De l'abstraction des modèles de composants logiciels pour la programmation d'applications scientifiques distribuées

Hinde Lilia BOUZIANE

Équipe-projet PARIS
Centre INRIA Rennes - Bretagne Atlantique / IRISA
Université Rennes 1

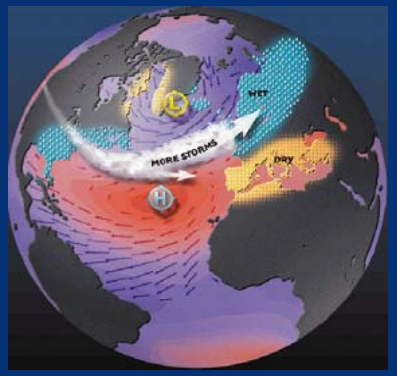
Encadrants : Thierry PRIOL & Christian PEREZ

Plan

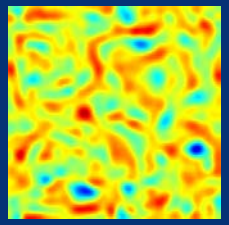
- Contexte et problématique
  - Calcul scientifique
  - Ressources matérielles
  - Problématique
- Analyse et objectifs
  - Limitations des modèles de composants
  - Objectifs
  - Méthodologie d'abstraction
- Contributions : modèles abstraits
  - Paradigme maître-travailleur
  - Assemblage spatio-temporel
  - Partage de données
- Conclusions et perspectives

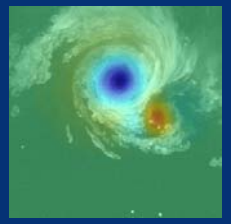
Le calcul scientifique

- Complexes
  - Taille
  - Hétérogénéité
    - Disciplines
    - Langages
  - Multi-paradigmes
- Gourmandes en puissance
  - Calcul
  - Stockage

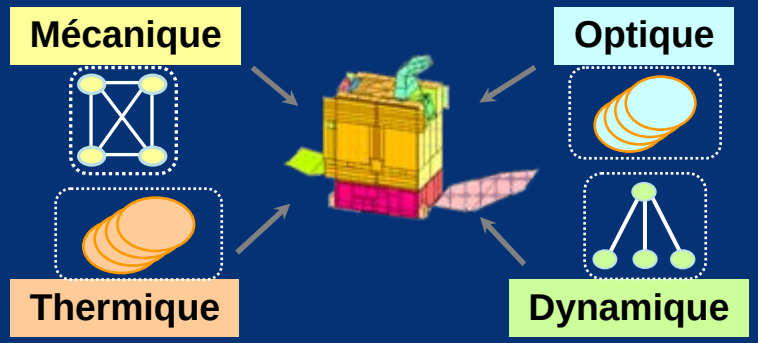


Climatologie : couplage océan-atmosphère





Traitement d'images : analyse de mouvements et trajectographie



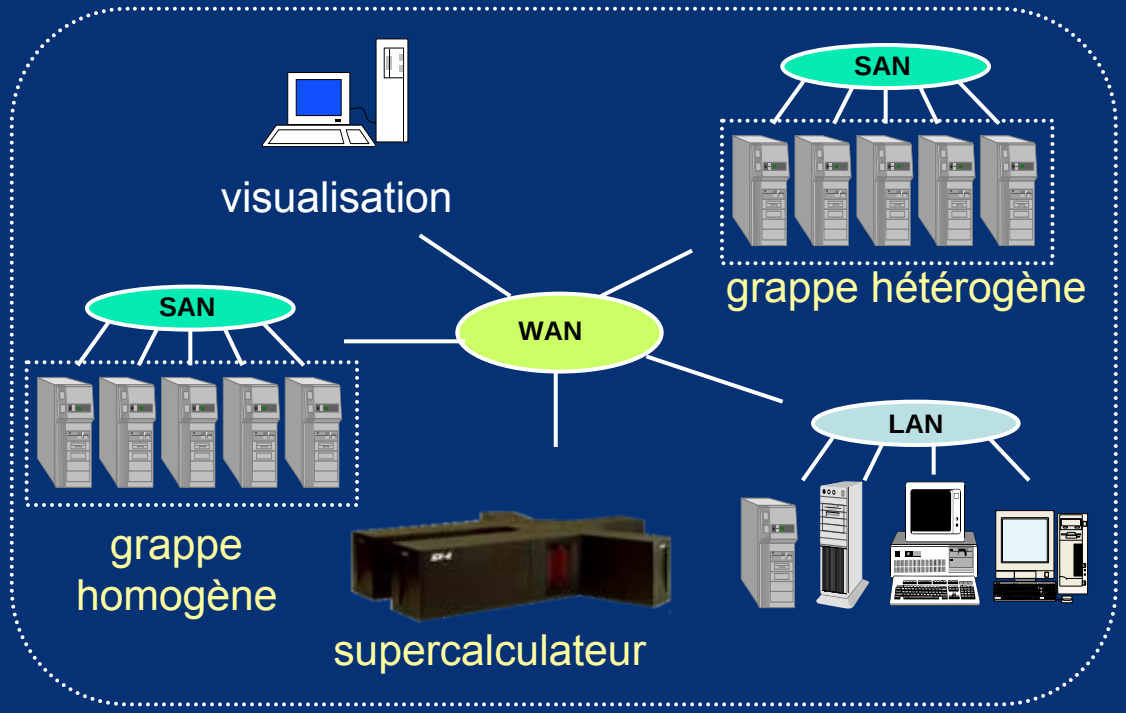
Conception satellite : multi-physique

Ressources matérielles

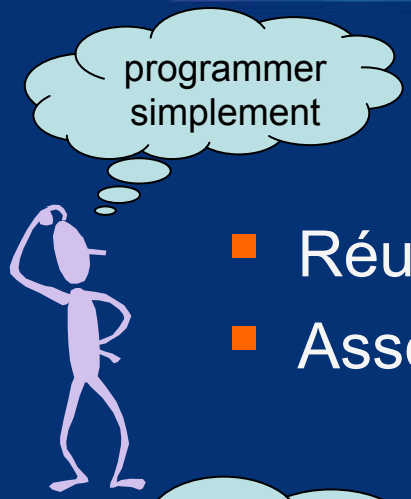
- Diverses
- Complexes
  - Nature distribuée/parallèle
  - Hétérogénéité
    - Matérielle
    - Logicielle (OS, …)
  - Volatilité
  - Partage
  - …

