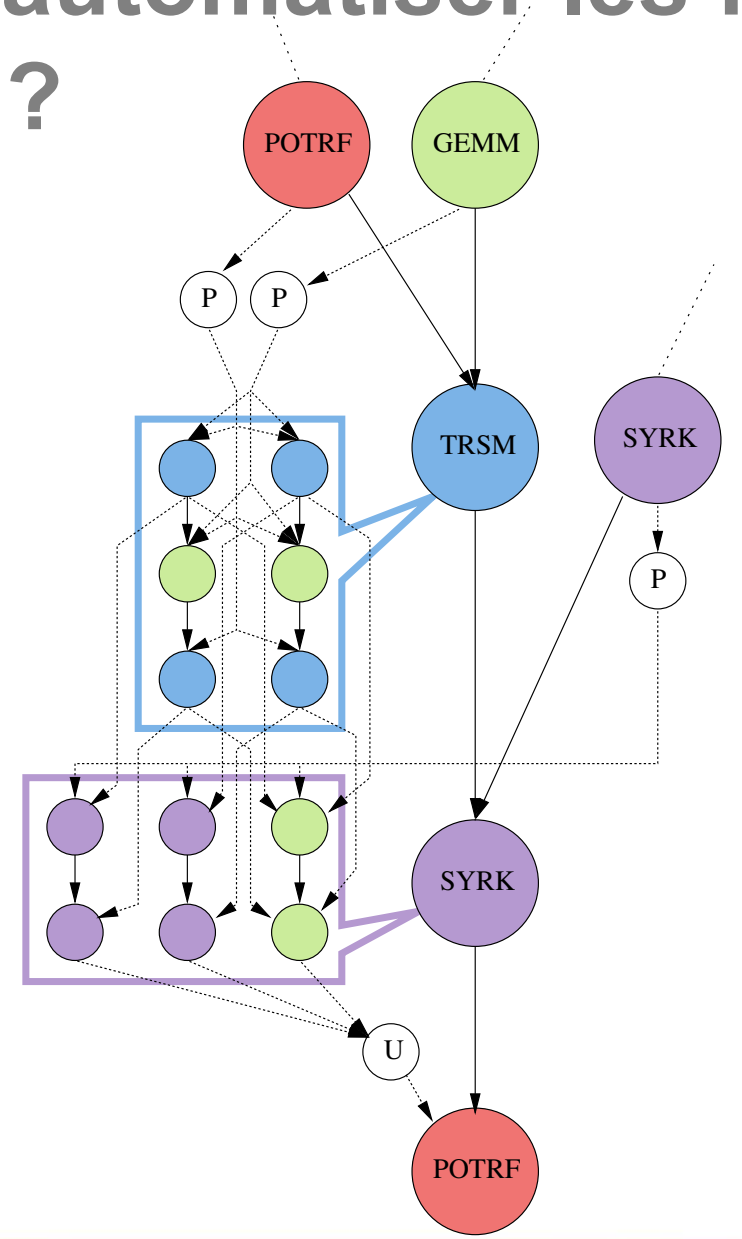
```
starpu_task_end_dep_add(t, 1);

ret = starpu_task_insert(&unpartition1_codelet,
        STARPU_DATA_ARRAY, subdata, f1.nchildren,
        STARPU_VALUE, &m_a, sizeof(struct merge_arg*),
        STARPU_BUBBLE_PARENT, t,
        STARPU_CALLBACK_WITH_ARG_NFREE, &starpu_task_end_dep_release, t,
        STARPU_NAME, "unpartition_task", 0);
```

Formules magiques à automatiser !

Comment automatiser les formules magiques ?

