L'environnement EPSN

M. Dussere, A. Esnard, N. Richart, O. Coulaud

INRIA - Institut National de Recherche en Informatique et Automatisme
unité de recherche Futurs Bordeaux

October 4, 2005

Plan

Approche impérative

L'approche impérative est basée sur des actions prédéfinies.

- Dirigée par la simulation
- Commandes de steering dans le code la simulation
- Préparation explicite de messages et envois systématiques

Pros & Cons

- Très peu flexible
- Toute modification nécessite une nouvelle instrumentation
- Instrumentation simple

Approche impérative

L'approche impérative est basée sur des actions prédéfinies.

- Dirigée par la simulation
- Commandes de steering dans le code la simulation
- Préparation explicite de messages et envois systématiques

Pros & Cons

- Très peu flexible
- Toute modification nécessite une nouvelle instrumentation
- Instrumentation simple

Approche descriptive

L'approche descriptive est basée sur les propriété de la simulations qui sont révélée par l'instrumentation. C'était l'approche d'Epsilon.

- Dirigée par le client
- Définition des données d'interaction
- Définition de *droits d'accès* dans l'instrumentation

Pros & Cons

- Flexible
- Toute modification nécessite une nouvelle instrumentation
- L'instrumentation devient très complexe quand on prend en compte des simulations parallèles

Approche descriptive

L'approche descriptive est basée sur les propriété de la simulations qui sont révélée par l'instrumentation. C'était l'approche d'Epsilon.

- Dirigée par le client
- Définition des données d'interaction
- Définition de *droits d'accès* dans l'instrumentation

Pros & Cons

- Flexible
- Toute modification nécessite une nouvelle instrumentation
- L'instrumentation devient très complexe quand on prend en compte des simulations parallèles

Approche générique

L'approche générique utilisée dans EPSN est une approche descriptive où une partie de la description est dissociée de l'instrumentation

- Dissociation du squelette et des autres informations
- Description externe combinant le squelette et les autres informations
- Description du squelette de la simulation dans l'instrumentation

Pros & Cons

- Flexible
- Les modifications de la description externe ne nécessitent pas une nouvelle instrumentation
- L'instrumentation reste simple
- La généricité permet la gestion du parallélisme et d'opération de haut niveau

Approche générique

L'approche générique utilisée dans EPSN est une approche descriptive où une partie de la description est dissociée de l'instrumentation

- Dissociation du squelette et des autres informations
- Description externe combinant le squelette et les autres informations
- Description du squelette de la simulation dans l'instrumentation

Pros & Cons

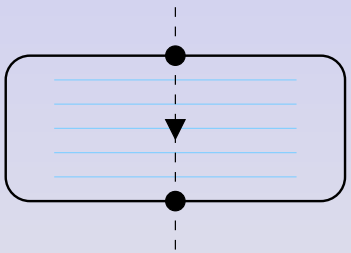
- Flexible
- Les modifications de la description externe ne nécessitent pas une nouvelle instrumentation
- L'instrumentation reste simple
- La généricité permet la gestion du parallélisme et d'opération de haut niveau

Plan

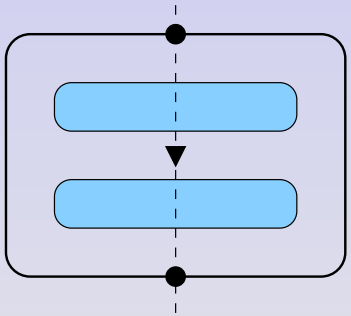
Graphe de Taches Hiérarchiques - HTG

HTG

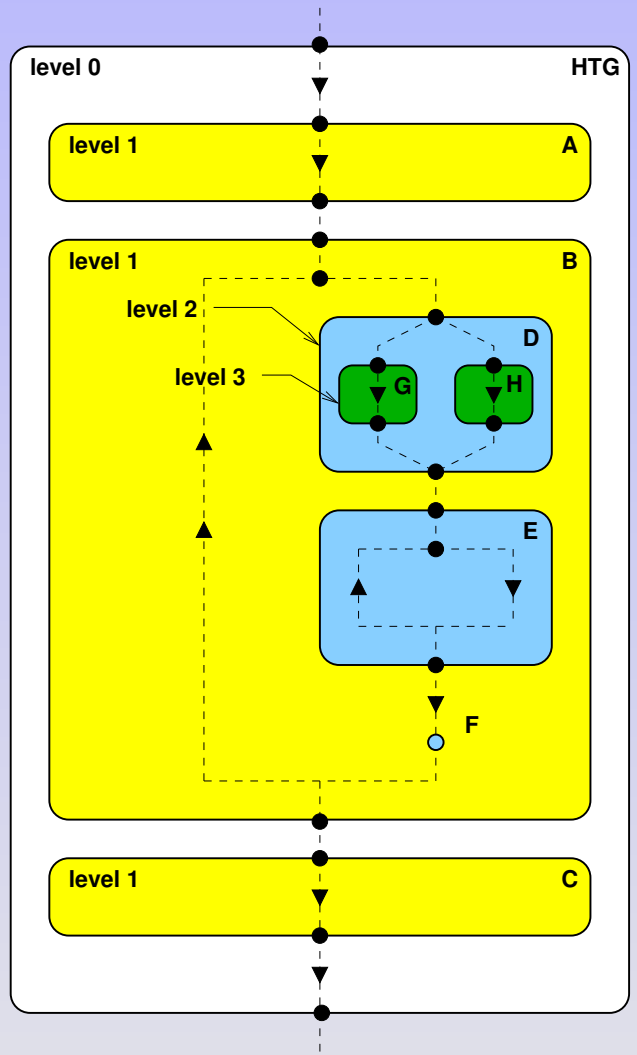
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



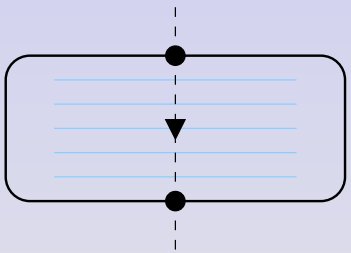
tâche composée



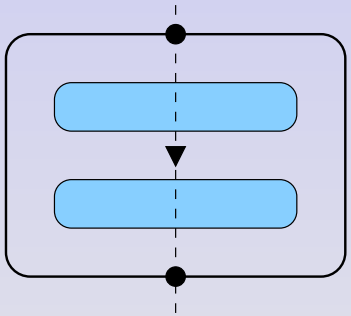
Graphe de Taches Hiérarchiques - HTG

HTG

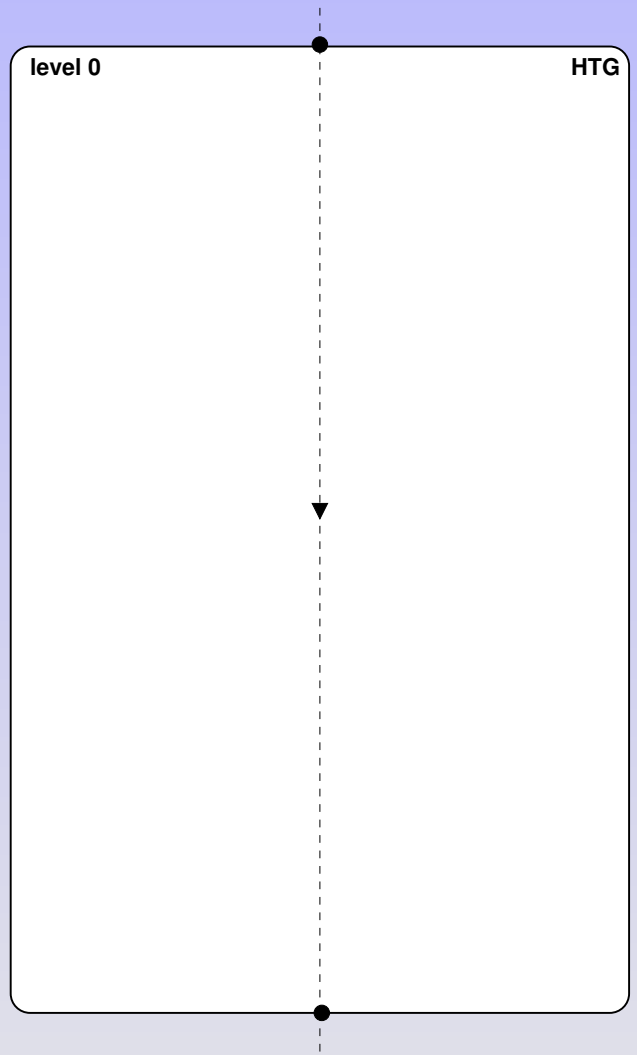
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



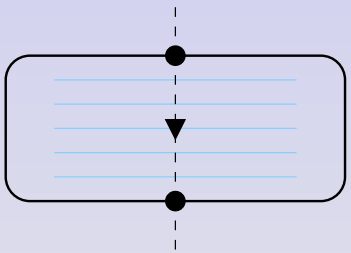
tâche composée



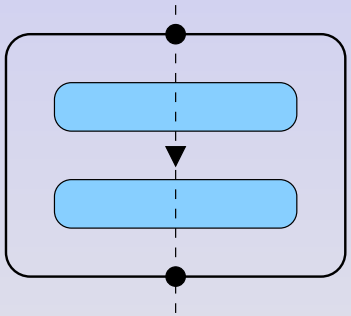
Graphe de Taches Hiérarchiques - HTG

HTG

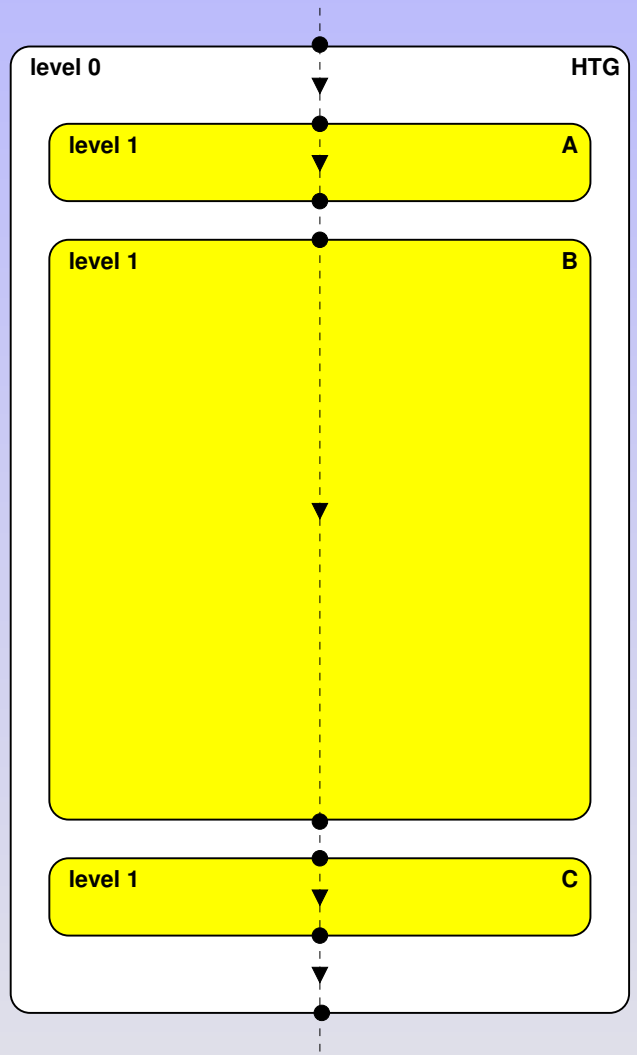
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



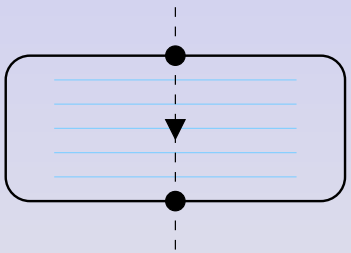
tâche composée



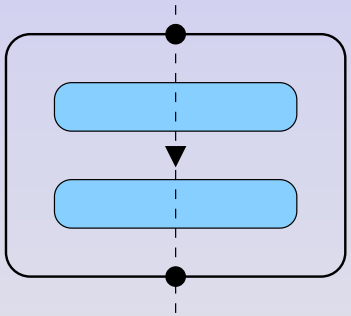
Graphe de Taches Hiérarchiques - HTG

HTG

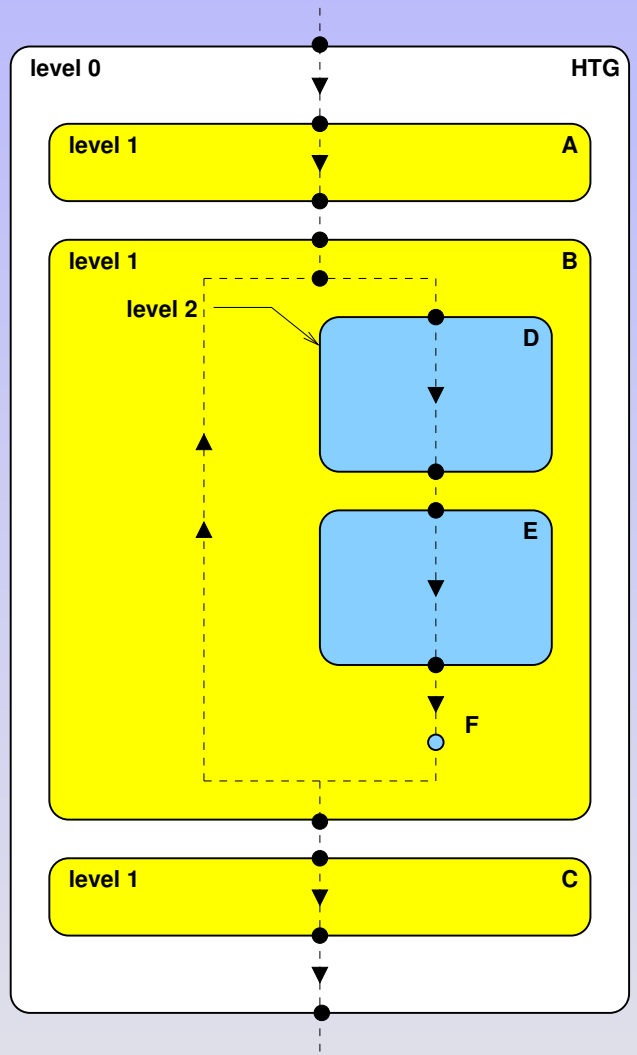
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



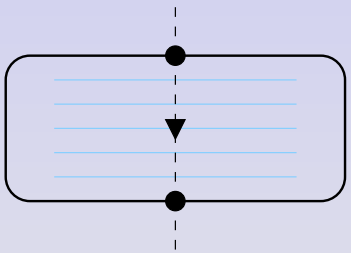
tâche composée



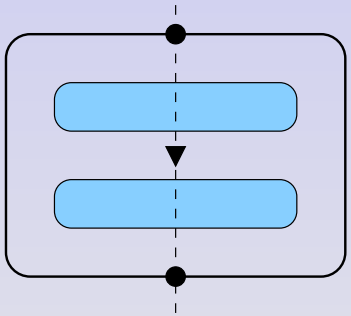
Graphe de Taches Hiérarchiques - HTG

HTG

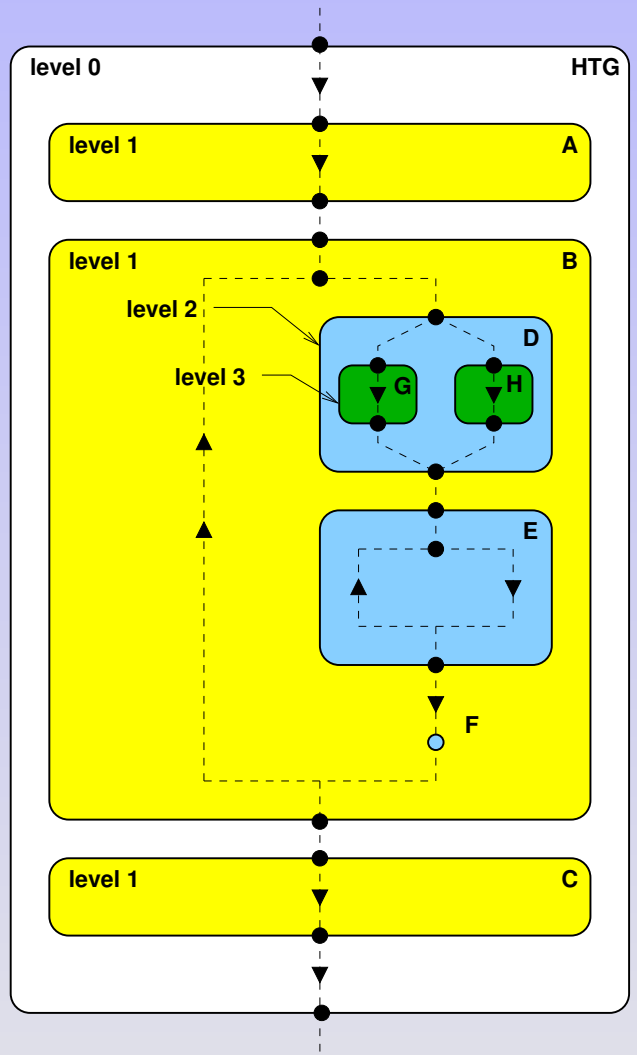
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