multi-cœurs

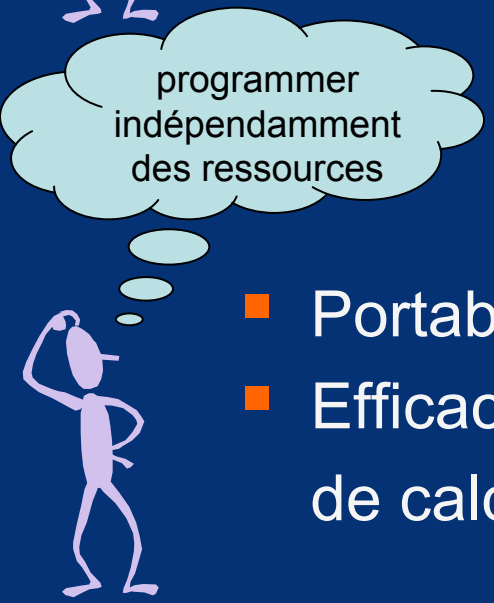
grille



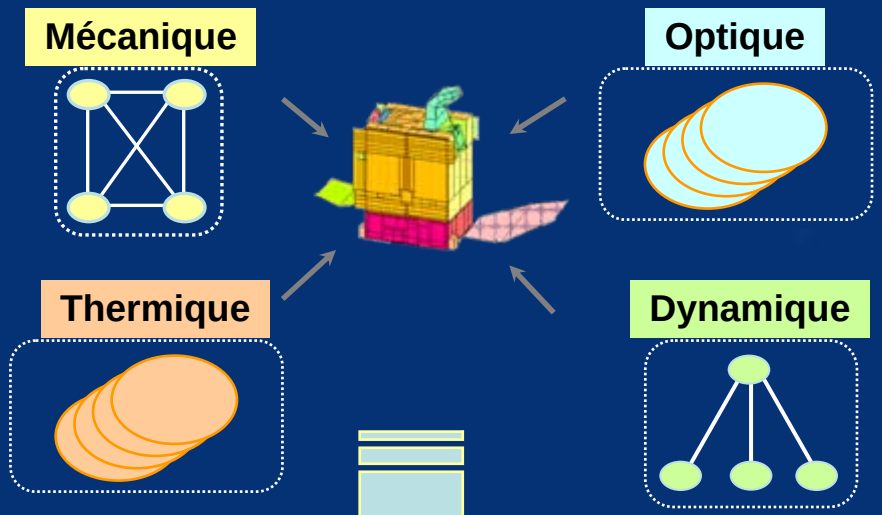
Problématique



- Réutilisation
- Assemblage



- Portabilité
- Efficacité de calcul



Modèles de composants logiciels

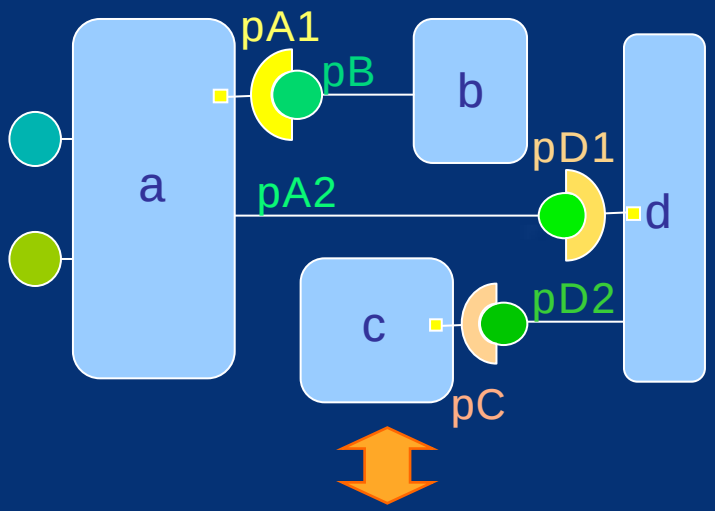
- Bibliographie
  - CCA – Common Component Architecture (CCA Forum)
    - Calcul haute performance
  - CCM – CORBA Component Model (OMG)
    - Applications distribuées hétérogènes
  - GCM – Grid Component Model (CoreGrid)
    - Basé sur Fractal
  - Fractal (France télécom & INRIA)
    - Système complexes (systèmes d'exploitation)
  - SCA – Service Component Architecture (OSOA group)
    - Applications orientées services
  - Grid.it / ASSIST (UNIPI)
  - …

Composant logiciel (1/2)

- Boite noire
- Ports
  - Invocation de méthodes (CCM, CCA, Fractal/GCM, SCA)
  - Événements (CCM)
  - Flux de données (Grid.it/ ASSIST)
  - Passage de messages (Darwin)
  - Passage de documents (SCA)
  - …
- Plusieurs implémentations
  - Différents algorithmes, langages, OS, processeurs, etc.

Assemblage

- Assemblage
  - Instances de composants et connexions
  - Architecture Description Language (ADL)
    - CCM, Fractal
  - Dynamique (API)
    - CCA, CCM, Fractal



ADL

```
instanceComp:
        a: A, b: B, c: C, d: D;
connections:
        a.pA1 <-> b.pB;
        c.pC <-> d.pD2;
        d.pD1 <-> a.pA2;
```

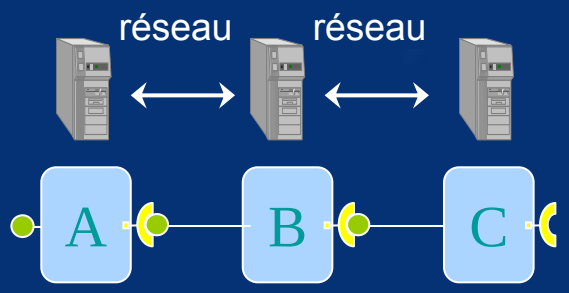
Plan

- Contexte et problématique
  - Calcul scientifique
  - Ressources matérielles
  - Problématique
- Analyse et objectifs
  - Limitations des modèles de composants
  - Objectifs
  - Méthodologie d'abstraction
- Contributions : modèles abstraits
  - Paradigme maître-travailleur
  - Assemblage spatio-temporel
  - Partage de données
- Conclusions et perspectives

Limitations de l'existant

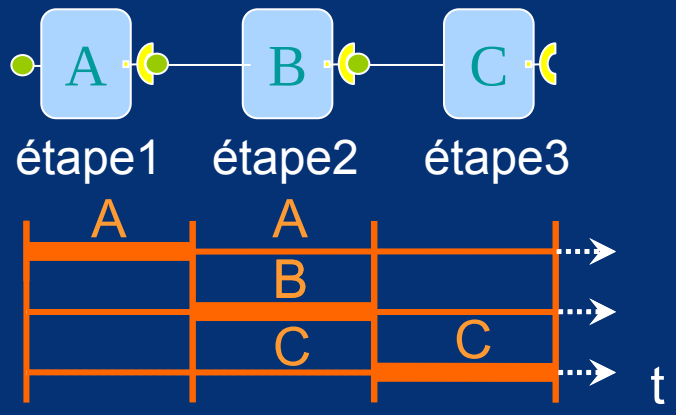
- Modèles d'assemblage proches du matériel
  - Assemblage dans l'espace