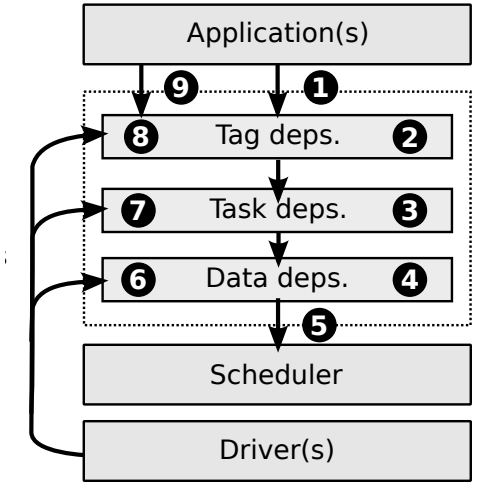


Comment automatiser les formules magiques ?



Comment automatiser les formules magiques ?

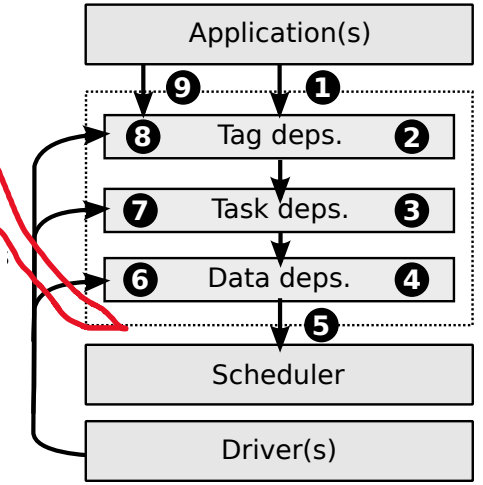
- Insertion automatique de tâches de partitionnement
  - ➢ Création des sous handles
  - ➢ Soumission des tâches de partitionnement
  - ➢ Exécution immédiate de la bulle



- Insertion automatique de tâches de recollement :
  - – Une tâche a ses dépendances satisfaites
  - – Des données sont partitionnées
  - ➢ Génération de tâches de recollement (récursif !)
  - ➢ Mise en attente de la tâche via une dépendance de tag

Comment automatiser les formules magiques ?

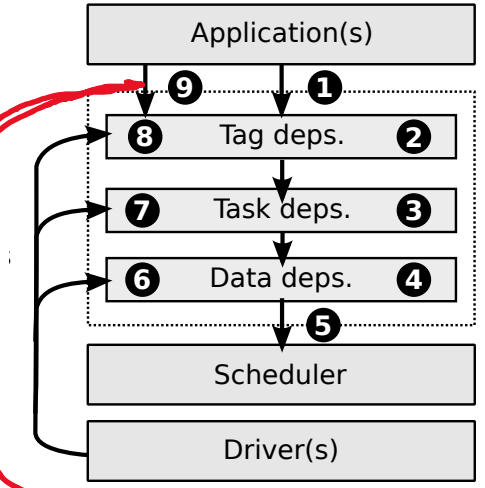
- Insertion automatique de tâches de partitionnement
  - Création des sous handles
  - Soumission des tâches de partitionnement
  - Exécution immédiate de la bulle



- Insertion automatique de tâches de recollement :
  - Une tâche a ses dépendances satisfaites
  - Des données sont partitionnées
  - Génération de tâches de recollement (récursif !)
  - Mise en attente de la tâche via une dépendance de tag

Comment automatiser les formules magiques ?

- Insertion automatique de tâches de partitionnement
  - ➢ Création des sous handles
  - ➢ Soumission des tâches de partitionnement
  - ➢ Exécution immédiate de la bulle



- Insertion automatique de tâches de recollement :
  - – Une tâche a ses dépendances satisfaites
  - – Des données sont partitionnées
  - ➢ Génération de tâches de recollement (récursif !)
  - ➢ Mise en attente de la tâche via une dépendance de tag

Travaux

- Consolider l'existant
- Interface utilisateur
  - Visualisation de DAG récursifs
- Solharis
  - Perf. Model
  - H-Mat, Chameleon : ordonnancement, structures hiérarchiques
  - QR-MUMPS : memory aware scheduling

- Solharis++
  - Tâches hiérarchiques distribuées