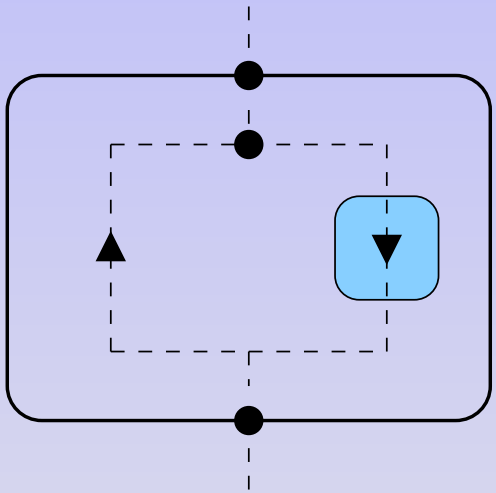
tâche composée



Les tâches en boucles

Propriétés

- Tache hiérarchique contenant une seule sous-tache
- La sous-tache peut être parcouru plusieurs fois
- Garantit le caractère acyclique du graphe
- Modélise les structures itératives: `do`, `for`, `while`
- Ne peut modéliser les renvois type `goto`

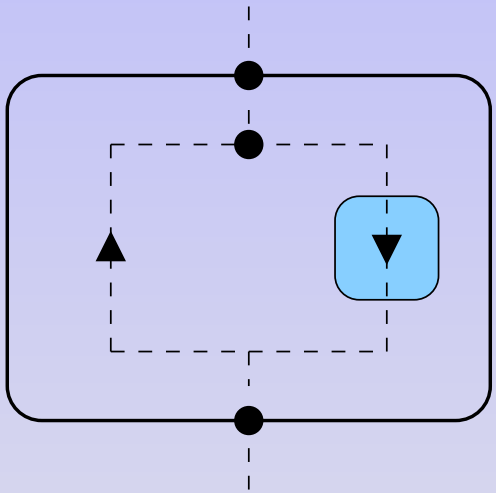


tâche en boucle

Les tâches en boucles

Propriétés

- Tache hiérarchique contenant une seule sous-tache
- La sous-tache peut être parcouru plusieurs fois
- Garantit le caractère acyclique du graphe
- Modélise les structures itératives: `do`, `for`, `while`
- Ne peut modéliser les renvois type `goto`

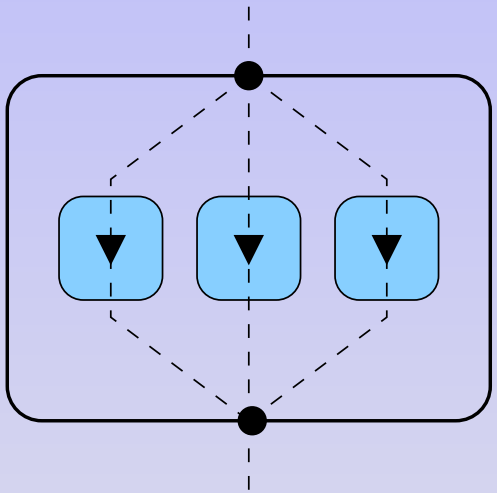


tâche en boucle

Les switch

Propriétés

- Tache hiérarchique contenant une ou plusieurs sous-taches
- Les sous-taches s'excluent mutuellement
- Modélise les structures conditionnelles: `if/then/else`, `switch/case`



tâche conditionnelle

Le HTG dans les simulations parallèles

Principe

- Tous les processus suivent le même HTG
- les points d'instrumentation ne sont pas synchronisants
- La simulation est dans un état cohérent si :
  - tous les processus sont sur le même point du HTG
  - (pour une boucle) tous les processus sont sur la même itération
  - (pour un switch) tous les processus sont sur la même branche
  - les parents de la tache courante présentent la même cohérence.

Propriétés pour le parallélisme

- **acyclique**
  $\Rightarrow$ *datation* précise de l'évolution des processus
- **hiérarchique**
  $\Rightarrow$ robustesse face à des différences d'exécution mineures

Le HTG dans les simulations parallèles

Principe

- Tous les processus suivent le même HTG
- les points d'instrumentation ne sont pas synchronisants
- La simulation est dans un état cohérent si :
  - tous les processus sont sur le même point du HTG
  - (pour une boucle) tous les processus sont sur la même itération
  - (pour un switch) tous les processus sont sur la même branche
  - les parents de la tache courante présentent la même cohérence.

Propriétés pour le parallélisme

- **acyclique**
  $\Rightarrow$ *datation* précise de l'évolution des processus
- **hiérarchique**
  $\Rightarrow$ robustesse face à des différences d'exécution mineures

La description des données

Description minimale

- Identifiant:
  lien avec la bibliothèque de redistribution
- Classe:
  scalaire, champs, particules, maillage . . .
- Localisation:
  - localisé sur un processus
  - répliqué sur tous les processus
  - partagé entre les processus

Plan

Pilotage par les requêtes

Principe

- La simulation joue le rôle de serveur en attente de requêtes clientes
- La simulation peut être connectée à plusieurs clients
- Les interactions sont des réponses à des requêtes client
- Les communications et les synchronisations sont des réponses aux requêtes client
- Pas de requêtes $\Rightarrow$ Pas de surcoût

Pilotage par les requêtes

Principe

- La simulation joue le rôle de serveur en attente de requêtes clientes
- La simulation peut être connectée à plusieurs clients
- Les interactions sont des réponses à des requêtes client
- Les communications et les synchronisations sont des réponses aux requêtes client
- Pas de requêtes $\Rightarrow$ Pas de surcoût

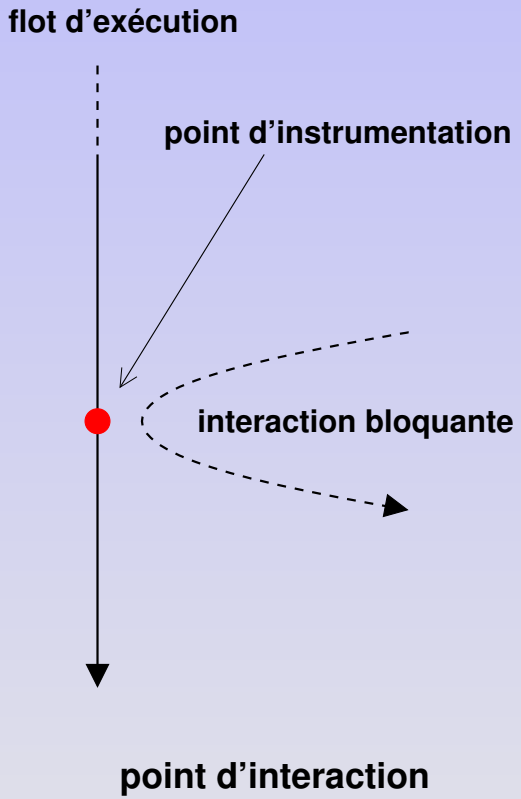
Points et plages d'interaction

Point d'interaction

- Contrôle du flux d'exécution
- Support d'interactions bloquantes
- Correspond à un point d'instrumentation: début/fin de tache, tache ponctuelle

Plage d'interaction

- Support d'interactions concurrentes à la simulation
- Délimitée par 2 points d'instrumentation
- Nous ne considérons que les plages correspondant aux tâches
- $\Rightarrow$ Permet du recouvrement



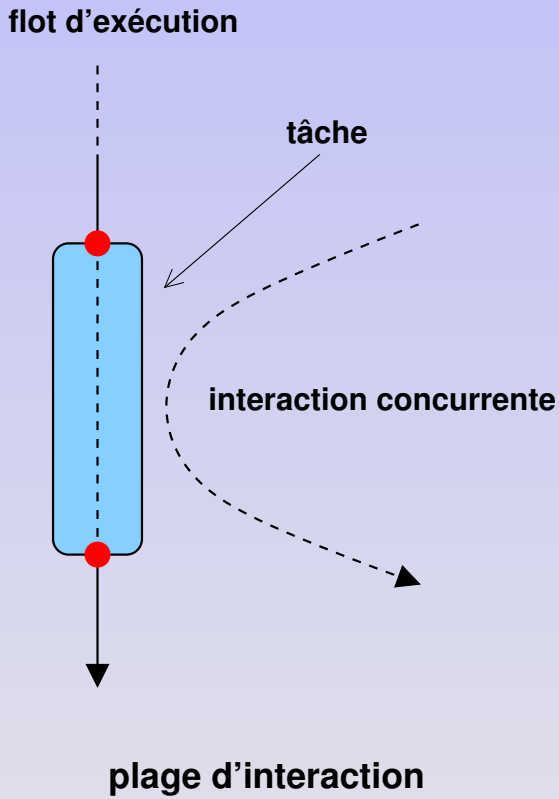
Points et plages d'interaction

Point d'interaction

- Contrôle du flux d'exécution
- Support d'interactions bloquantes
- Correspond à un point d'instrumentation: début/fin de tache, tache ponctuelle

Plage d'interaction

- Support d'interactions concurrentes à la simulation
- Délimitée par 2 points d'instrumentation
- Nous ne considérons que les plages correspondant aux tâches
- $\Rightarrow$ Permet du recouvrement



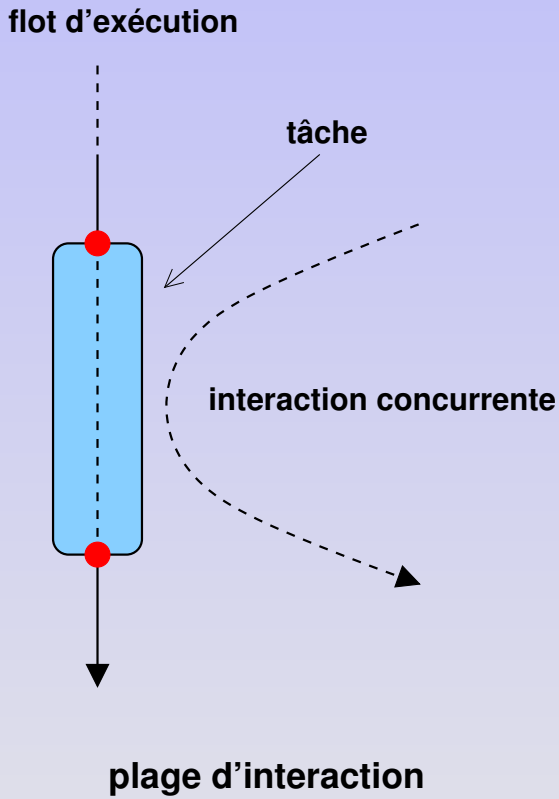
Points et plages d'interaction

Point d'interaction

- Contrôle du flux d'exécution
- Support d'interactions bloquantes
- Correspond à un point d'instrumentation: début/fin de tache, tache ponctuelle

Plage d'interaction

- Support d'interactions concurrentes à la simulation
- Délimitée par 2 points d'instrumentation
- Nous ne considérons que les plages correspondant aux tâches
- $\Rightarrow$ Permet du recouvrement



Cohérence des traitements parallèles

Planification des traitements

- Chaque traitement est planifié pour être exécuté à un moment précis de la simulation
- Exécution indépendante des traitements sur chaque processus
- → Peu coûteux: pas de communications entre processus
- → Non synchronisant

Problème

Comment déterminer une date pour la planification ?

Cohérence des traitements parallèles

Planification des traitements

- Chaque traitement est planifié pour être exécuté à un moment précis de la simulation
- Exécution indépendante des traitements sur chaque processus
- $\longrightarrow$ Peu coûteux: pas de communications entre processus
- $\longrightarrow$ Non synchronisant

Problème

Comment déterminer une date pour la planification ?

Cohérence des traitements parallèles

Planification des traitements

- Chaque traitement est planifié pour être exécuté à un moment précis de la simulation
- Exécution indépendante des traitements sur chaque processus
- $\longrightarrow$ Peu coûteux: pas de communications entre processus
- $\longrightarrow$ Non synchronisant

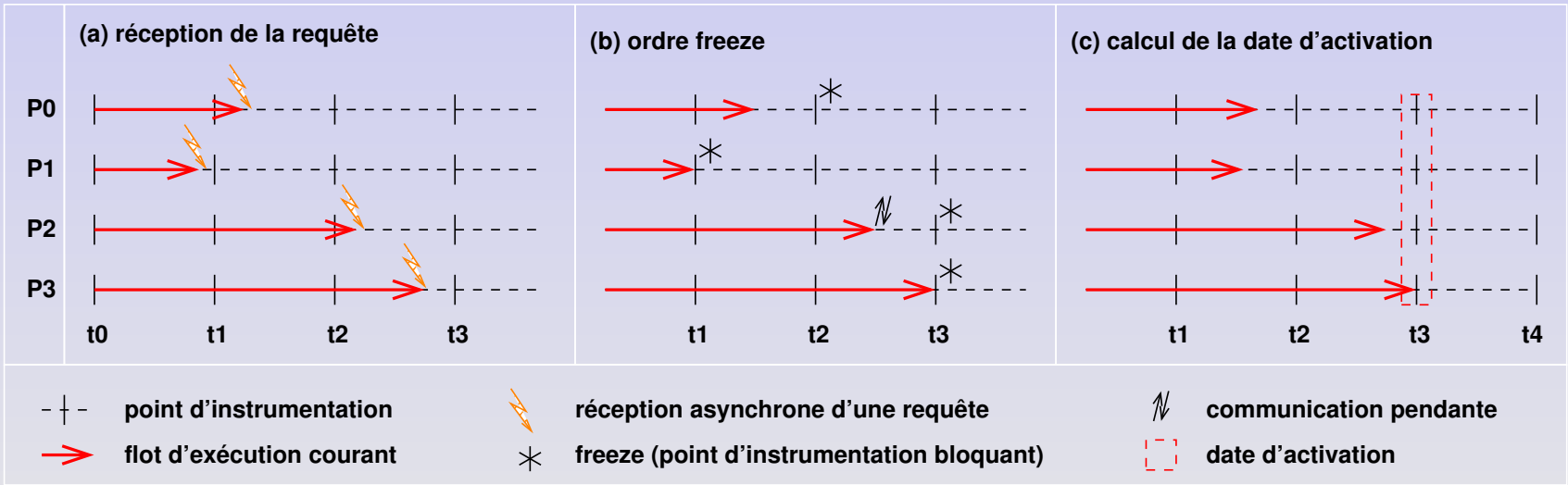
Problème

Comment déterminer une date pour la planification ?

Planification à la volée

Principe

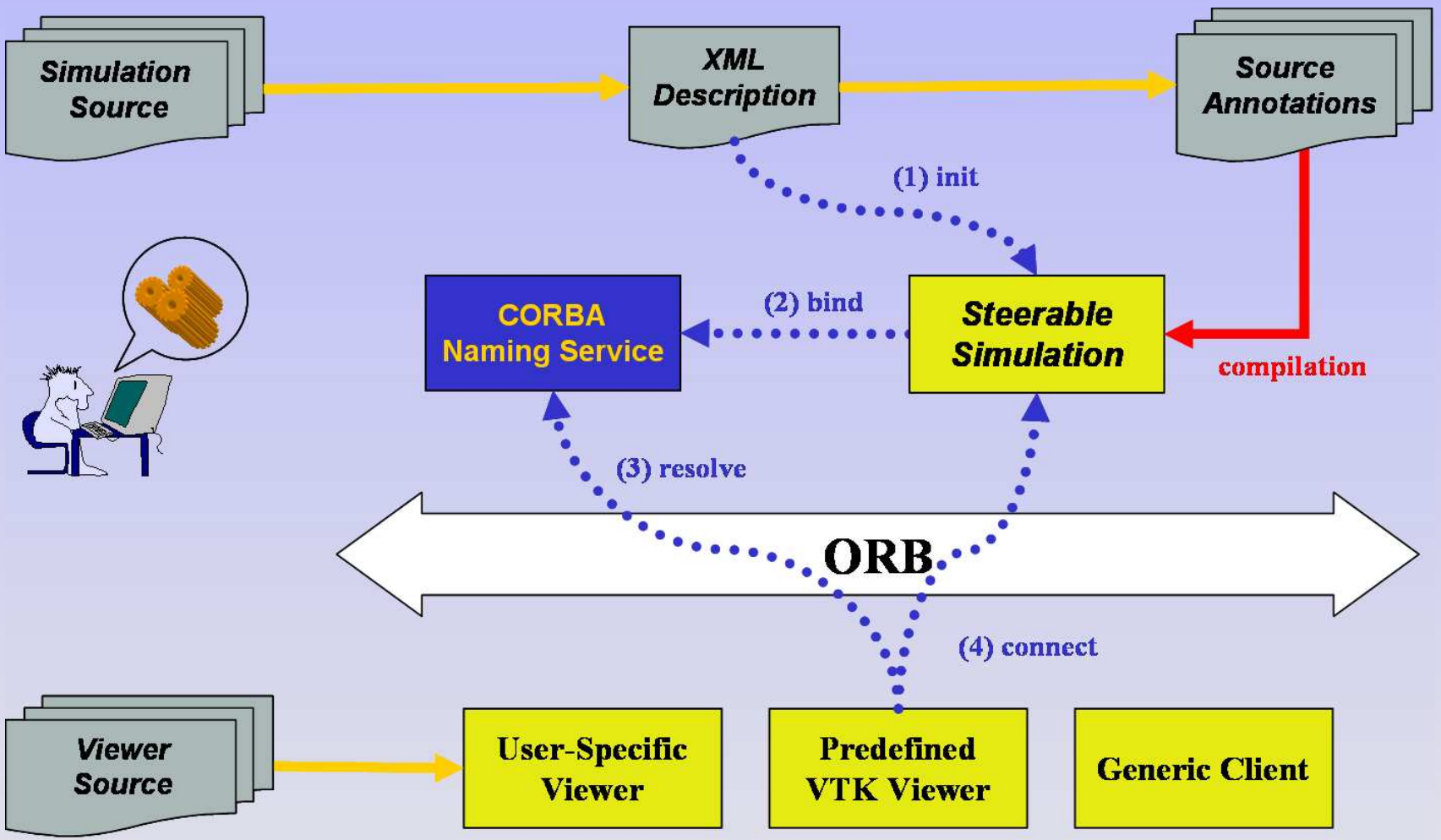
- Déterminer une date commune qu'aucun processus n'a encore dépassé
- Ajouter un critère de validité pour le début du traitement
- Comme la simulation est SPMD on obtient une date de planification valide