    - Dépendance vis-à-vis des ressources
    - Complexité de conception
      - Paradigmes parallèles (ex. maître-travailleur)



  - Comportement masqué dans l'assemblage

    - Surconsommation des ressources



s'exécutant

actif

Objectifs de la thèse

- Séparer la conception des applications des ressources
  - Portabilité
  - Pouvoir adapter une application à différents types de ressources (efficacité de calcul)
- Se concentrer sur les aspects métiers
  - Automatiser la gestion des aspects non-fonctionnels
- Modèle générique

Abstraction & modèles de composants

- Composant (suffisamment abstrait)
- Communications
  - Modèles à mémoire distribuée
    - Point à point : appel de méthodes, évènements
    - Communications parallèles
      - MxN : GCM, GridCCM (thèse André Ribes - PARIS)
    - À la pair à pair (destinataire inconnu) : Darwin
    - Passage de documents : SCA
  - Modèles à mémoire partagée
    - Actuellement masqué dans l'implémentation des composants
- Assemblage
  - Paradigme de programmation dans l'espace
    - Formes de parallélismes : maître-travailleur, diviser pour régner, décomposition géométrique, etc.
  - Paradigme de programmation dans le temps
    - Flot de données
    - Flux de travail (flot de contrôle et flot de données)

Abstraction & modèles de composants

- Composant (suffisamment abstrait)
- Communications
  - Modèles à mémoire distribuée
    - Point à point : appel de méthodes, évènements
    - Communications parallèles
      - MxN : GCM, GridCCM (thèse André Ribes - PARIS)
    - À la pair à pair (destinataire inconnu) : Darwin
    - Passage de documents : SCA
  - Modèles à mémoire partagée
    - Actuellement masqué dans l'implémentation des composants
- Assemblage
  - Paradigme de programmation dans l'espace
    - Formes de parallélismes : maître-travailleur, diviser pour régner, décomposition géométrique, etc.
  - Paradigme de programmation dans le temps
    - Flot de données
    - Flux de travail (flot de contrôle et flot de données)

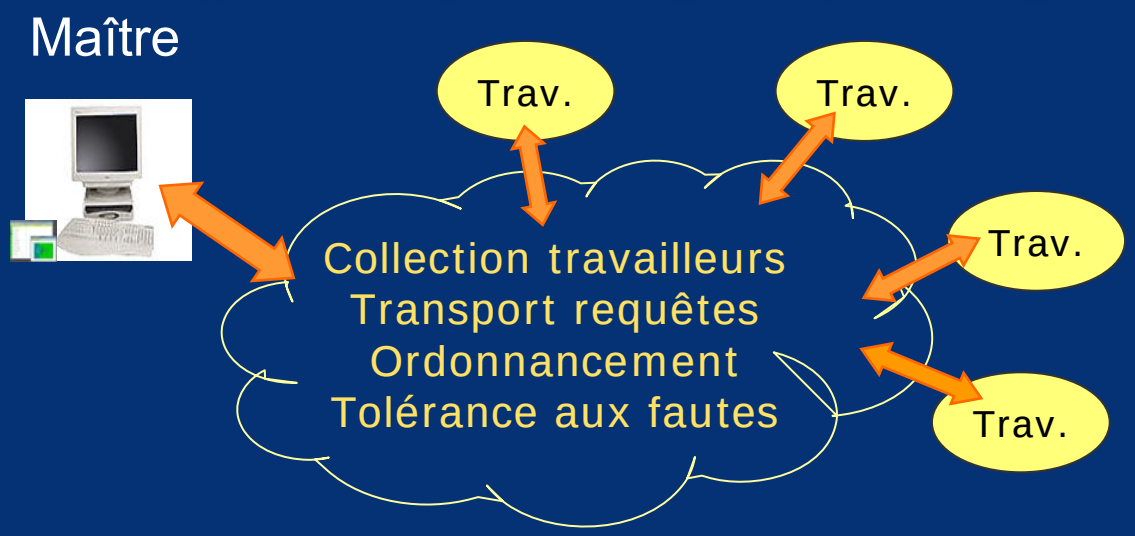
Plan

- Contexte et problématique
  - Calcul scientifique
  - Ressources matérielles
  - Problématique
- Analyse et objectifs
  - Limitations des modèles de composants
  - Objectifs
  - Méthodologie d'abstraction
- Contributions : modèles abstraits
  - Paradigme maître-travailleur
  - Assemblage spatio-temporel
  - Partage de données
- Conclusions et perspectives

Un modèle abstrait pour le paradigme maître-travailleur

- Paradigme maître-travailleur
- Limitations dans les modèles de composants existants
- Un modèle abstrait générique
- Évaluation

Paradigme maître-travailleur

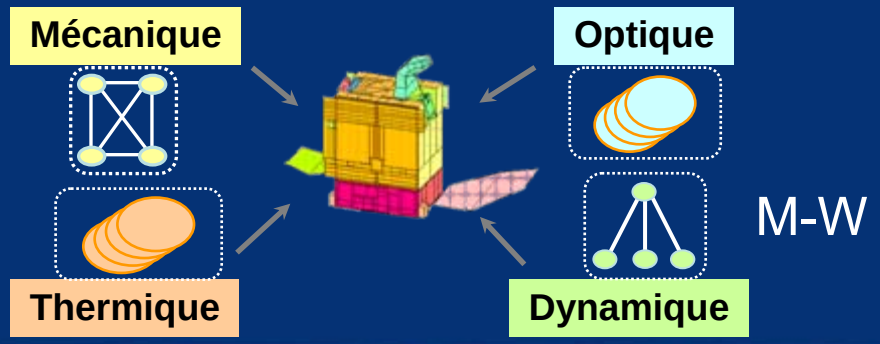


- Plusieurs calculs indépendants (boucle ~ForAll)
- Environnements/API dédiés
  - GridRPC : DIET, NetSolve, Ninf-G, …
  - *Desktop Grid* : BOINC, XtremWEB, …

Caractéristiques des environnements maître-travailleur

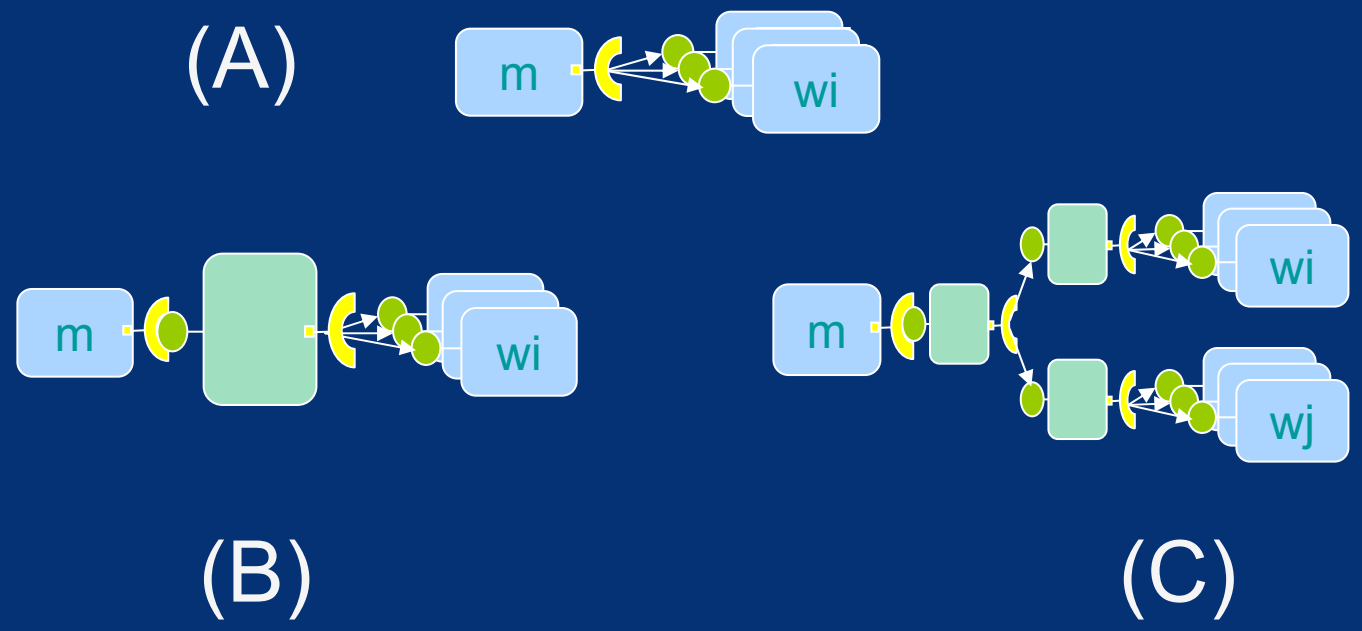
- Politiques avancées de transfert de requêtes
- Gestion transparente des aspects non-fonctionnels

- APIs dédiées
- Paradigmes de programmation limités



Assemblage d'une application maître-travailleur dans les modèles de composants

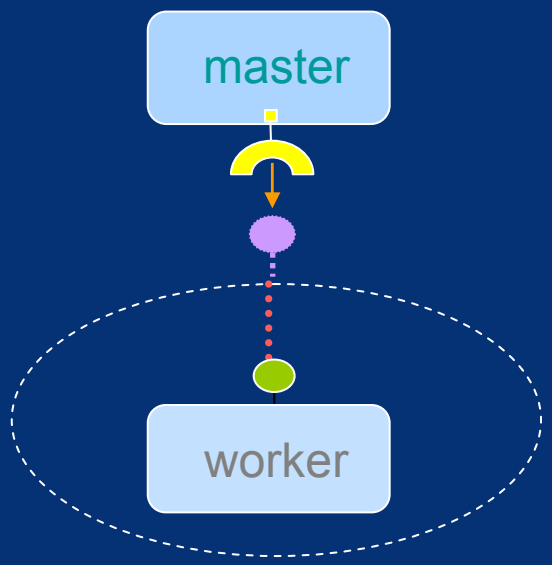
- Considérations non fonctionnelles
- Dépendances vis-à-vis des ressources



...

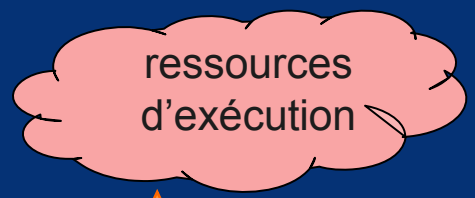
Idée générale

Vue concepteur



Collection
Assemblage
abstrait

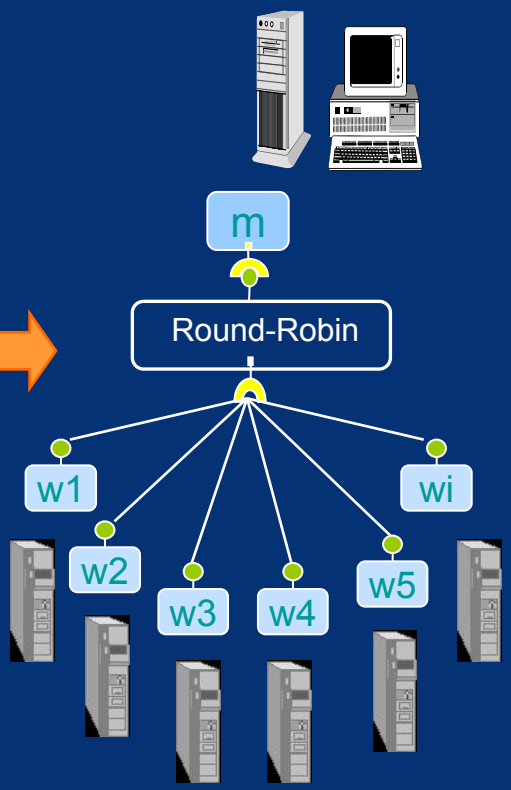
Vue système/plateforme



travailleurs
+
sélection patron

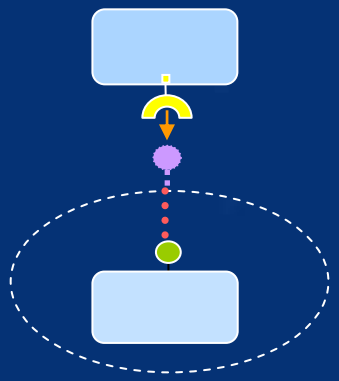
liste de patrons de transport de requêtes

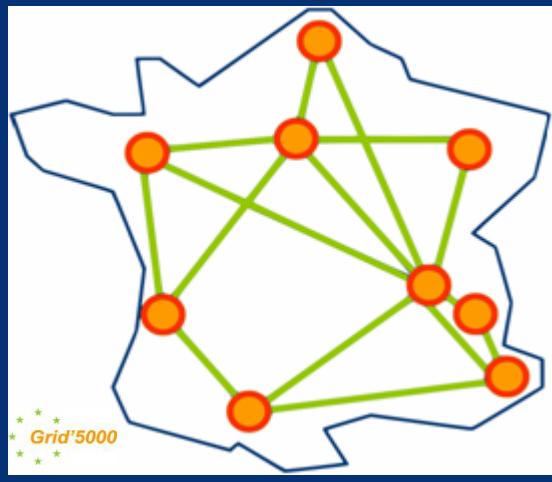
1. *Random*
2. *Round-Robin*
3. NetSolve
4. Diet