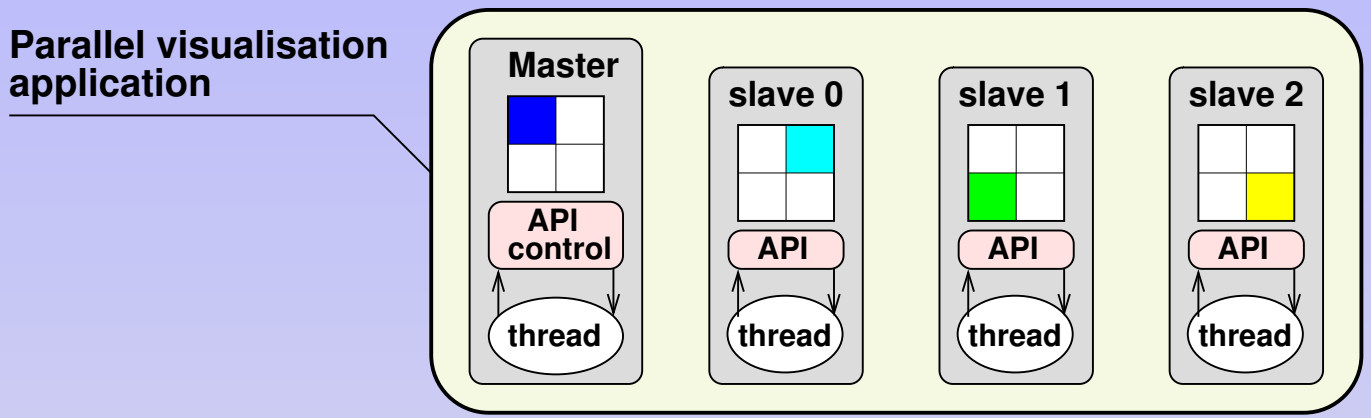
Plan

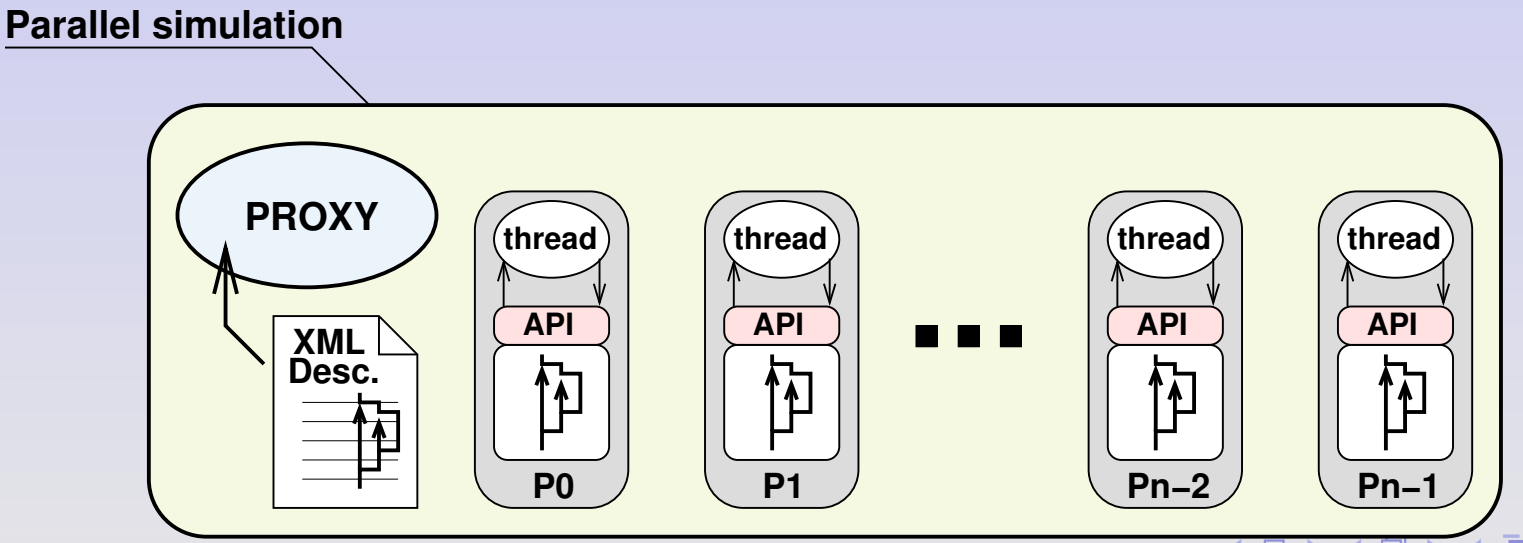
Plan

L'intégration

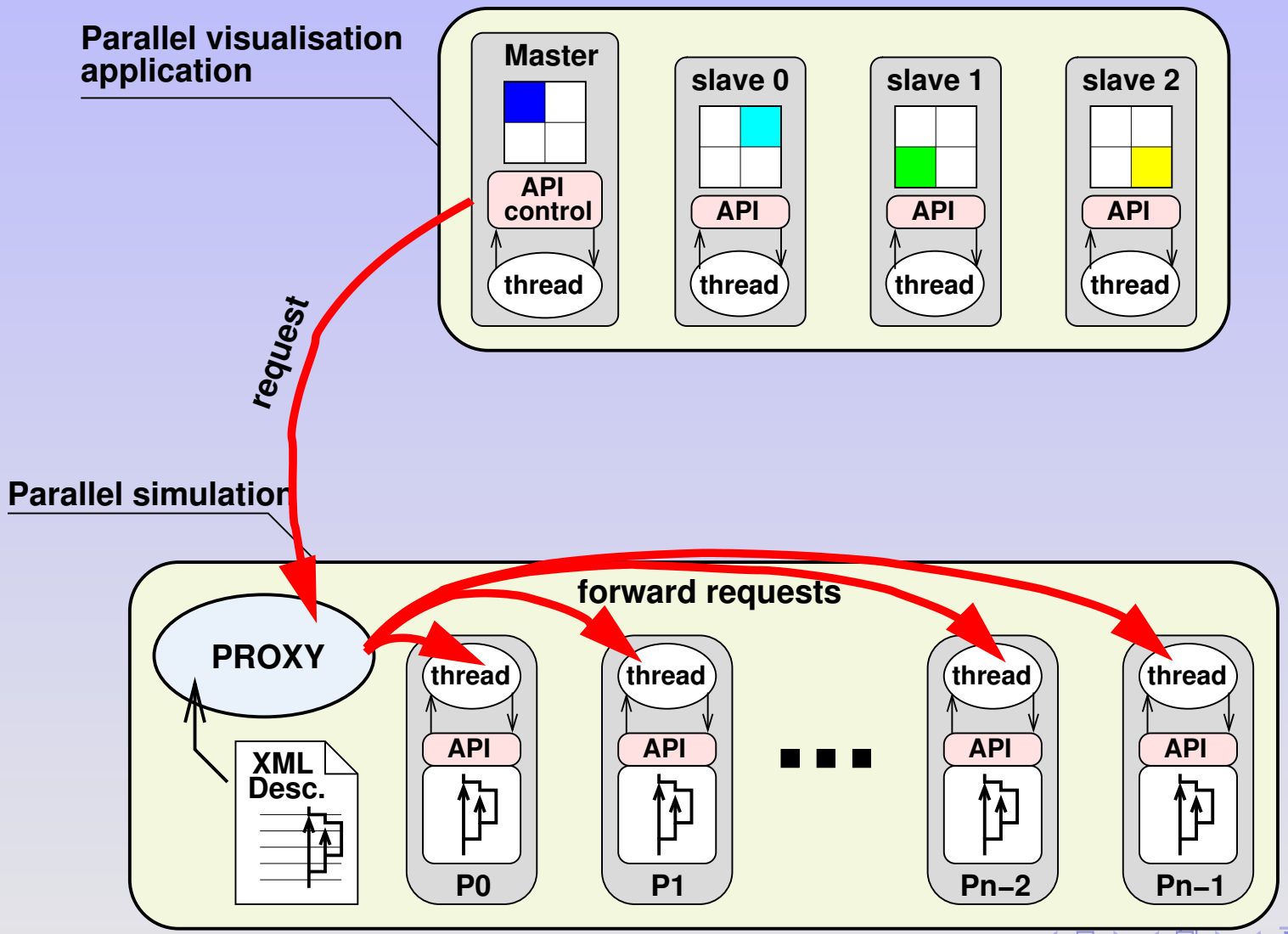


Architecture distribuée

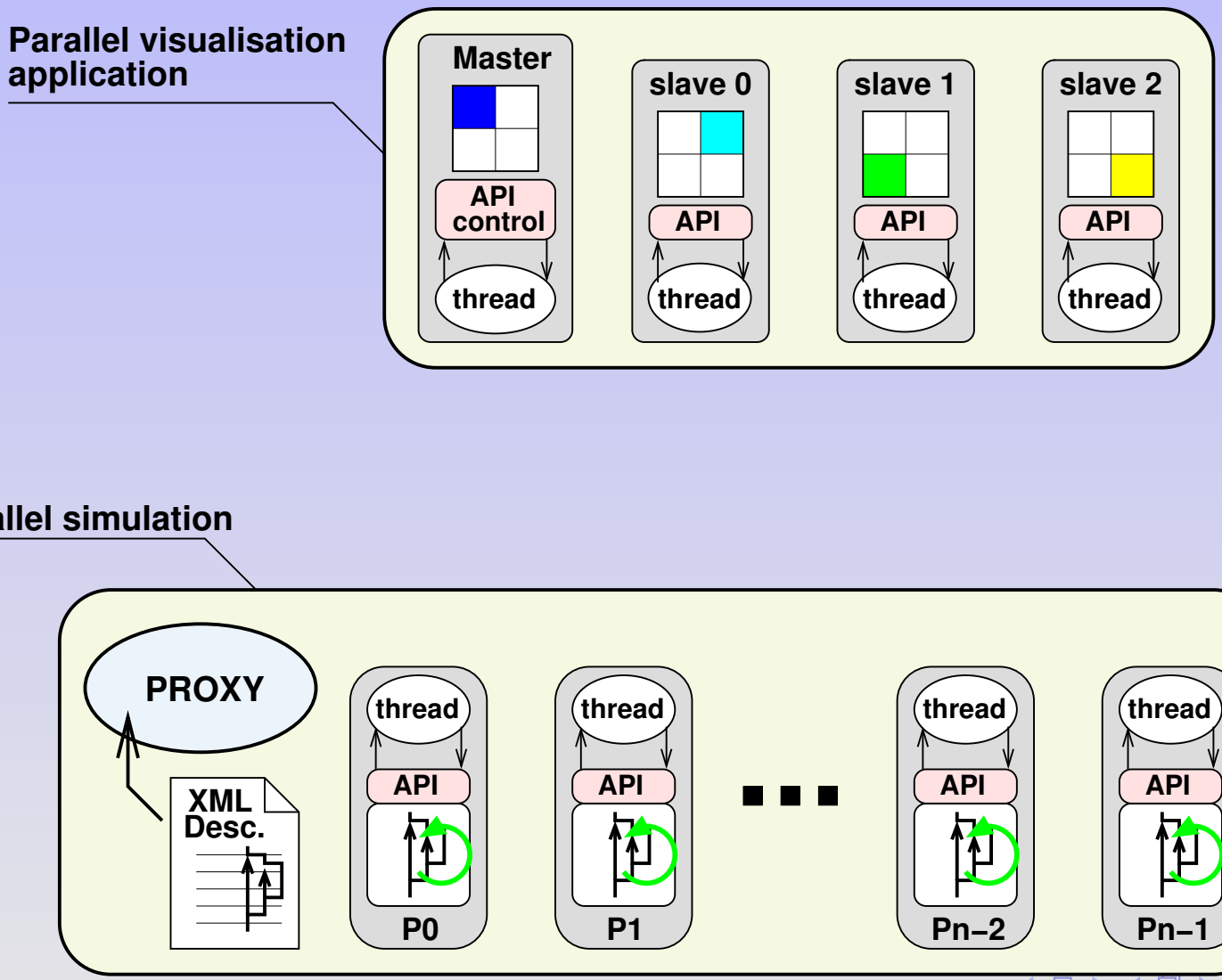




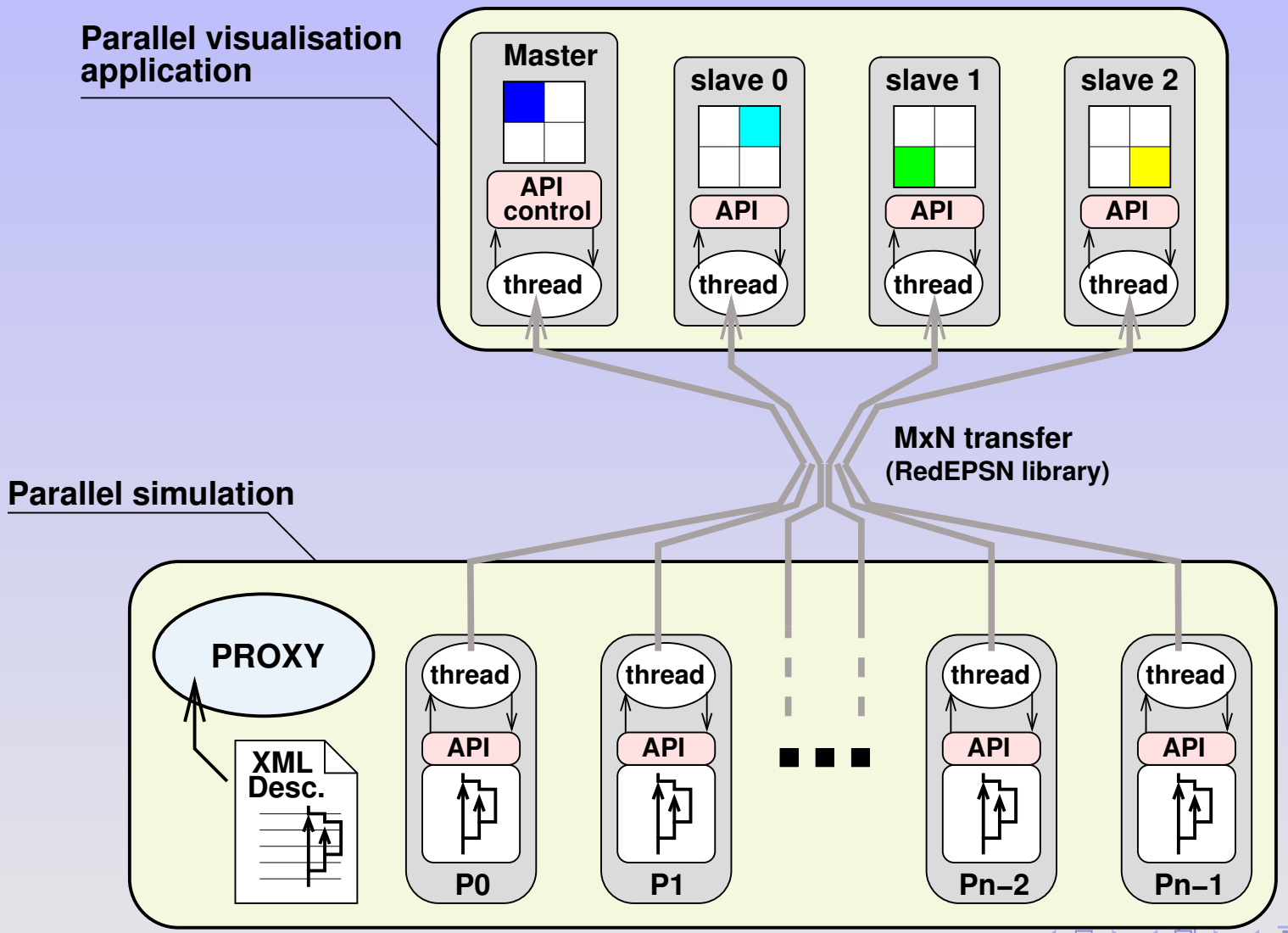
Architecture distribuée



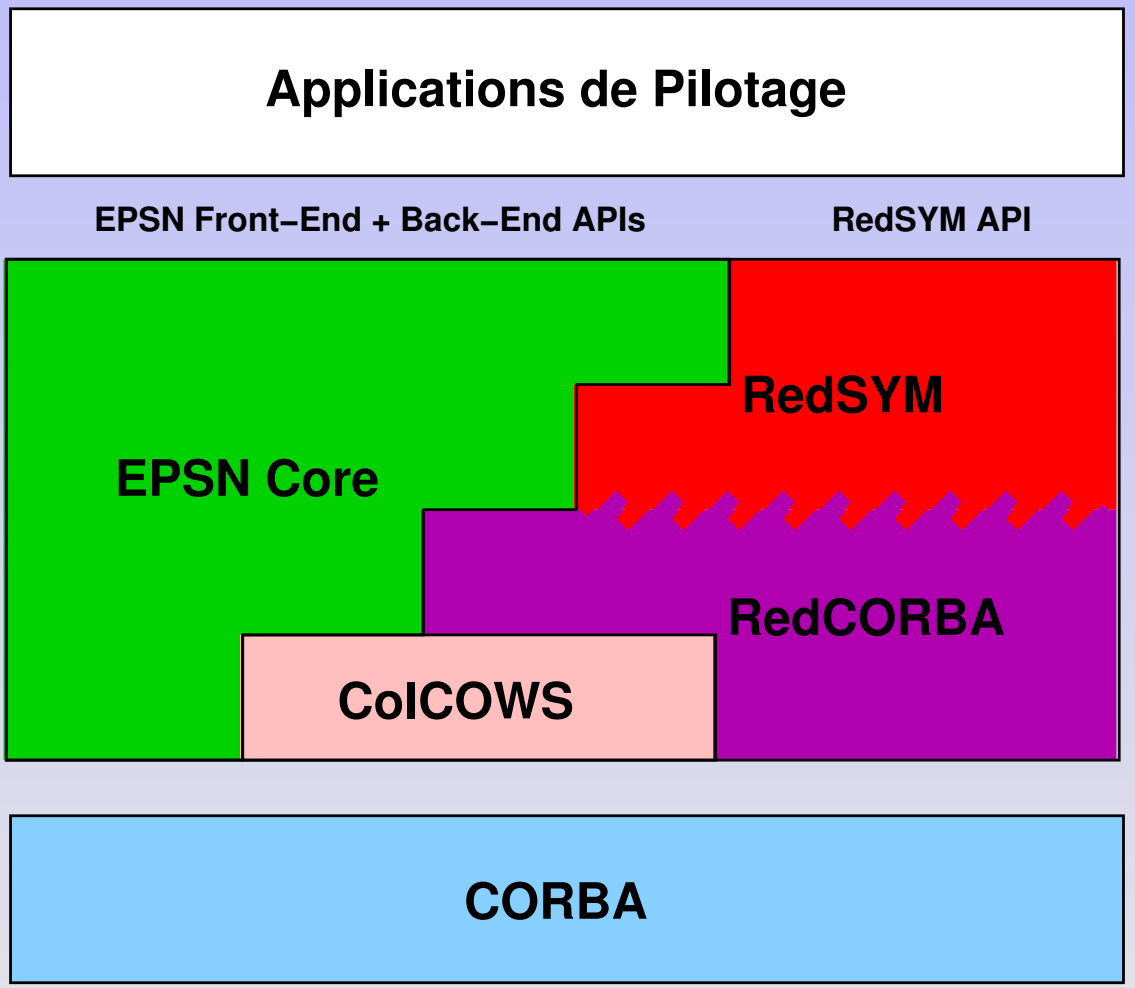
Architecture distribuée



Architecture distribuée



Les bibliothèques associées



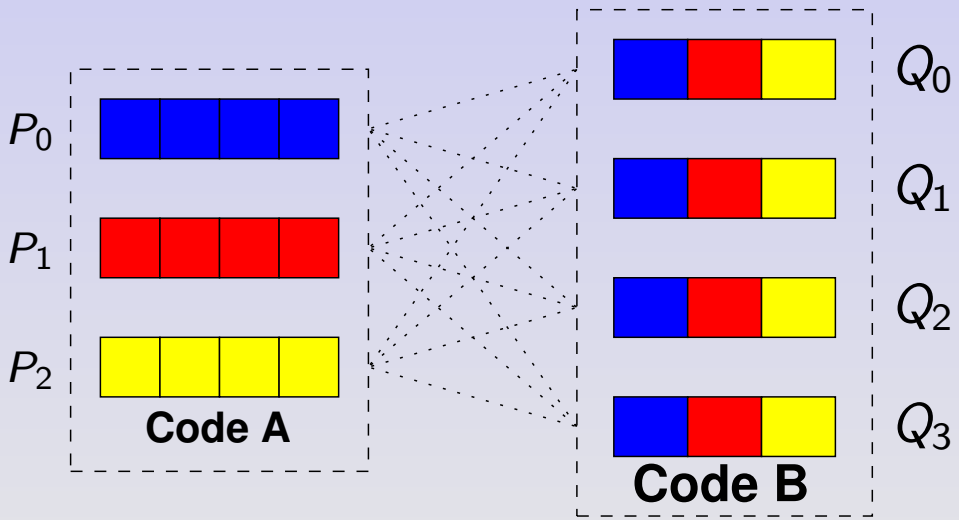
RedSYM/RedCORBA

Bibliothèque RedSYM

- Description des distribution de données
- Accès direct en mémoire
- Calcul des matrices de redistribution

Bibliothèque RedCORBA

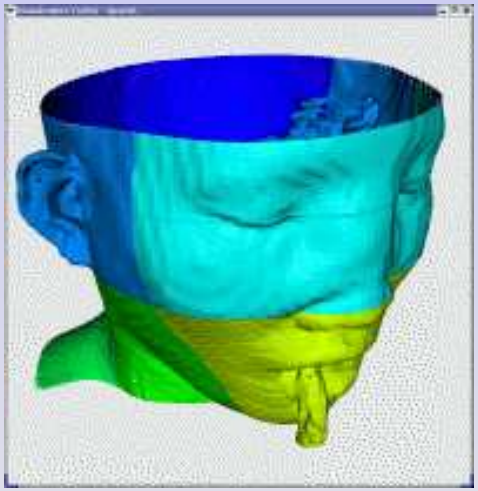
- Transfert des blocs de redistribution
- Communication via CORBA

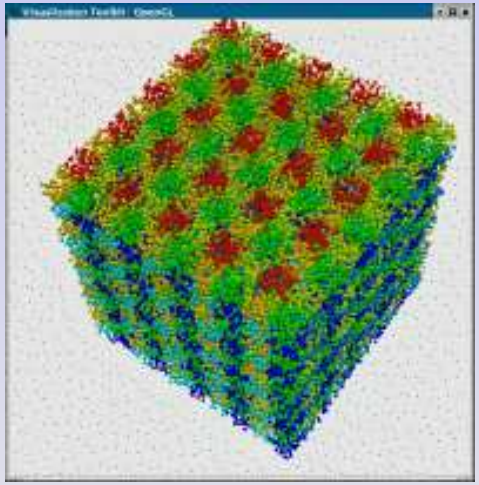


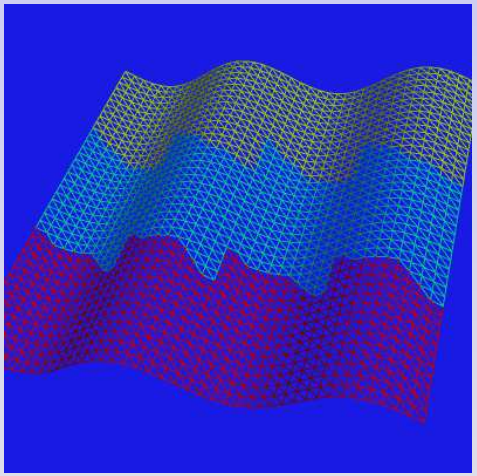
Modèle de données

Modèle de données

- Gestion de données complexes:
  scalaires, champs, points, points en blocs, maillages
- Gestion de données multi-series
- Stockage complexe (strides, structures)



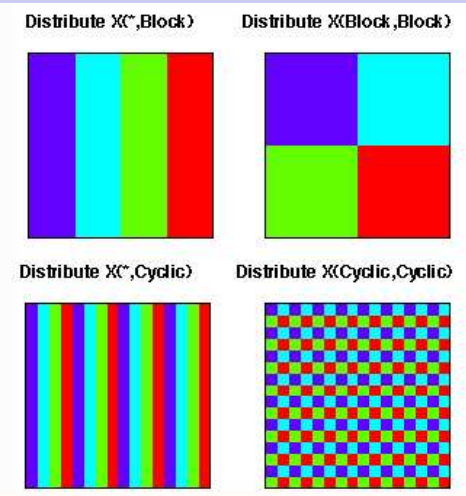


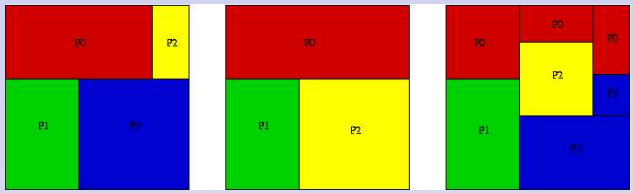


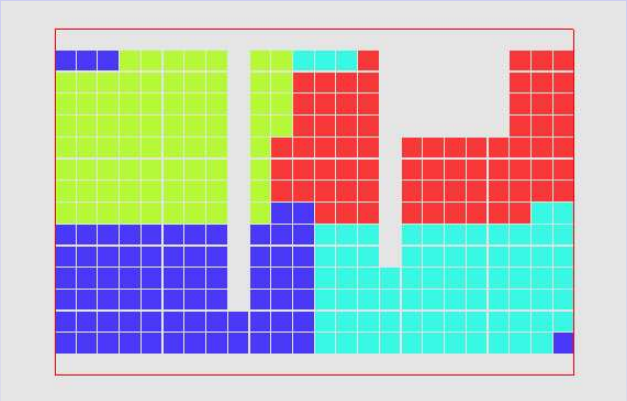
Modèle de distribution

Modèle de distribution

- Distributions bloc-cycliques à la HPF
- Distributions en blocs rectilinéaires et creux
- Partitionnement de maillage (Metis/Scotch)







Plan

Les fonctionnalités de l'API

Controle

- Stop
- Play
- Step

Accès aux données

- Get
- Put

Déclenchement d'actions

- Déclenchement d'actions