Évaluation

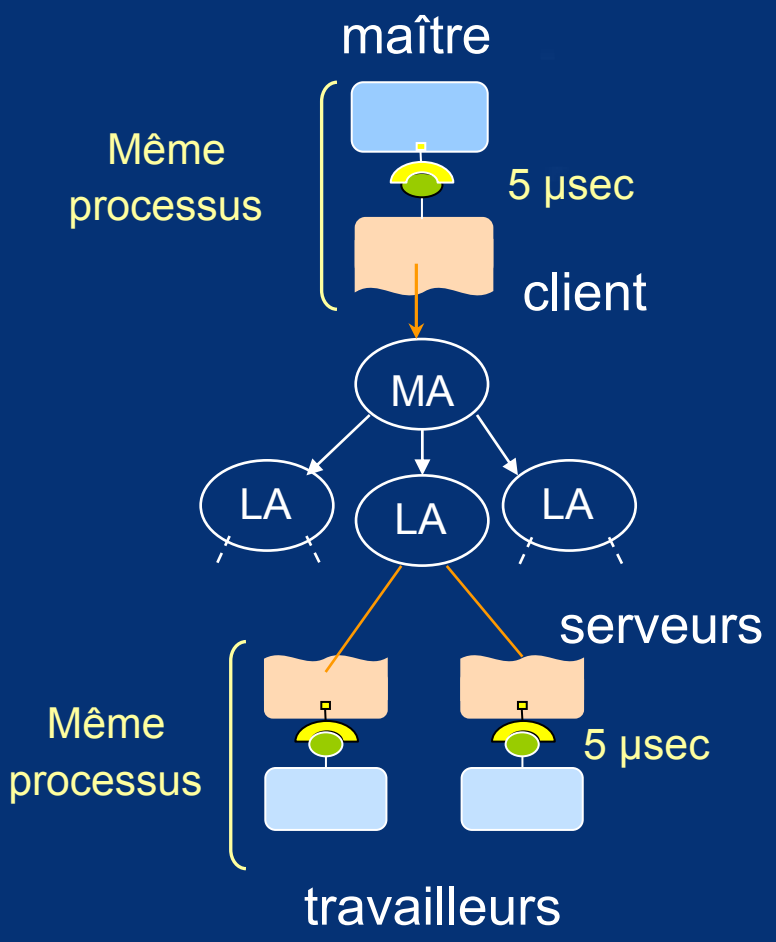
- Projection sur CCM
- Application synthétique
  - Paramétrique
  - Variation du type et taille des requêtes (temps, données)
- Implémentation des composants
  - OmniORB 4.1.0
- Ressources
  - Grappe : Rennes
    - AMD Opteron 246, 2Ghz
    - Gigabit Ethernet
  - Grid'5000 : 7 sites (jusqu'à 1025 processeurs)
    - 10 Gbit/sec
- Déploiement avec ADAGE (PARIS)





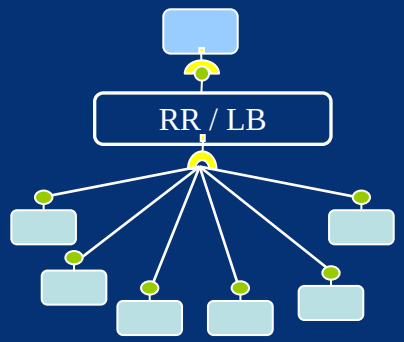
Surcoûts d'utilisation de DIET

- DIET (GRAAL - ENS/LYON)
  - Env. hiérarchique pour grille
- DIET comme patron
  - Composants adaptateurs
- Implémentation DIET
  - CORBA
- Comparaison avec utilisation classique
- Surcoûts d'envoi d'une requête
  - 1 travailleur : 10 µsec



Performances sur grille

- Politiques d'ordonnancement
  - Tourniquet (RR)
  - Équilibrage de charge (LB)
  - Séquencement de requêtes (SR)
- Répartition sur 7 sites



1024 trav.	4096 req. Homogènes (20 sec)	
	temps (s)	Eff. (%)
RR	**80.11**	**99.87**
LB	**80.12**	**99.85**

1024 trav.	4096 req hétérogènes (]5 sec – 20 sec[)	
	temps (s)	Eff. (%)
RR	**72.75**	**68.61**
LB	**62.28**	**80.13**

128 trav. grille	512 req. de 20 sec + 1 req. de 100s	
	temps (s)	Eff. (%)
LB	**180.01**	**44.9**
DIET/SR	**100.82**	**80.1**

128 trav. grappe	256 req. Homogènes (500 msec)	
	temps (msec)	Eff. (%)
RR	**1004**	**99.70**
DIET/SR	**1737**	**57.60**

Résumé

- Abstraction du paradigme maître-travailleur
  - Collection
    - Simplicité de conception
  - Patrons de transferts de requêtes
    - Moyen d'adaptation à différents contextes d'exécution
- Résultats expérimentaux prometteurs
  - Projection sur CCM
- Projection aussi sur Fractal/GCM et CCA

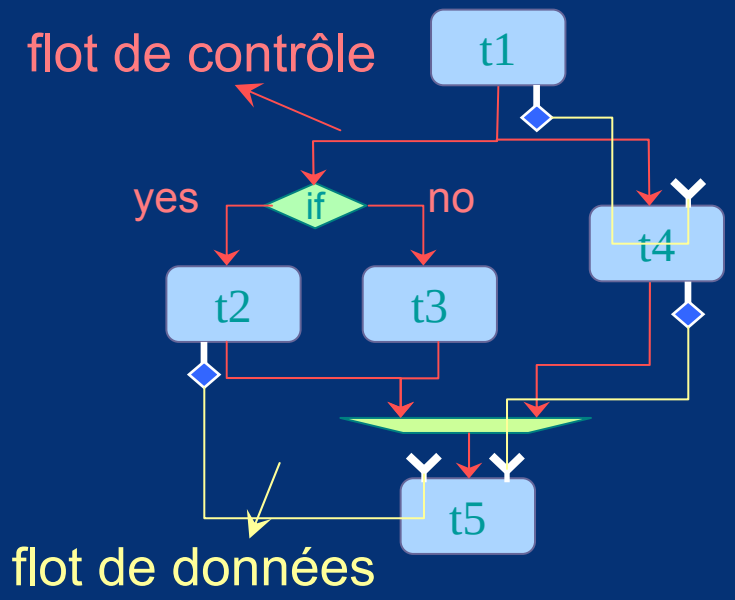
STCM : un modèle pour la composition spatio-temporel

- Modèles de flux de travail
- Limitations des approches de combinaison des dimensions spatiale et temporelle dans l'existant
- Le modèle STCM

Modèles de flux de travail

- Bibliographie
  - Askalon-AGWL (Abstract Grid Wrokflow Language) - Innsbruck, Autriche
  - Triana - Cardiff, UK
  - GriCoL (Grid Concurrent Language) - HLRS Stuttgart, Allemagne
  - Kepler - SEEK, SDM, GEON, …