Facilités

- Benchmark
- Date courante
- Listes de requêtes

Pilotage

Commandes

- Stop
  - mise en pause de l'application
  - pause sur un état cohérent
- Play
  - reprise de l'exécution
- Step
  - reprise de l'exécution pour quelques pas
  - pause sur un état cohérent

Fonctionnalités XML

- Démarrage contrôlé de l'application
- Pause automatique sur un point d'instrumentation

Accès aux données

Commandes

- Get
  - récupération ponctuelle d'une donnée de la simulation
  - récupération systématique d'une donnée avec fréquence
  - applicable à une ou toutes les séries de la donnée
- Put
  - écriture d'une donnée dans la simulation
  - applicable à une ou toutes les séries de la donnée

Fonctionnalités XML

- Gestion des droits d'accès sur les points
- Gestion des droits d'accès sur les taches

Déclenchement d'actions

Commandes

- Triger
  - déclenchement ponctuel d'une action
  - déclenchement systématique d'une action avec fréquence

Fonctionnalités XML

- Gestion des droits d'exécution sur les taches

Facilités

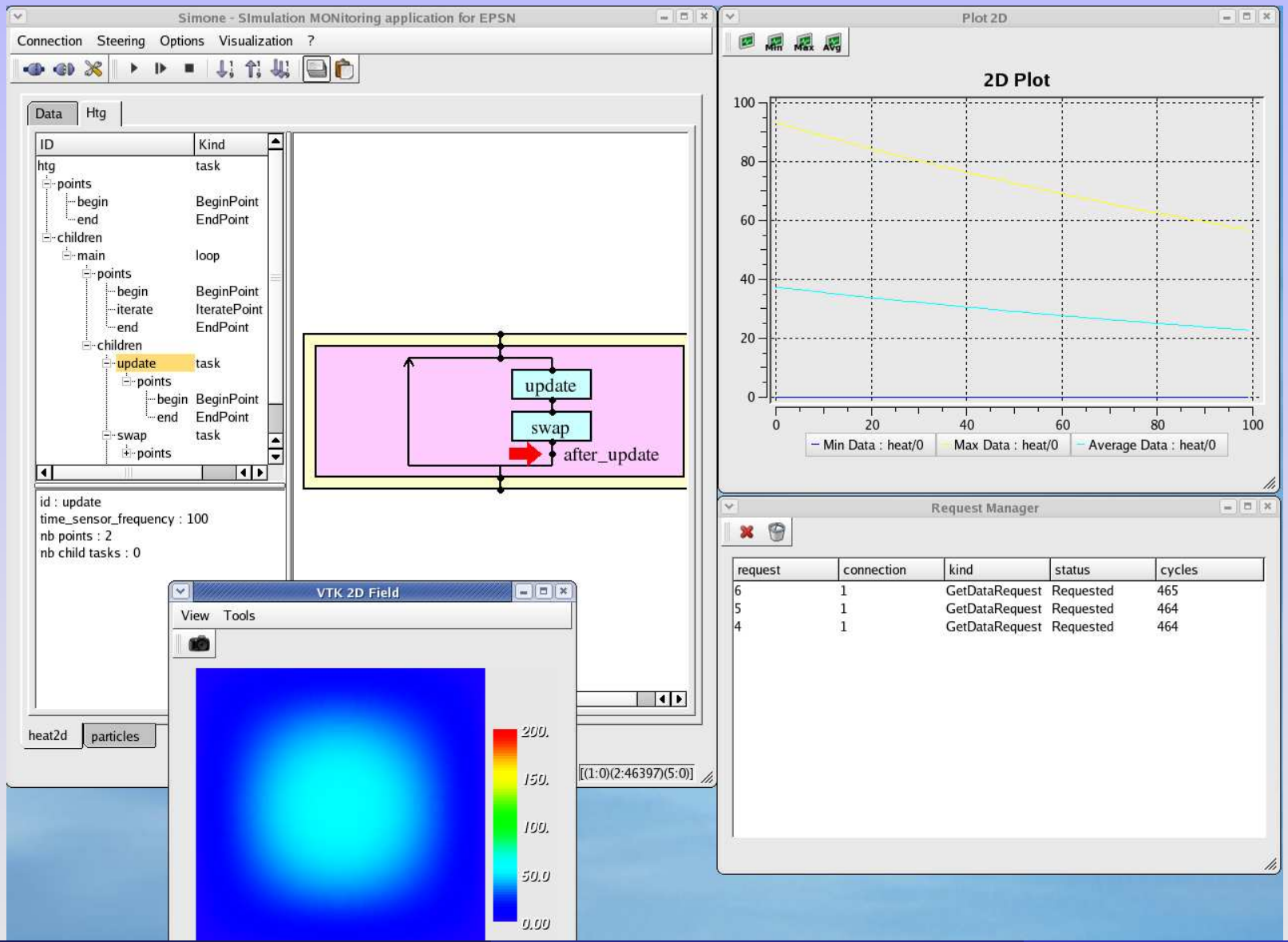
Commandes

- Benchmark
  - récupération ponctuelle des temps d'exécution d'une tache
  - récupération ponctuelle des temps d'exécution d'un point
- Date courante
  - récupération de la date courante de tous les processus
- Listes de requêtes
  - récupération de la liste de requêtes du client
  - récupération de la liste de requêtes des autres clients
  - gestion des requêtes en cours (état, suppression . . .)

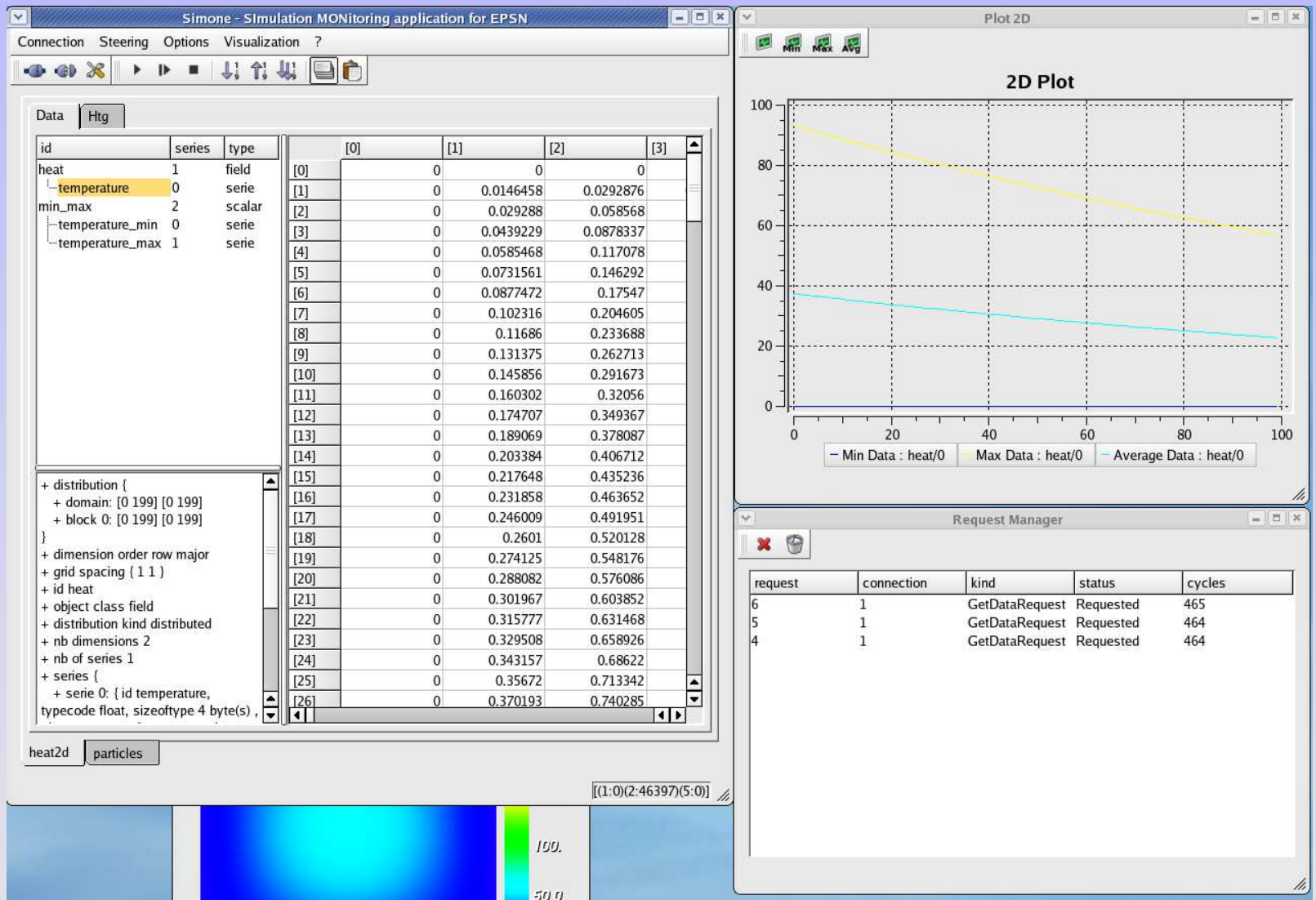
Fonctionnalités XML

- Configuration des benchmarks
  - activation par tache/point
  - réglage de la fréquence des mesures

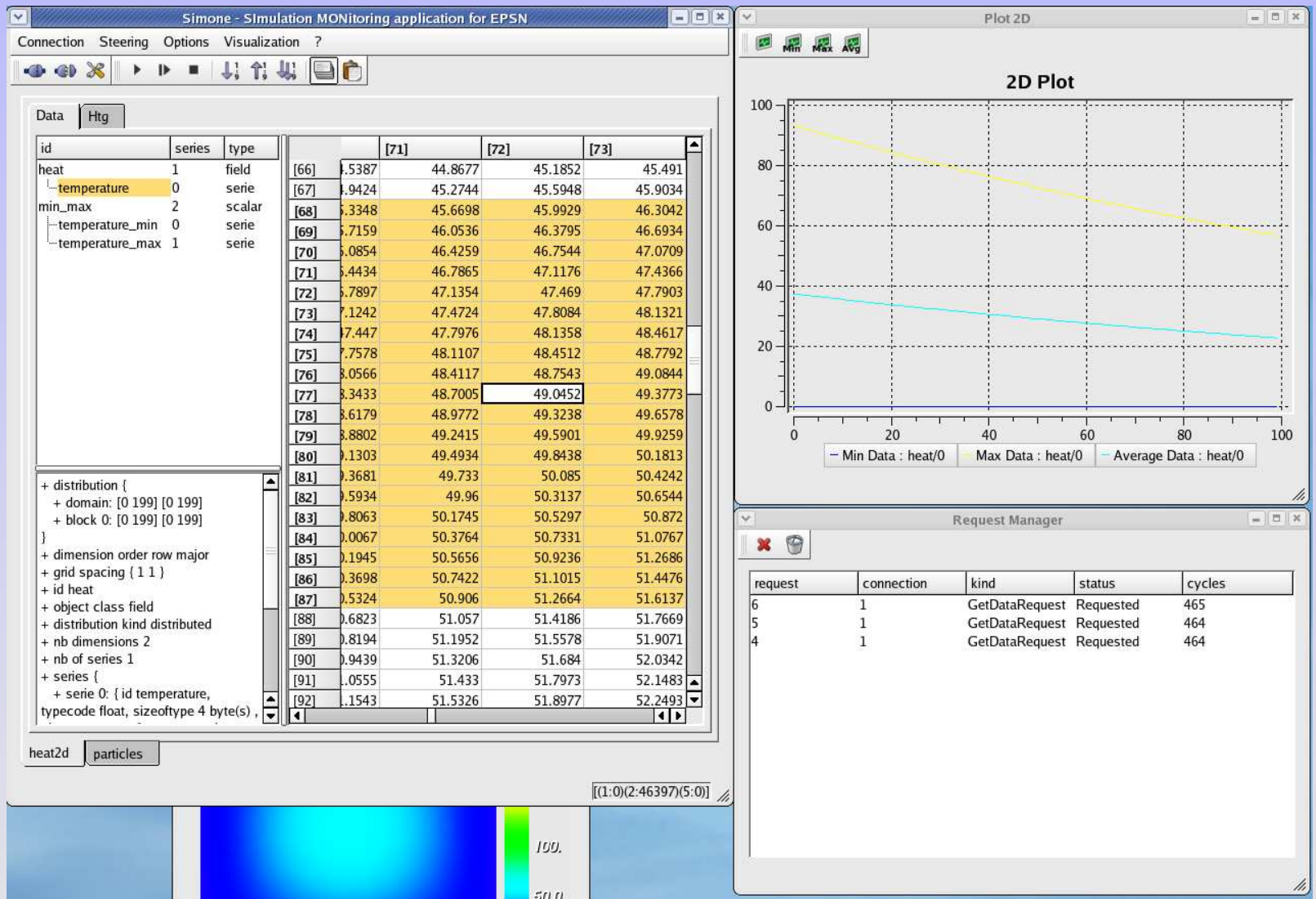
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:46397)(5:0)]
Plot 2D
2D Plot
Min Max Avg
100
80
60
40
20
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
6 1 GetDataRequest Requested 465
5 1 GetDataRequest Requested 464
4 1 GetDataRequest Requested 464
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

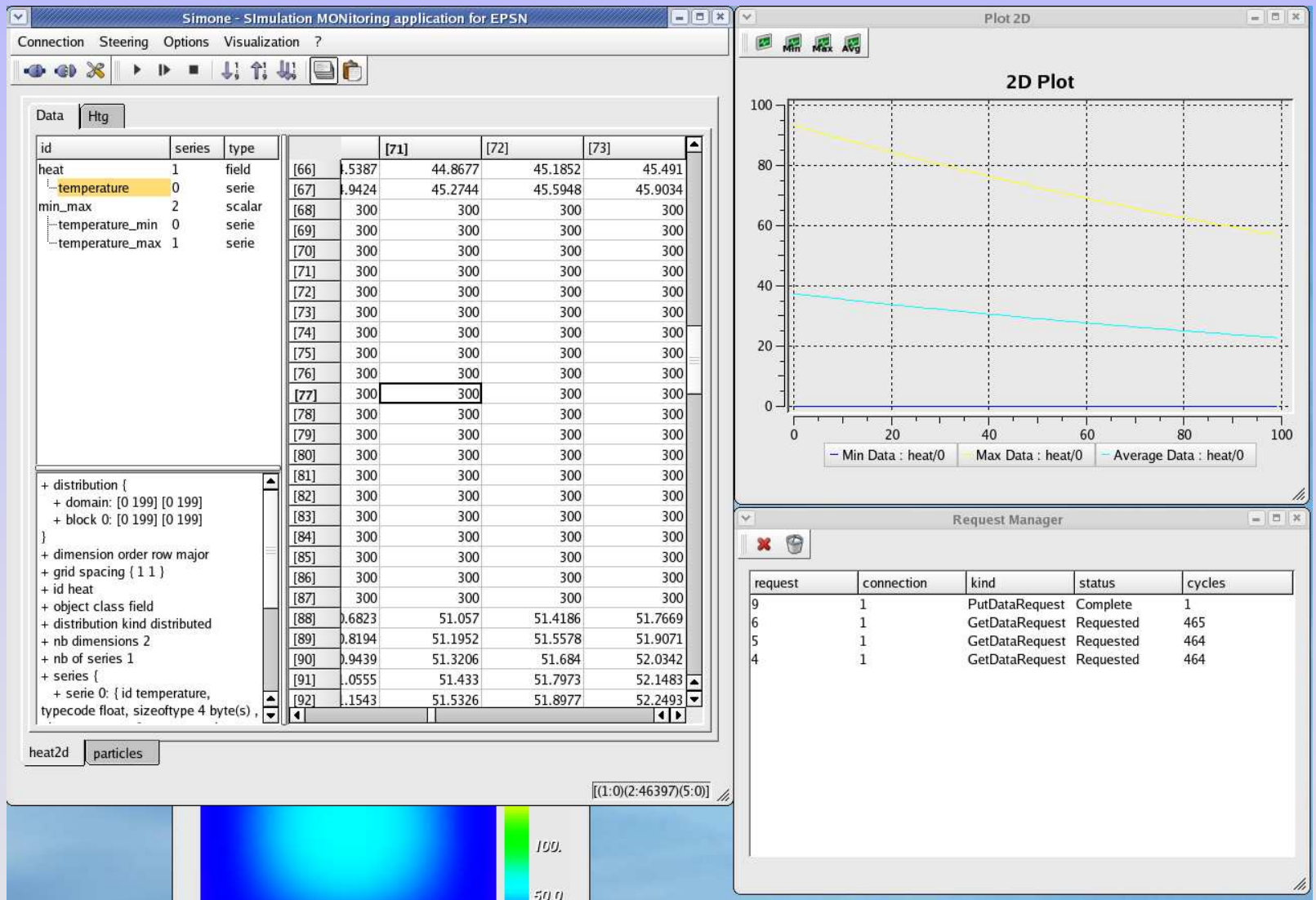
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
id series type
heat 1 field
temperature 0 serie
min_max 2 scalar
temperature_min 0 serie
temperature_max 1 serie
+ distribution {
+ domain: [0 199] [0 199]
+ block 0: [0 199] [0 199]
}
+ dimension order row major
+ grid spacing { 1 1 }
+ id heat
+ object class field
+ distribution kind distributed
+ nb dimensions 2
+ nb of series 1
+ series {
+ serie 0: { id temperature,
typecode float, sizeoftype 4 byte(s) ,
heat2d particles
[(1:0)(2:46397)(5:0)]
Plot 2D
2D Plot
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
6 1 GetDataRequest Requested 465
5 1 GetDataRequest Requested 464
4 1 GetDataRequest Requested 464
100.
50.0
0.00

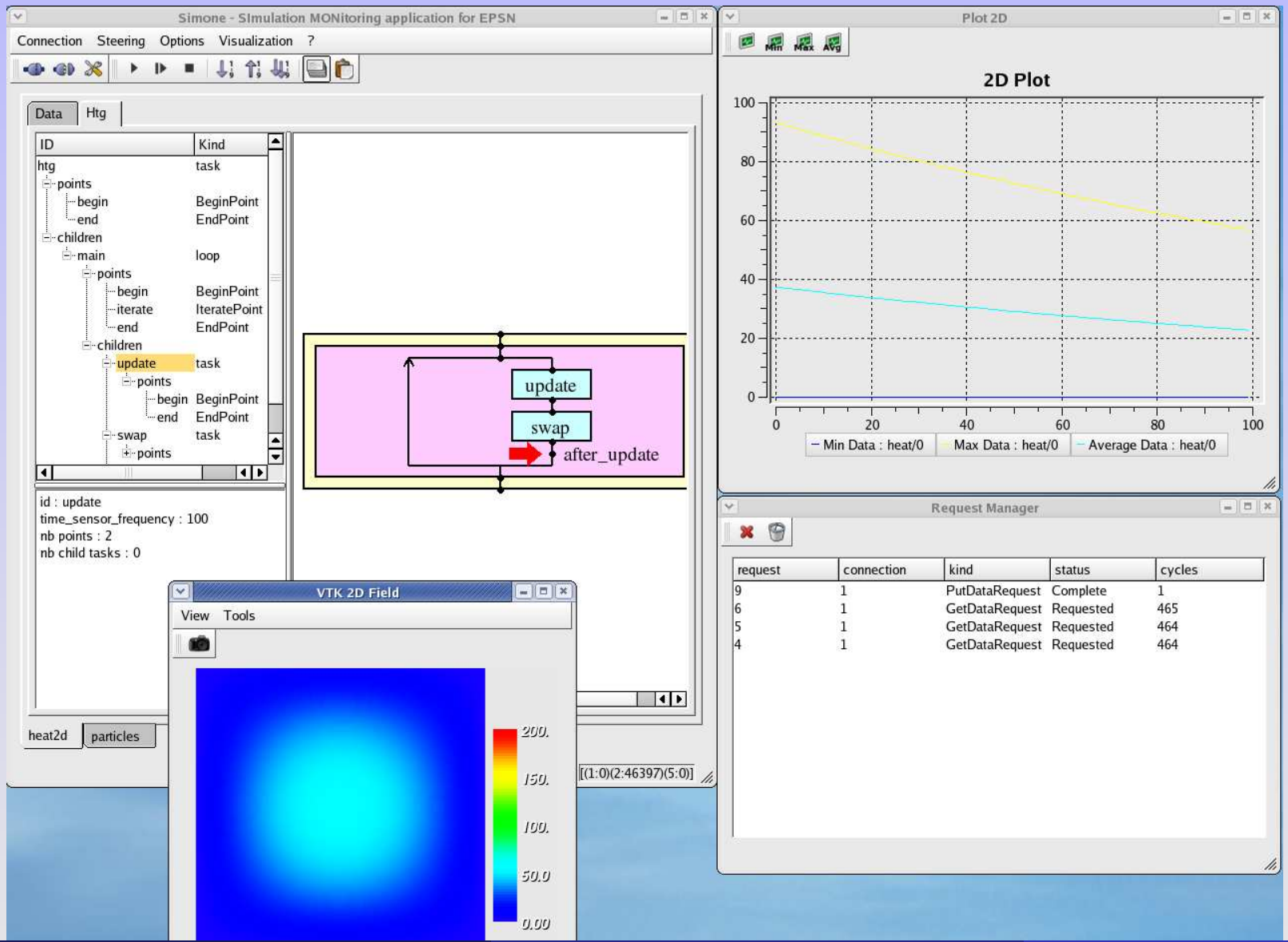
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
id series type
heat 1 field
temperature 0 serie
min_max 2 scalar
temperature_min 0 serie
temperature_max 1 serie
[71] [72] [73]
[66] 44.8677 45.1852 45.491
[67] 45.2744 45.5948 45.9034
[68] 45.6698 45.9929 46.3042
[69] 46.0536 46.3795 46.6934
[70] 46.4259 46.7544 47.0709
[71] 46.7865 47.1176 47.4366
[72] 47.1354 47.469 47.7903
[73] 47.4724 47.8084 48.1321
[74] 47.7976 48.1358 48.4617
[75] 48.1107 48.4512 48.7792
[76] 48.4117 48.7543 49.0844
[77] 48.7005 49.0452 49.3773
[78] 48.9772 49.3238 49.6578
[79] 49.2415 49.5901 49.9259
[80] 49.4934 49.8438 50.1813
[81] 49.733 50.085 50.4242
[82] 49.96 50.3137 50.6544
[83] 50.1745 50.5297 50.872
[84] 50.3764 50.7331 51.0767
[85] 50.5656 50.9236 51.2686
[86] 50.7422 51.1015 51.4476
[87] 50.906 51.2664 51.6137
[88] 51.057 51.4186 51.7669
[89] 51.1952 51.5578 51.9071
[90] 51.3206 51.684 52.0342
[91] 51.433 51.7973 52.1483
[92] 51.5326 51.8977 52.2493
+ distribution {
+ domain: [0 199] [0 199]
+ block 0: [0 199] [0 199]
}
+ dimension order row major
+ grid spacing { 1 1 }
+ id heat
+ object class field
+ distribution kind distributed
+ nb dimensions 2
+ nb of series 1
+ series {
+ serie 0: { id temperature,
typecode float, sizeoftype 4 byte(s) ,
heat2d particles
[(1:0)(2:46397)(5:0)]
Plot 2D
Min Max Avg
2D Plot
100
80
60
40
20
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
6 1 GetDataRequest Requested 465
5 1 GetDataRequest Requested 464
4 1 GetDataRequest Requested 464
100.
50.0
0.00