- Modèle générique
  - Tâche
  - Ports
    - Données en entrée/sortie
  - Assemblage
    - Flot de contrôle / données



Modèles de flux de travail vs. modèles de composants

	assemblage	simplicité	couplage de codes	utilisation des ressources
Flux de travail	+	+	-	+
Comp. logiciels	+	+	+	-

Approches existantes pour combiner les dimensions spatiale et temporelle

- Modèles de composants
  - Meta-informations décrivant le comportement d'une implémentation d'un composant (ICENI)
  - Objectif : optimiser le placement des composants au déploiement

  Connaissance du code
  Complexité de conception

- Modèles de flux de travail
  - Encapsulation d'une composition spatiale dans l'implémentation d'une tâche
  - Objectif: permettre le couplage de code

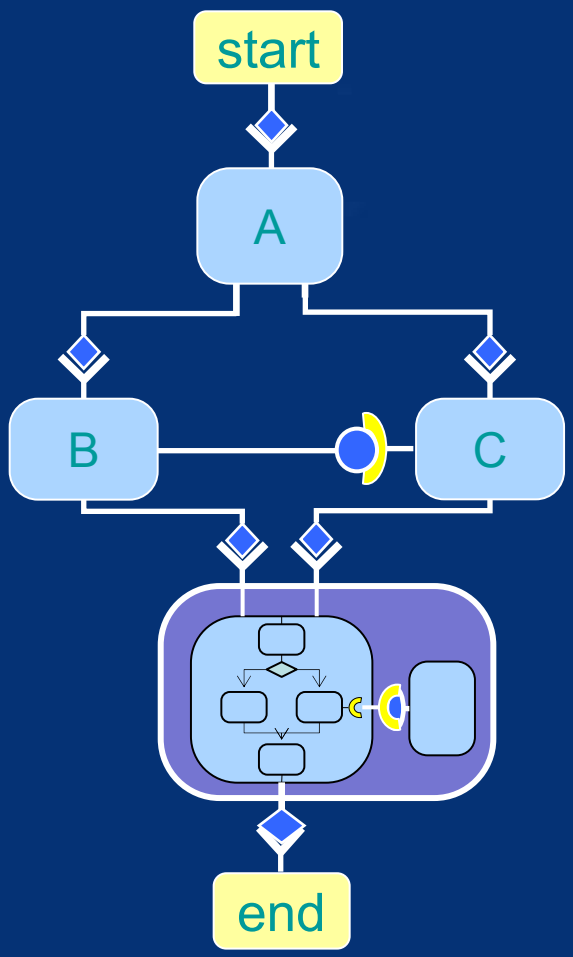
  Limitation de la hiérarchie d'un assemblage à 2 niveaux
  Limitation de la réutilisation

Conclusion

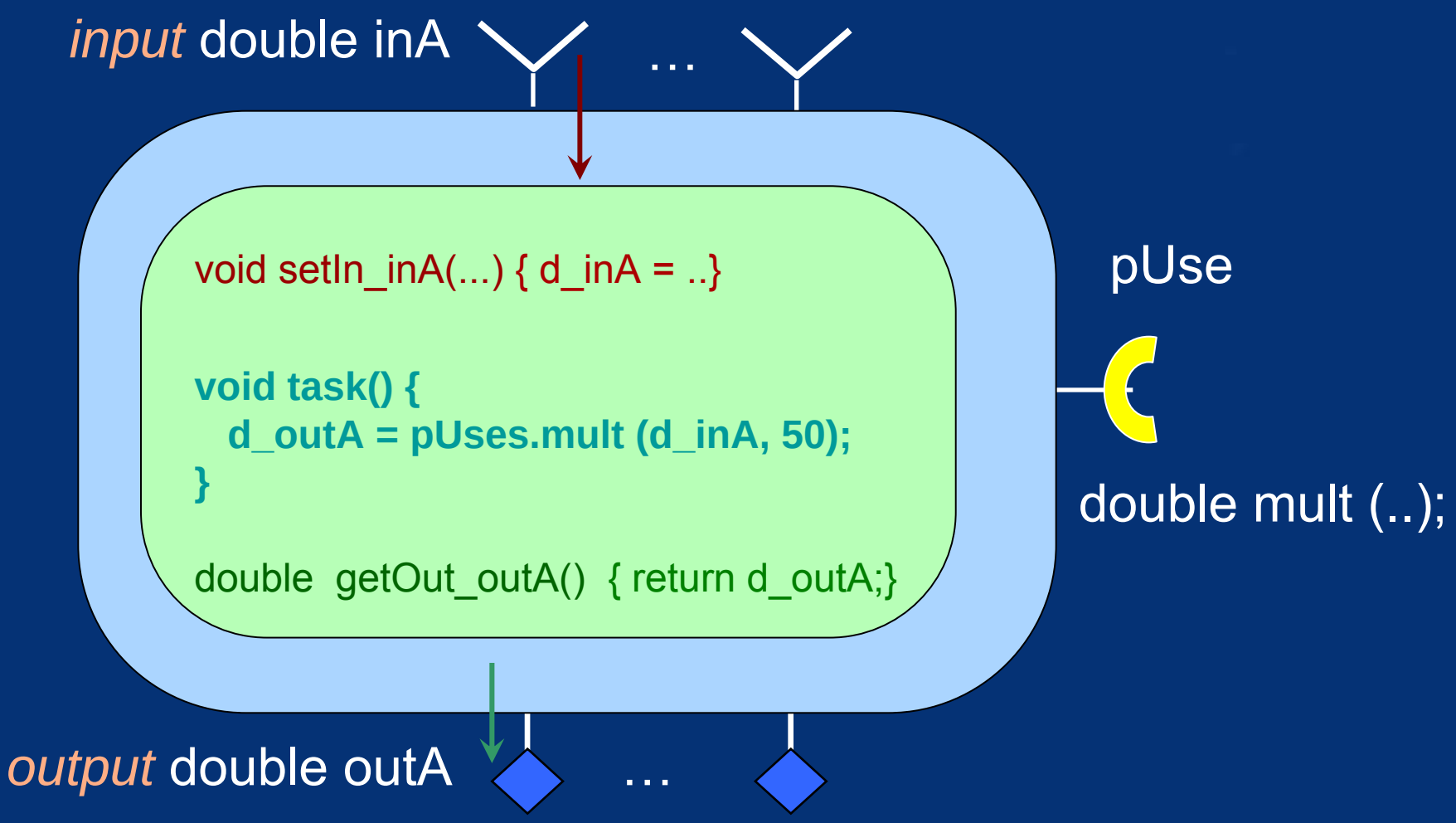
Limitations dues à différents niveaux de considération des deux dimensions