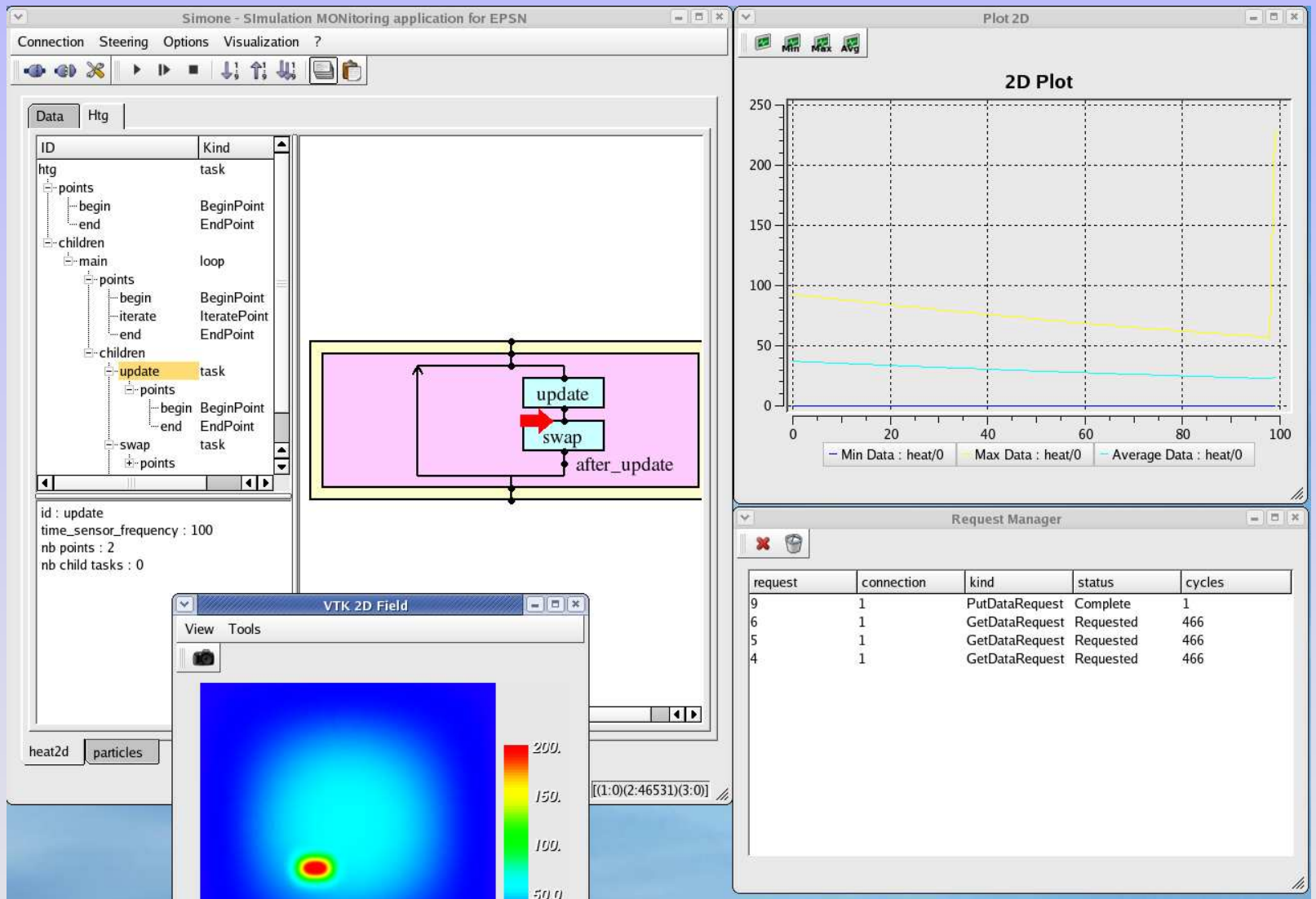
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
id series type
heat 1 field
temperature 0 serie
min_max 2 scalar
temperature_min 0 serie
temperature_max 1 serie
+ distribution {
+ domain: [0 199] [0 199]
+ block 0: [0 199] [0 199]
}
+ dimension order row major
+ grid spacing { 1 1 }
+ id heat
+ object class field
+ distribution kind distributed
+ nb dimensions 2
+ nb of series 1
+ series {
+ serie 0: { id temperature,
typecode float, sizeoftype 4 byte(s) ,
heat2d particles
[(1:0)(2:46397)(5:0)]
Plot 2D
2D Plot
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 465
5 1 GetDataRequest Requested 464
4 1 GetDataRequest Requested 464
100.
50.0
0.00

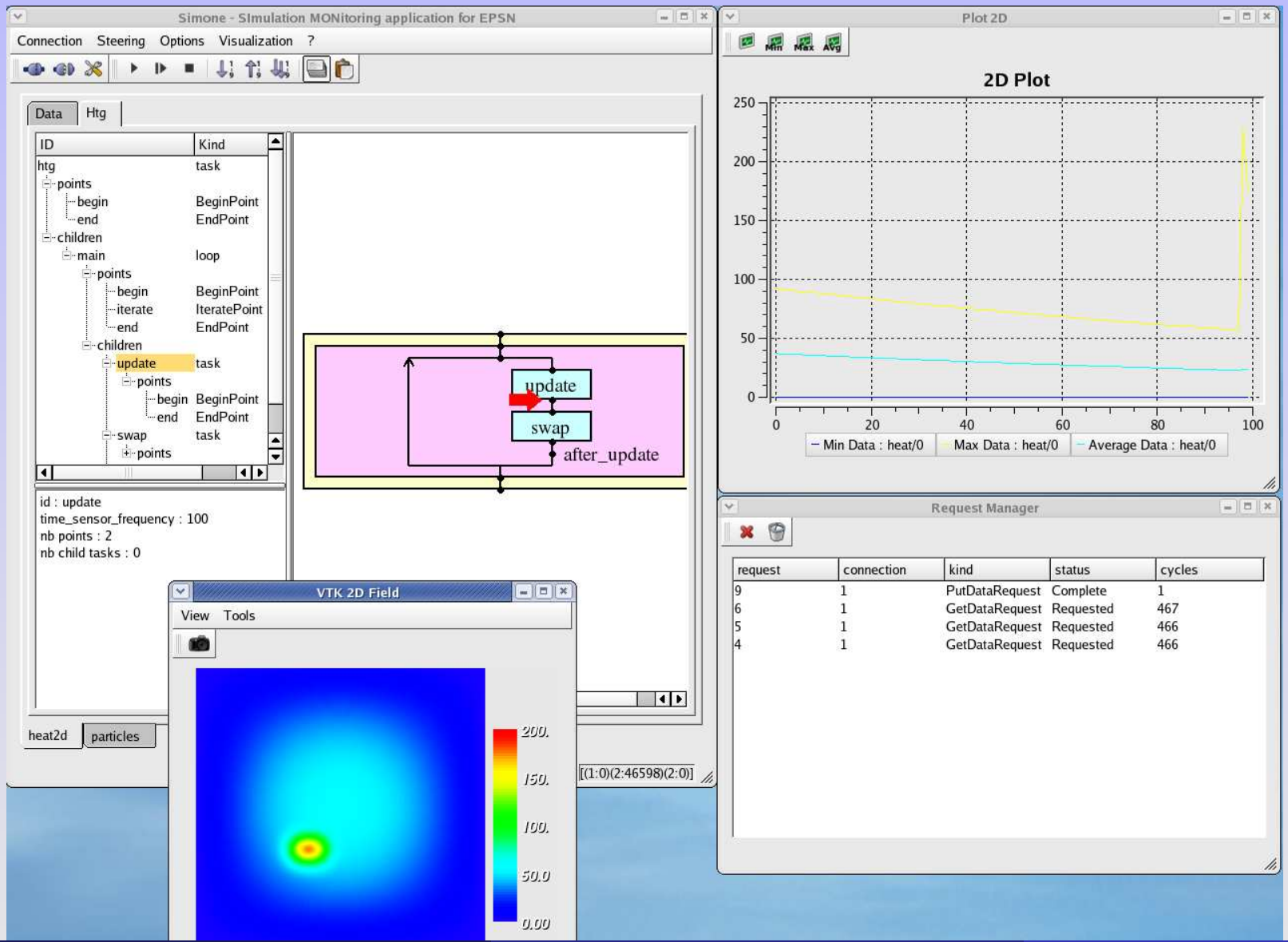
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:46397)(5:0)]
Plot 2D
2D Plot
Min Max Avg
100
80
60
40
20
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 465
5 1 GetDataRequest Requested 464
4 1 GetDataRequest Requested 464
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

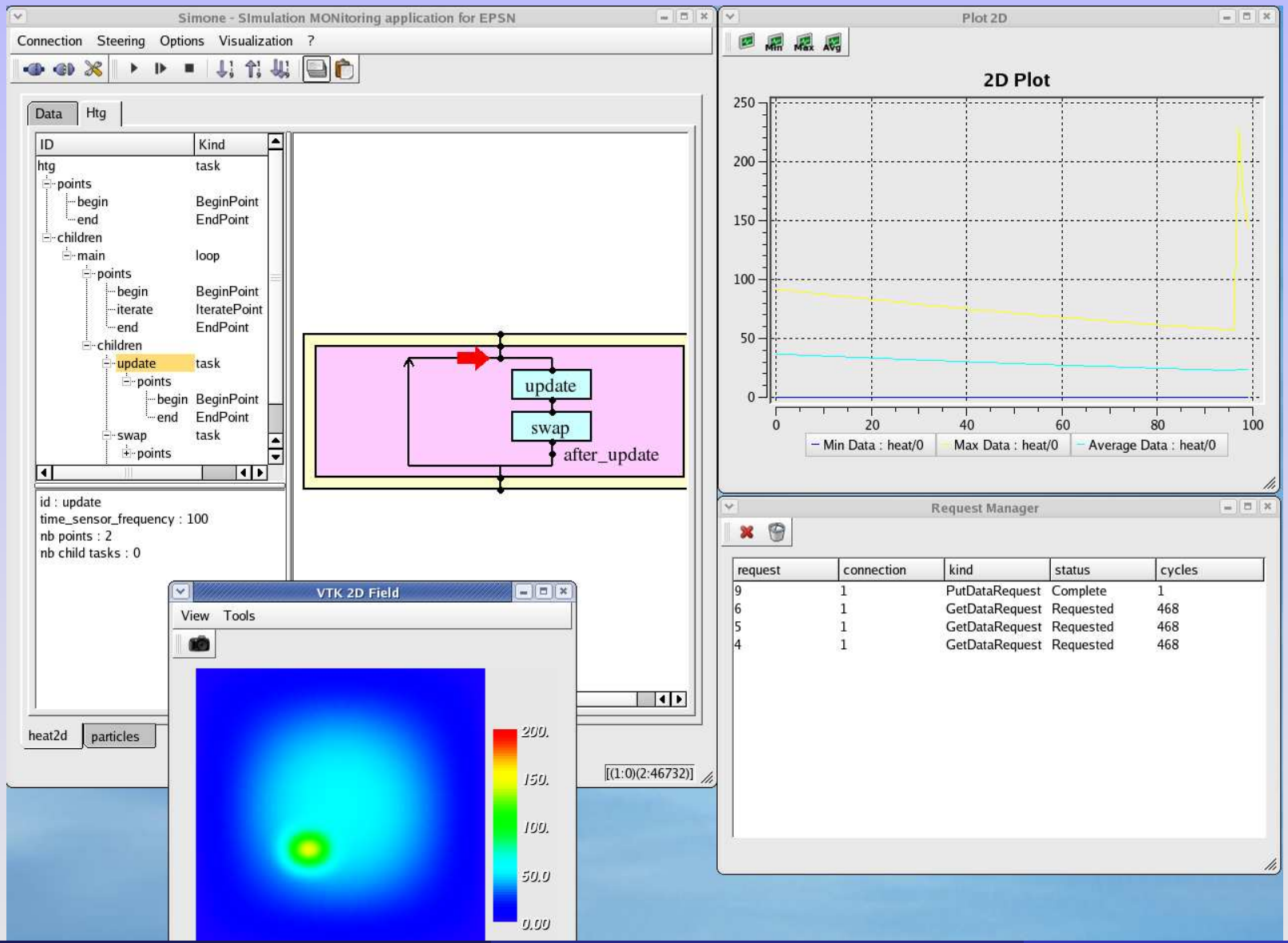
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:46531)(3:0)]
Plot 2D
2D Plot
250
200
150
100
50
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 466
5 1 GetDataRequest Requested 466
4 1 GetDataRequest Requested 466
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

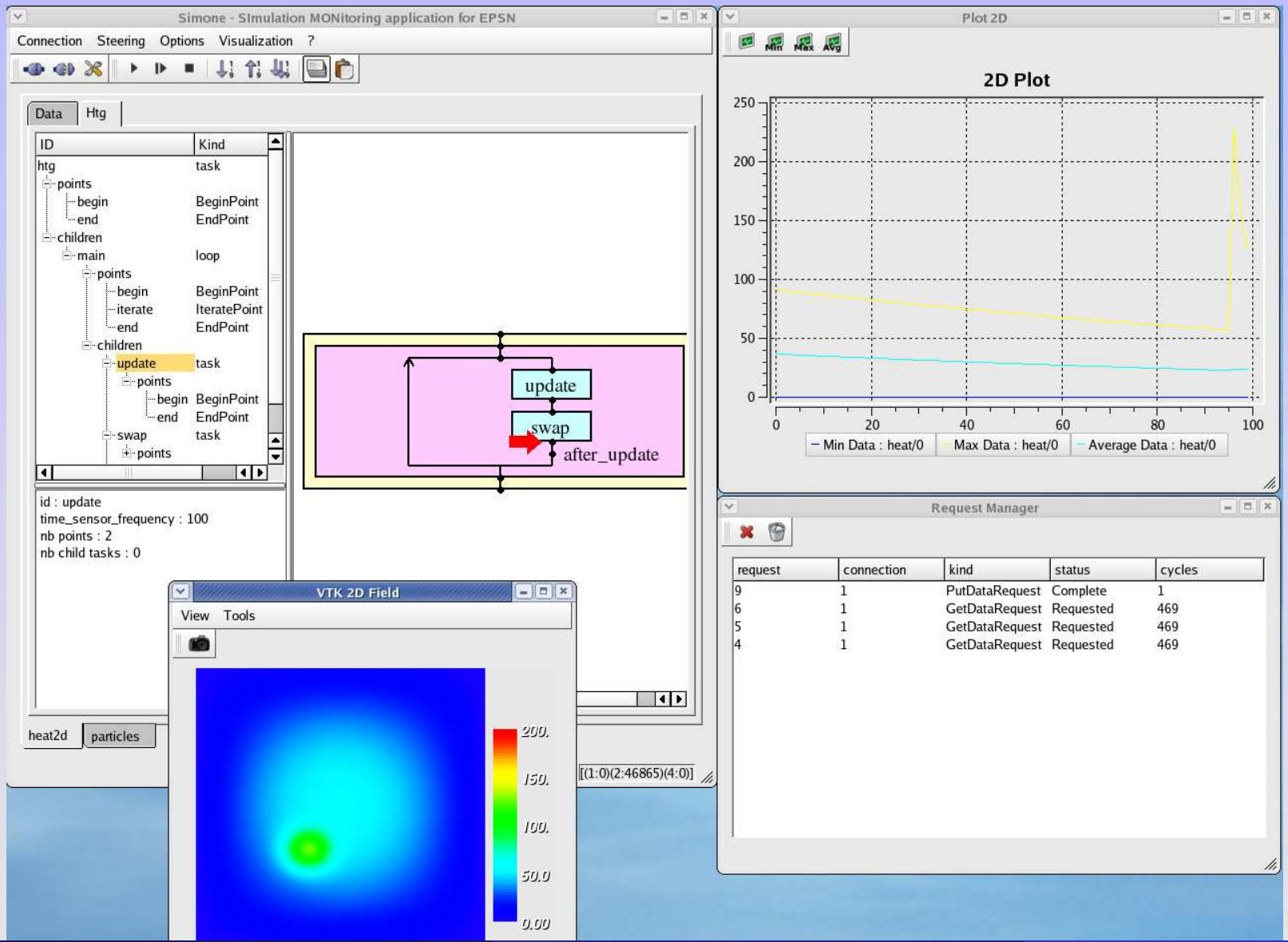
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:46598)(2:0)]
Plot 2D
2D Plot
250
200
150
100
50
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 467
5 1 GetDataRequest Requested 466
4 1 GetDataRequest Requested 466
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

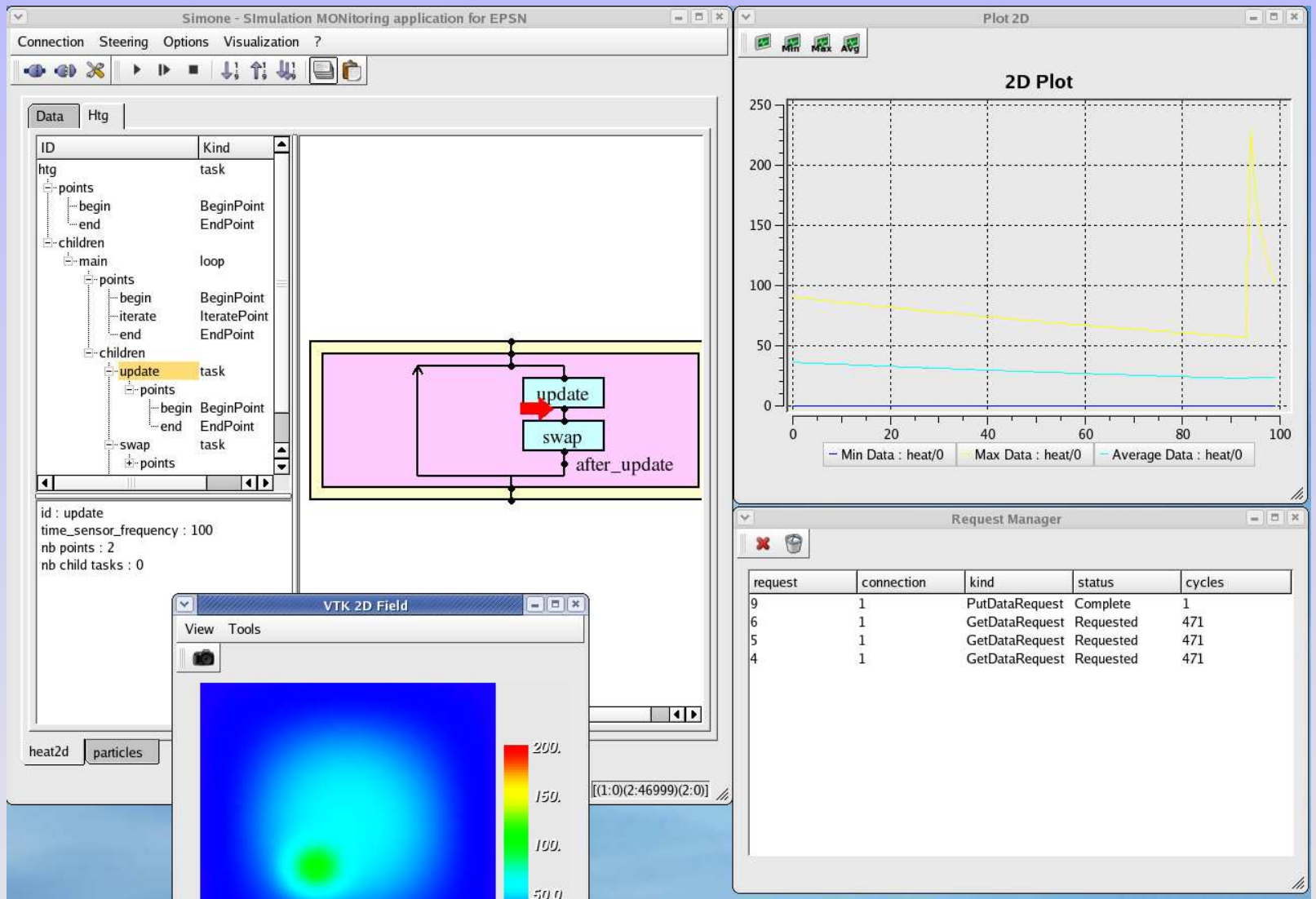
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:46732)]
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00
Plot 2D
2D Plot
250
200
150
100
50
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 468
5 1 GetDataRequest Requested 468
4 1 GetDataRequest Requested 468