Idée générale

- Combiner les modèles de composants avec les modèles de flux de travail
  - Dimensions spatiale et temporelle au même niveau d'assemblage
- Composant-tâche
  - Ports d'entrée/sortie (temporels)
  - Ports spatiaux
  - Tâche
- Modèle d'assemblage
  - Adaptation d'un langage de flux de travail

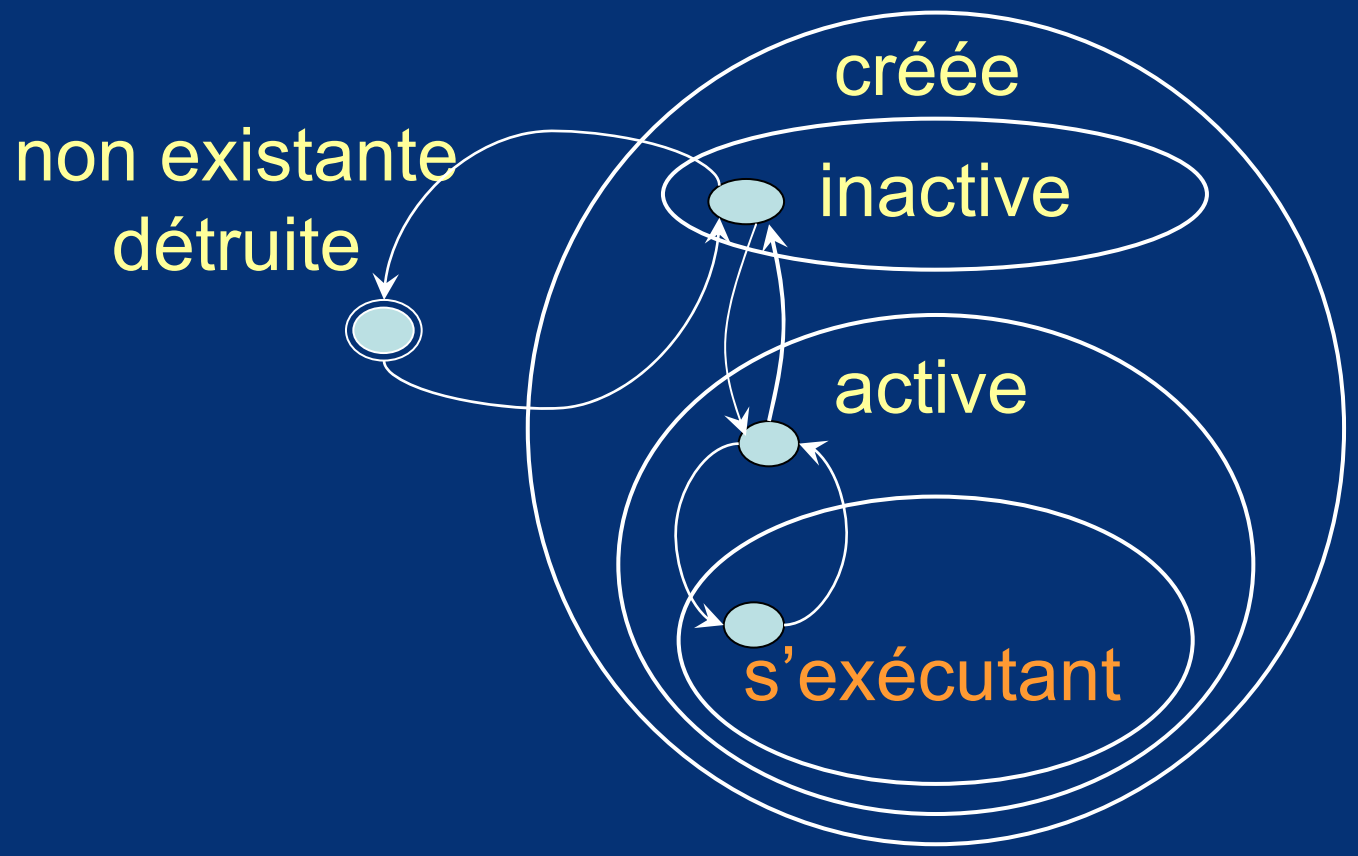


Ports temporels & tâche



Cycle de vie

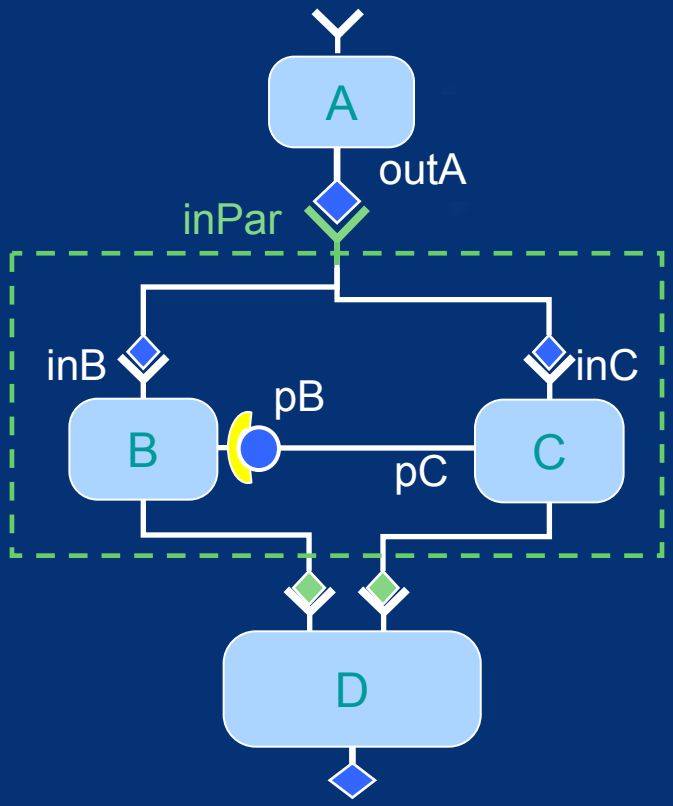
- États d'une instance de composant à l'exécution



Modèle d'assemblage

```
component Example {
  …
  parallel parCtrl {
    dataIn Double inPar <= a.outA;
    component B { dataIn Double inB;
                  clientPort Compute pB;
     … };
     component C { dataIn Double inC;
                   serverPort Compute pC;
     … };

     instance B b;
     instance C c;
     connect b.inB  to parCtrl.inPar;
     connect c.inC to parCtrl.inPar;
     connect b.pB to c.pC;
    // instructions
    section : exectask (a);
    section : exectask (b);
  } // fin parallel
  …
}
```