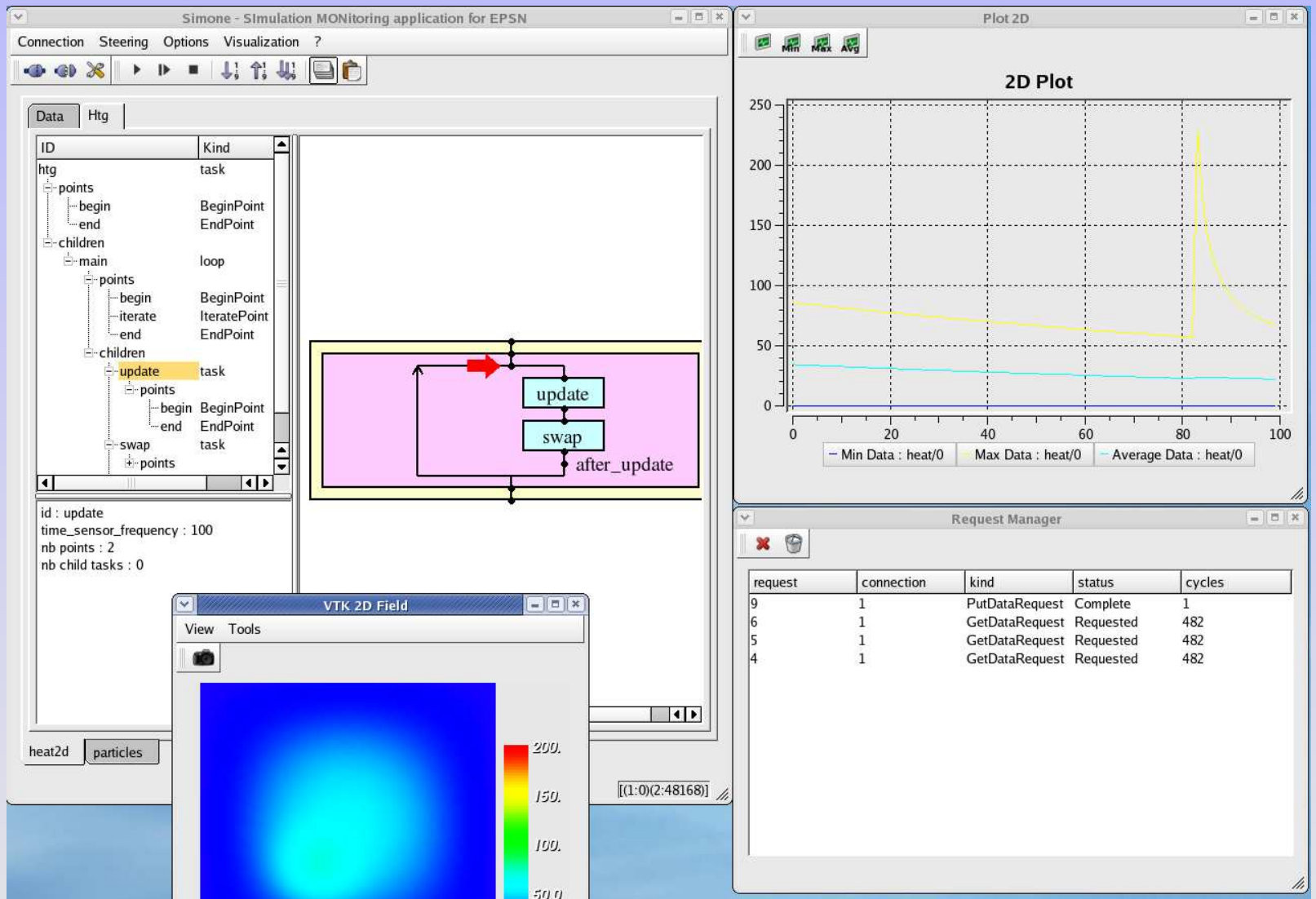
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:46865)(4:0)]
Plot 2D
2D Plot
250 200 150 100 50 0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 469
5 1 GetDataRequest Requested 469
4 1 GetDataRequest Requested 469
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

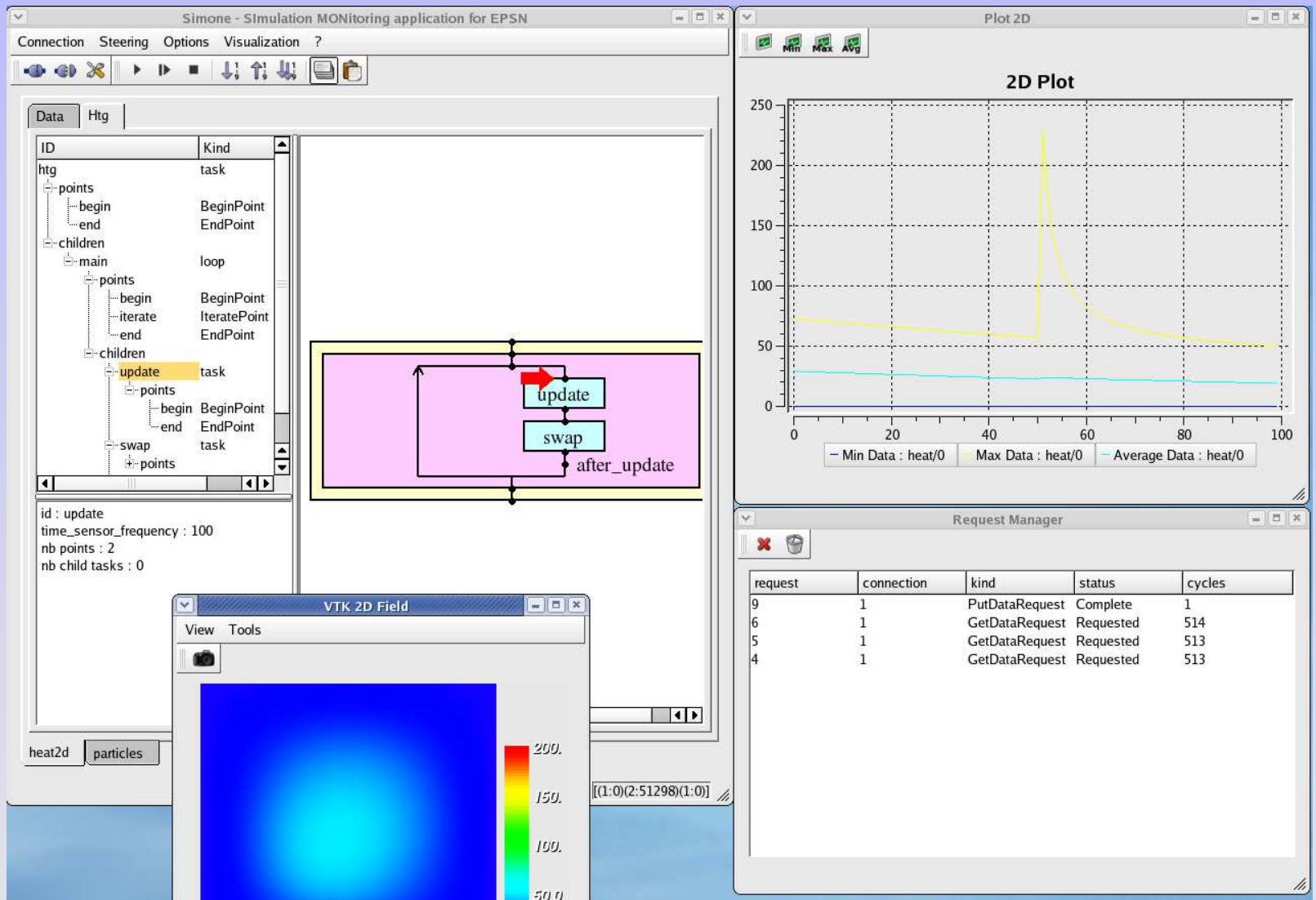
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:46999)(2:0)]
Plot 2D
2D Plot
250
200
150
100
50
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 471
5 1 GetDataRequest Requested 471
4 1 GetDataRequest Requested 471
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

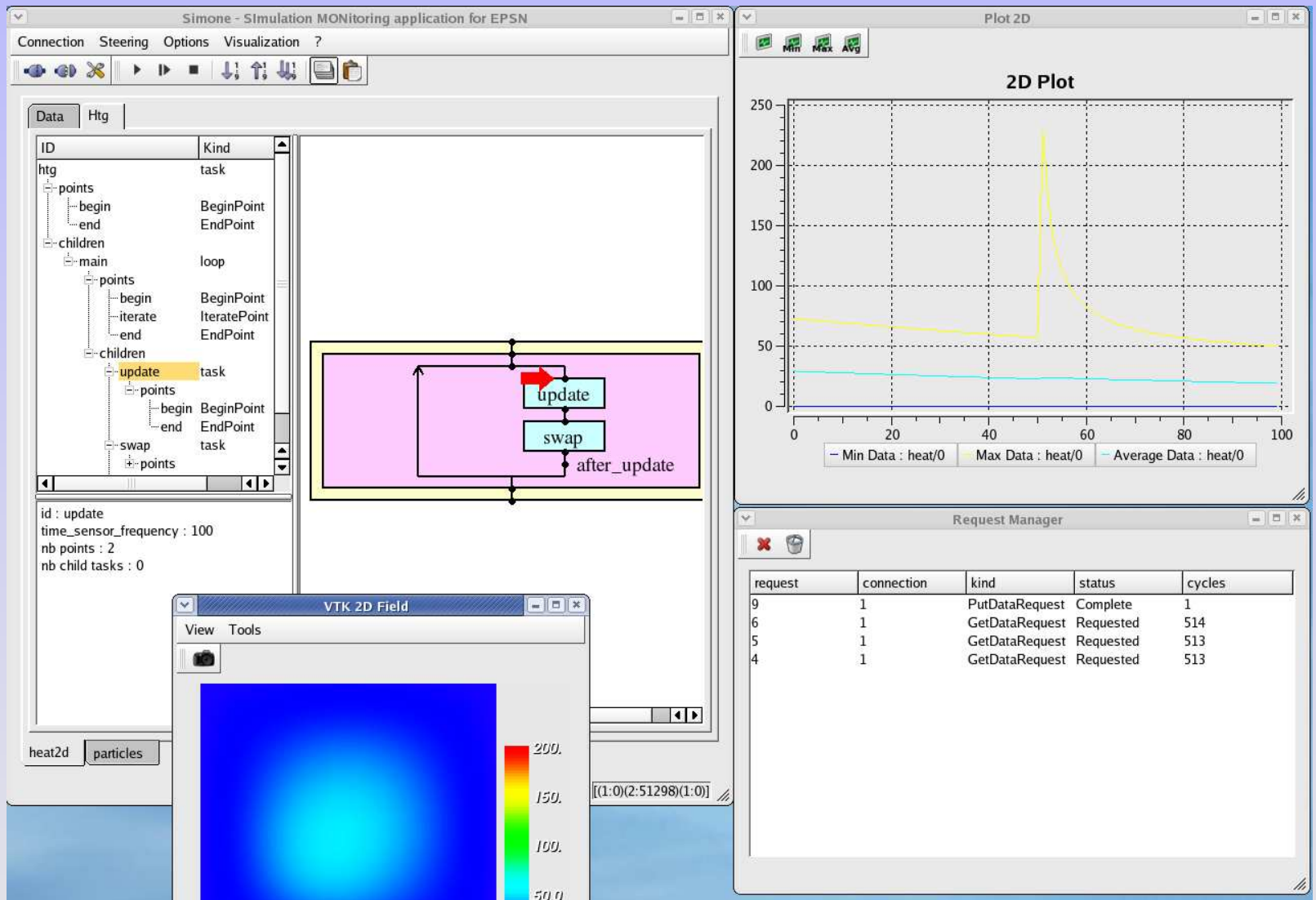
Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:48168)]
Plot 2D
2D Plot
250
200
150
100
50
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 482
5 1 GetDataRequest Requested 482
4 1 GetDataRequest Requested 482
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:51298)(1:0)]
Plot 2D
2D Plot
250
200
150
100
50
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 514
5 1 GetDataRequest Requested 513
4 1 GetDataRequest Requested 513
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

Simone


Simone - SImulation MONitoring application for EPSN
Connection Steering Options Visualization ?
Data Htg
ID Kind
htg task
points
begin BeginPoint
end EndPoint
children
main loop
points
begin BeginPoint
iterate IteratePoint
end EndPoint
children
update task
points
begin BeginPoint
end EndPoint
swap task
points
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
update
swap
after_update
heat2d particles
[(1:0)(2:51298)(1:0)]
Plot 2D
2D Plot
250
200
150
100
50
0
0 20 40 60 80 100
Min Data : heat/0
Max Data : heat/0
Average Data : heat/0
Request Manager
request connection kind status cycles
9 1 PutDataRequest Complete 1
6 1 GetDataRequest Requested 514
5 1 GetDataRequest Requested 513
4 1 GetDataRequest Requested 513
VTK 2D Field
View Tools
200.
150.
100.
50.0
0.00

Plan

Analyse de la structure du code

Structure du code

Code source

```
struct Particle  double x, y, z, vx, vy, vz, ax, ay, az, mass, charge; ;
Particle  particles[MAX_PARTICLES];
  ...
init_particles( particles, my_offset, particles_number)

for (count = 0; count < N_OF_ITERATIONS; count++) {

    exchange_particles( particles, my_offset, particles_number)

    compute_acceleration( particles, my_offset, particles_number)

    move_particles( particles, my_offset, particles_number)
}
MPI_Finalize ();
```

Analyse de la structure du code

Structure du code

Code source

```
struct Particle  double x, y, z, vx, vy, vz, ax, ay, az, mass, charge; ;
Particle  particles[MAX_PARTICLES];
  ...
init_particles( particles, my_offset, particles_number)

for (count = 0; count < N_OF_ITERATIONS; count++) {

    exchange_particles( particles, my_offset, particles_number)

    compute_acceleration( particles, my_offset, particles_number)

    move_particles( particles, my_offset, particles_number)
}
MPI_Finalize ();
```

Analyse de la structure du code

Structure du code

Code source

```
struct Particle  double x, y, z, vx, vy, vz, ax, ay, az, mass, charge; ;
Particle  particles[MAX_PARTICLES];
  ...
init_particles( particles, my_offset, particles_number)

for (count = 0; count < N_OF_ITERATIONS; count++) {

    exchange_particles( particles, my_offset, particles_number)

    compute_acceleration( particles, my_offset, particles_number)

    move_particles( particles, my_offset, particles_number)
}
MPI_Finalize ();
```

Analyse de la structure du code

Structure du code

Code source

```
struct Particle  double x, y, z, vx, vy, vz, ax, ay, az, mass, charge; ;
Particle  particles[MAX_PARTICLES];
  ...
init_particles( particles, my_offset, particles_number)

for (count = 0; count < N_OF_ITERATIONS; count++) {

    exchange_particles( particles, my_offset, particles_number)

    compute_acceleration( particles, my_offset, particles_number)

    move_particles( particles, my_offset, particles_number)
}
MPI_Finalize ();
```

Description de la structure en XML

XML

```
<simulation id="particles">
  <data>
      <points id="particles" />
  </data>
  <htg>
      <loop id="main">
        <task id="body">
          <task id="acceleration"/>
          <task id="move"/>
        </task>
      </loop>
  </htg>
</simulation>
```

Description de la structure en XML

XML

```
<simulation id="particles">
  <data>
      <points id="particles" />
  </data>
  <htg>
      <loop id="main">
        <task id="body">
          <task id="acceleration"/>
          <task id="move"/>
        </task>
      </loop>
  </htg>
</simulation>
```