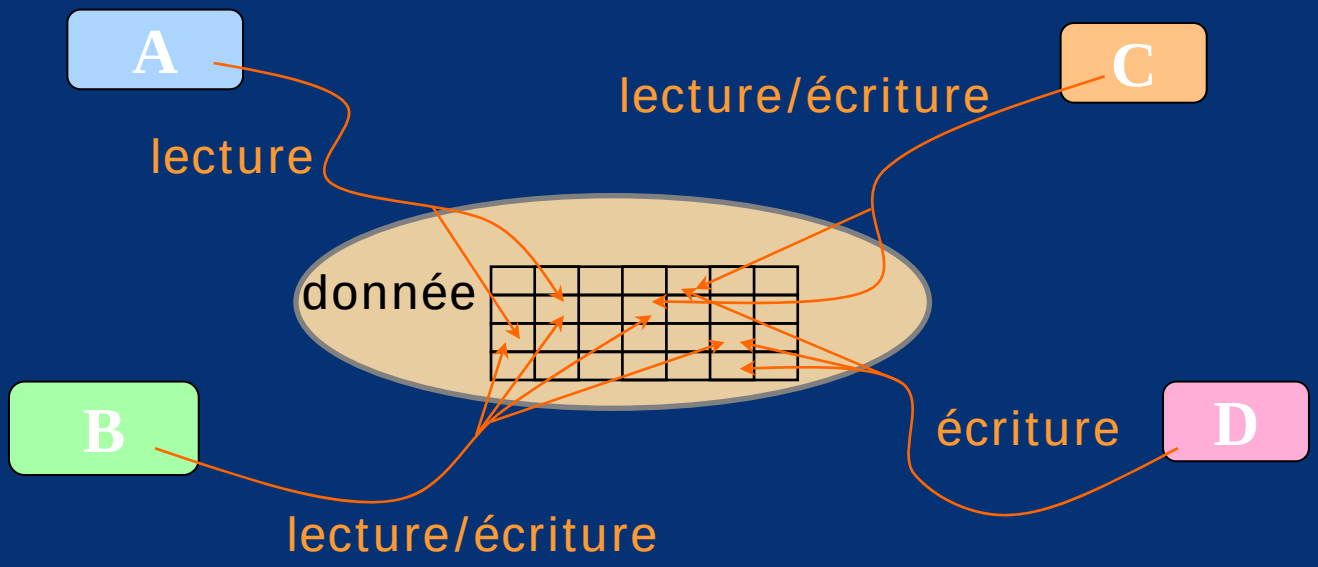
Résumé

- Union des modèles de composants et flux de travail
  - Composant-tâche
  - Ports temporels
  - Modèle d'assemblage à la flux de travail
- STCM
  - Extension de GCM
    - Ports temporels et tâche
  - Adaptation d'AGWL
    - Composant et composition spatiale

Un modèle abstrait pour le partage de données

- Partage de données
- Vue globale du modèle

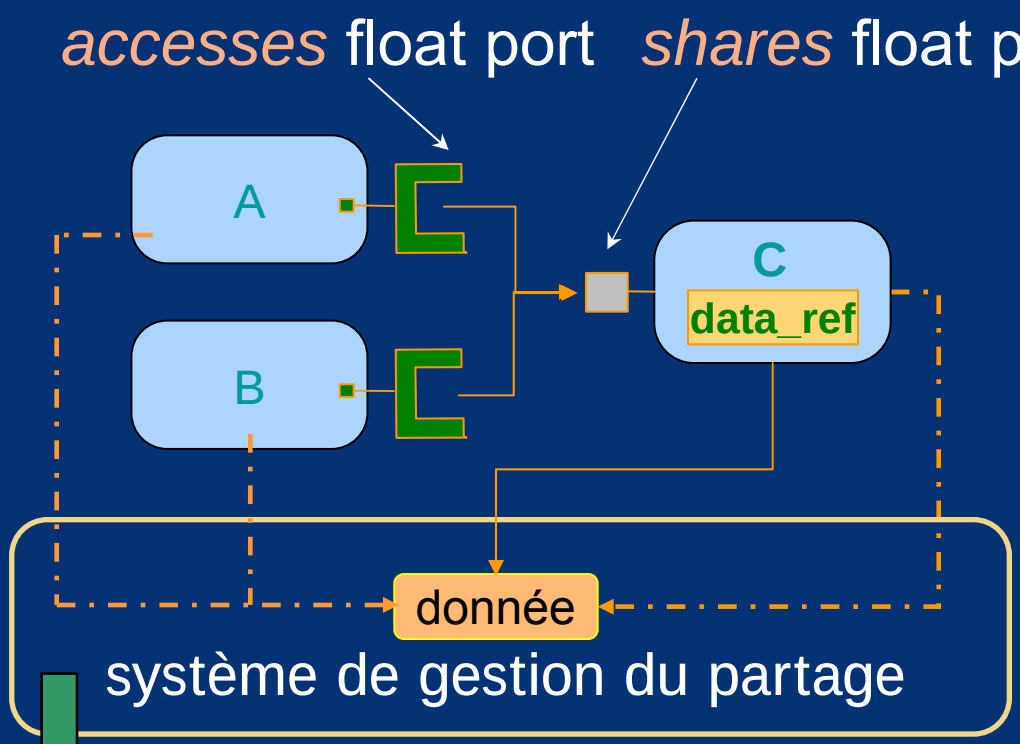
Le partage de données



- Plusieurs accès concurrents à une donnée
- Gestion de la localisation et de la concurrence
  - Intra-machine : système d'exploitation
  - Intra-grappe : mémoire partagée distribuée (DSM)
  - Intra-grille : service de partage de données (JuxMem/PARIS)

Vue globale du modèle

- Principe : accès transparent à une donnée




```
interface SharesPort {
  float* allocate_space(in ling size);
  void free_space();
  float* get_pointer();
  long get_size ();
  void acquire();
  void release ();
  void acquire_release();
};
```

- Choisi en fonction des ressources et placement
  - Système d'exploitation, DSM, JuxMem

Plan

- Contexte et problématique
  - Calcul scientifique
  - Ressources matérielles
  - Problématique
- Analyse et objectifs
  - Limitations des modèles de composants
  - Objectifs
  - Méthodologie d'abstraction
- Contributions : modèles abstraits
  - Paradigme maître-travailleur
  - Assemblage spatio-temporel
  - Partage de données
- Conclusions et perspectives

Conclusions

- Séparer la programmation des applications des ressources
- Abstraction des modèles de composants
  - Communications
    - Partage de données
  - Paradigme de programmation dans l'espace
    - Maître-travailleur
  - Paradigme de programmation dans le temps
    - STCM
- Projection sur plusieurs modèles de composants
  - GCM, CCA, CCM

Perspectives

- Spécification MDA
- Adaptation des paradigmes de programmation
  - Dynamicité du paradigme maître-travailleur
    - Stage de M2 (Guillaume Gauvrit)
- Finalisation de STCM
  - Sémantique formelle
  - Implémentation STCM
- Interactions entre modèles
  - Thèse en cours (Julien Bigot)
- Applications
  - Projet ANR LEGO (climatologie, cosmologie)