Initialisation et terminaison

Code instrumenté

```
Particle  particles[MAX_PARTICLES];
MPI_Init(&argc, &argv);
  ...
init_particles( particles, my_offset, particles_number)

EPSN::Simulation::Interface * epsn_itfc = new EPSN::Simulation::Interface;
epsn_itfc->initORB(argc,argv);
if(my_rank == 0)
     epsn_itfc->initProxyAndNode("particles", "particles.xml", 0, pool_size);
else
     epsn_itfc->initNode("particles", my_rank, pool_size );

for (count = 0; count < N_OF_ITERATIONS; count++) {

    exchange_particles( particles, my_offset, particles_number)

    compute_acceleration( particles, my_offset, particles_number)

    move_particles( particles, my_offset, particles_number)

}

epsn_itfc->finalize();
epsn_itfc->killORB();
delete epsn_itfc;
MPI_Finalize ();
```

Initialisation et terminaison

Code instrumenté

```
Particle  particles[MAX_PARTICLES];
MPI_Init(&argc, &argv);
  ...
init_particles( particles, my_offset, particles_number)

EPSN::Simulation::Interface * epsn_itfc = new EPSN::Simulation::Interface;
epsn_itfc->initORB(argc,argv);
if(my_rank == 0)
     epsn_itfc->initProxyAndNode("particles", "particles.xml", 0, pool_size);
else
     epsn_itfc->initNode("particles", my_rank, pool_size );

for (count = 0; count < N_OF_ITERATIONS; count++) {

    exchange_particles( particles, my_offset, particles_number)

    compute_acceleration( particles, my_offset, particles_number)

    move_particles( particles, my_offset, particles_number)

}

epsn_itfc->finalize();
epsn_itfc->killORB();
delete epsn_itfc;
MPI_Finalize ();
```

Insertion des points d'instrumentation

Code instrumenté

```
...
epsn_itfc->beginHtg();
epsn_itfc->beginLoop("main");
for (count = 0; count < N_OF_ITERATIONS; count++) {
    epsn_itfc->beginTask("body");
    exchange_particles( particles, my_offset, particles_number)
    epsn_itfc->beginTask("acceleration");
    compute_acceleration( particles, my_offset, particles_number)
    epsn_itfc->endTask("acceleration");
    epsn_itfc->beginTask("move");
    move_particles( particles, my_offset, particles_number)
    epsn_itfc->endTask("move");
    epsn_itfc->endTask("body");
}
epsn_itfc->endLoop("main");
epsn_itfc->endHtg();
...
```

Insertion des points d'instrumentation

Code instrumenté

```
...
epsn_itfc->beginHtg();
epsn_itfc->beginLoop("main");
for (count = 0; count < N_OF_ITERATIONS; count++) {
    epsn_itfc->beginTask("body");
    exchange_particles( particles, my_offset, particles_number)
    epsn_itfc->beginTask("acceleration");
    compute_acceleration( particles, my_offset, particles_number)
    epsn_itfc->endTask("acceleration");
    epsn_itfc->beginTask("move");
    move_particles( particles, my_offset, particles_number)
    epsn_itfc->endTask("move");
    epsn_itfc->endTask("body");
}
epsn_itfc->endLoop("main");
epsn_itfc->endHtg();
...
```

Instrumentation des données

Code instrumenté

```
Particle  particles[MAX_PARTICLES];
MPI_Init(&argc, &argv);
  ...
init_particles( particles, my_offset, particles_number)

EPSN::Simulation::Interface * epsn_itfc = new EPSN::Simulation::Interface;
epsn_itfc->initORB(argc,argv);
if(my_rank == 0)
     epsn_itfc->initProxyAndNode("particles", "particles.xml", 0, pool_size);
else
     epsn_itfc->initNode("particles", my_rank, pool_size );

RedSYM::Points * points = new RedSYM::Points("particles", nb_dims);
points->setNumberOfPoints(particle_number, MAX_PARTICLES);
points->wrapStridedCoordinates(particles+my_offset, sizeof(Particle), 0 );

epsn_itfc->addDataObject(points);
epsn_itfc->ready();
...
```

Instrumentation des données

Code instrumenté

```
Particle  particles[MAX_PARTICLES];
MPI_Init(&argc, &argv);
  ...
init_particles( particles, my_offset, particles_number)

EPSN::Simulation::Interface * epsn_itfc = new EPSN::Simulation::Interface;
epsn_itfc->initORB(argc,argv);
if(my_rank == 0)
     epsn_itfc->initProxyAndNode("particles", "particles.xml", 0, pool_size);
else
     epsn_itfc->initNode("particles", my_rank, pool_size );

RedSYM::Points * points = new RedSYM::Points("particles", nb_dims);
points->setNumberOfPoints(particle_number, MAX_PARTICLES);
points->wrapStridedCoordinates(particles+my_offset, sizeof(Particle), 0 );

epsn_itfc->addDataObject(points);
epsn_itfc->ready();
...
```

Definition des plages d'accéssibilité

XML

```
<simulation id="particles">
  <data>
     <points id="particles"/>
  </data>
  <htg auto-start="false">
     <loop id="main">
       <task id="body">
         <task id="acceleration"/>
         <task id="move" >
           <data-context ref="particles" context="modified"/>
         </task>
       </task>
     </loop>
  </htg>
</simulation>
```

Definition des plages d'accéssibilité

XML

```
<simulation id="particles">
  <data>
     <points id="particles"/>
  </data>
  <htg auto-start="false">
     <loop id="main">
       <task id="body">
         <task id="acceleration"/>
         <task id="move" >
           <data-context ref="particles" context="modified"/>
         </task>
       </task>
     </loop>
  </htg>
</simulation>
```

Configuration de la simulation

XML

```
<simulation id="particles">
  <data>
     <points id="particles" dimension="3" />
  </data>
  <htg  auto-start="false" >
     <loop id="main"  synchronize="no" >
       <task id="body"  synchronize="no"/>
         <task id="acceleration"  synchronize="no"/>
         <task id="move"  synchronize="no">
           <data-context ref="particles" context="modified"/>
         </task>
       </task>
     </loop>
  </htg>
</simulation>
```

Configuration de la simulation

XML

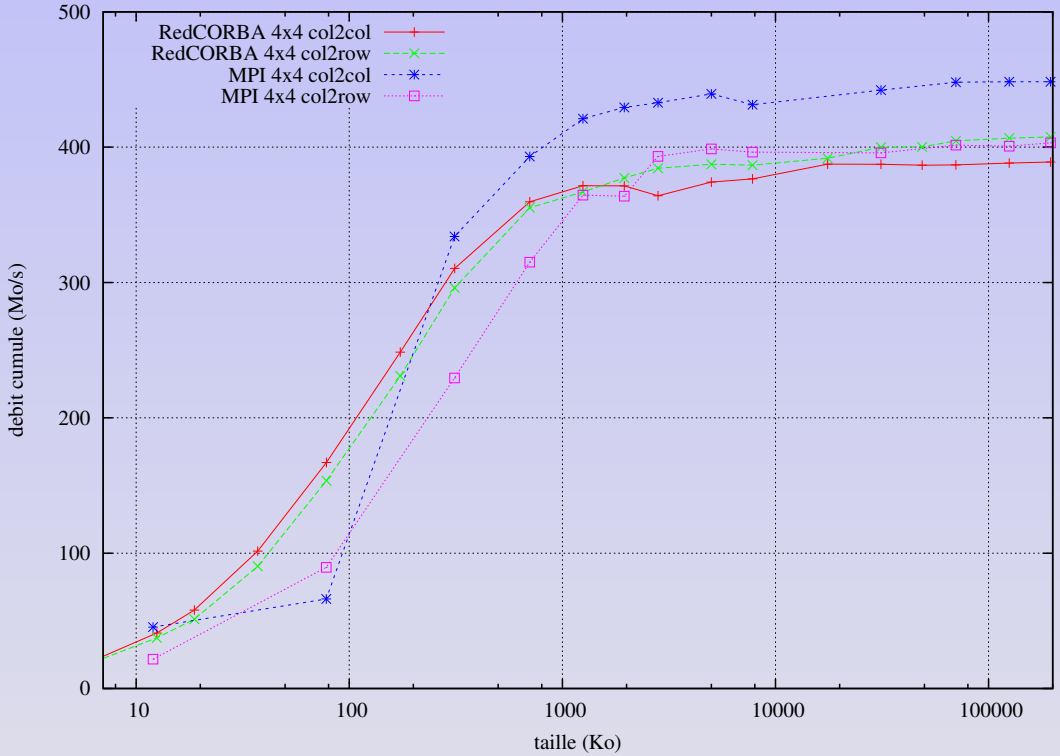
```
<simulation id="particles">
  <data>
     <points id="particles" dimension="3" />
  </data>
  <htg  auto-start="false" >
     <loop id="main"  synchronize="no" >
       <task id="body"  synchronize="no"/>
         <task id="acceleration"  synchronize="no"/>
         <task id="move"  synchronize="no">
           <data-context ref="particles" context="modified"/>
         </task>
       </task>
     </loop>
  </htg>
</simulation>
```

Plan

Performances RedGRID/RedCORBA

Transfert *MxN*

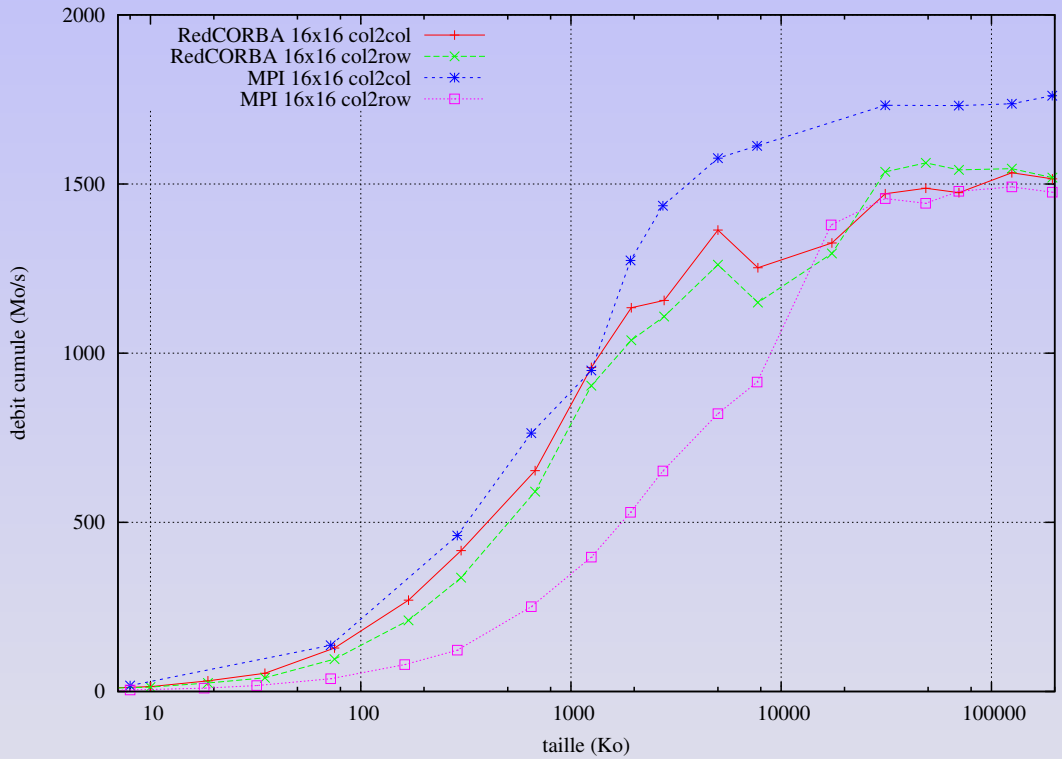
- Field 4x4
- Field 16x16
- Particles 8x8
- Points 8x8
- Points 9x8



Performances RedGRID/RedCORBA

Transfert *MxN*

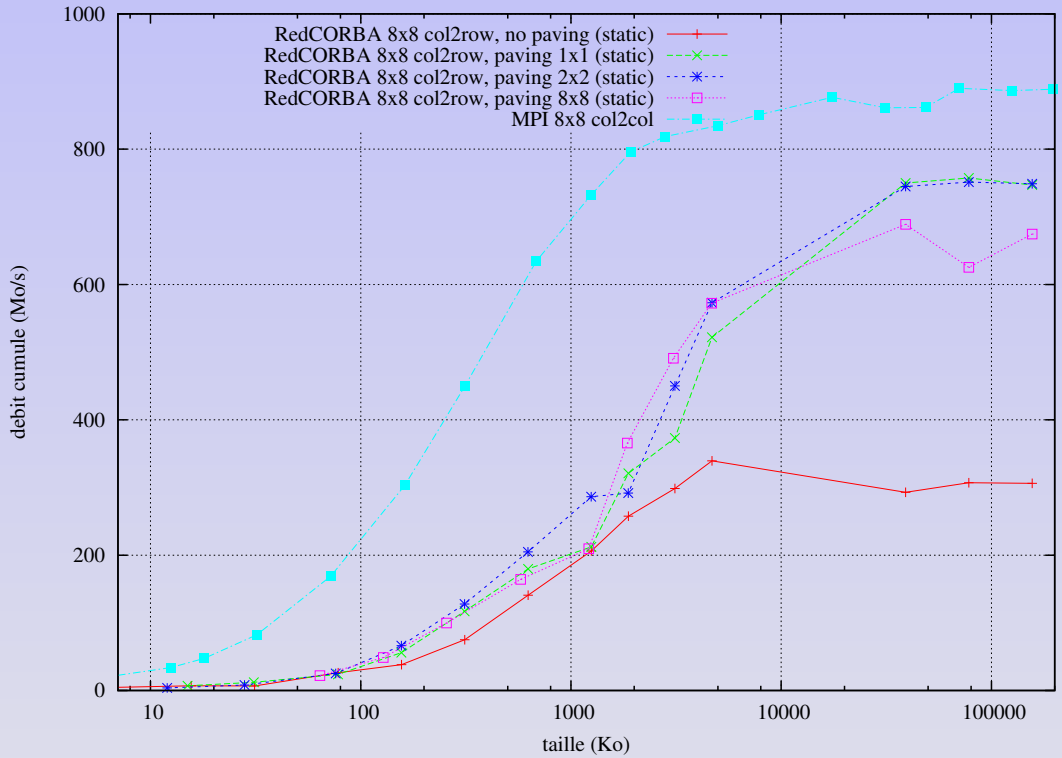
- Field 4x4
- Field 16x16
- Particles 8x8
- Points 8x8
- Points 9x8



Performances RedGRID/RedCORBA

Transfert *MxN*

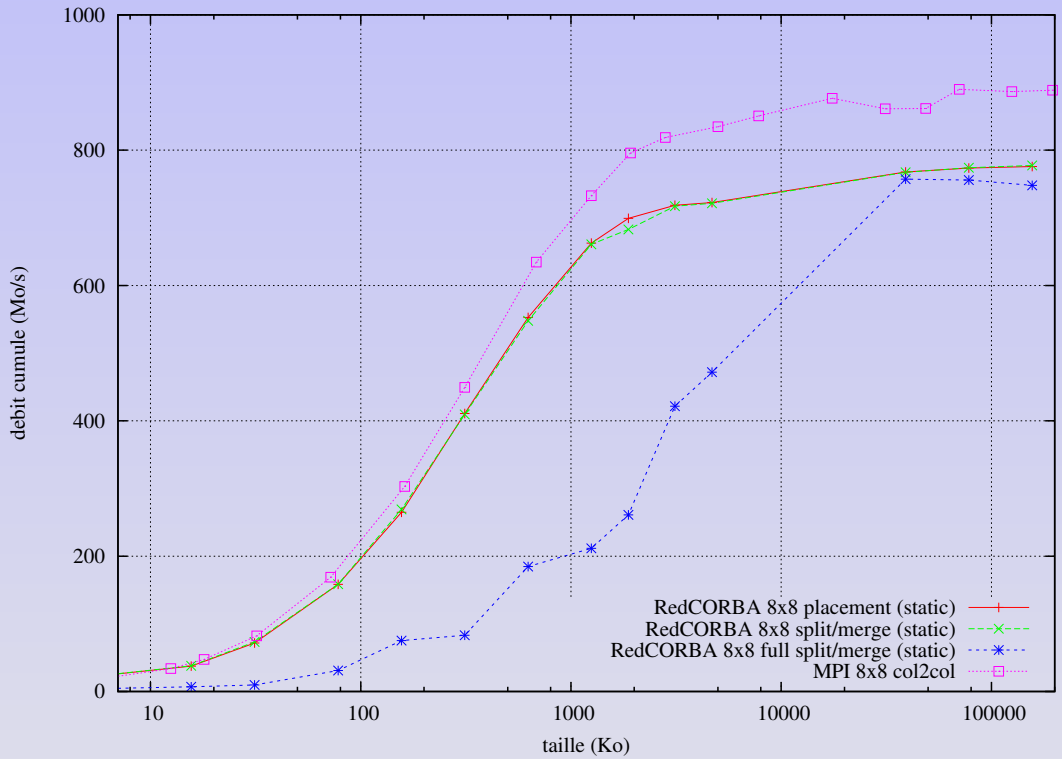
- Field 4x4
- Field 16x16
- Particles 8x8
- Points 8x8
- Points 9x8



Performances RedGRID/RedCORBA

Transfert *MxN*

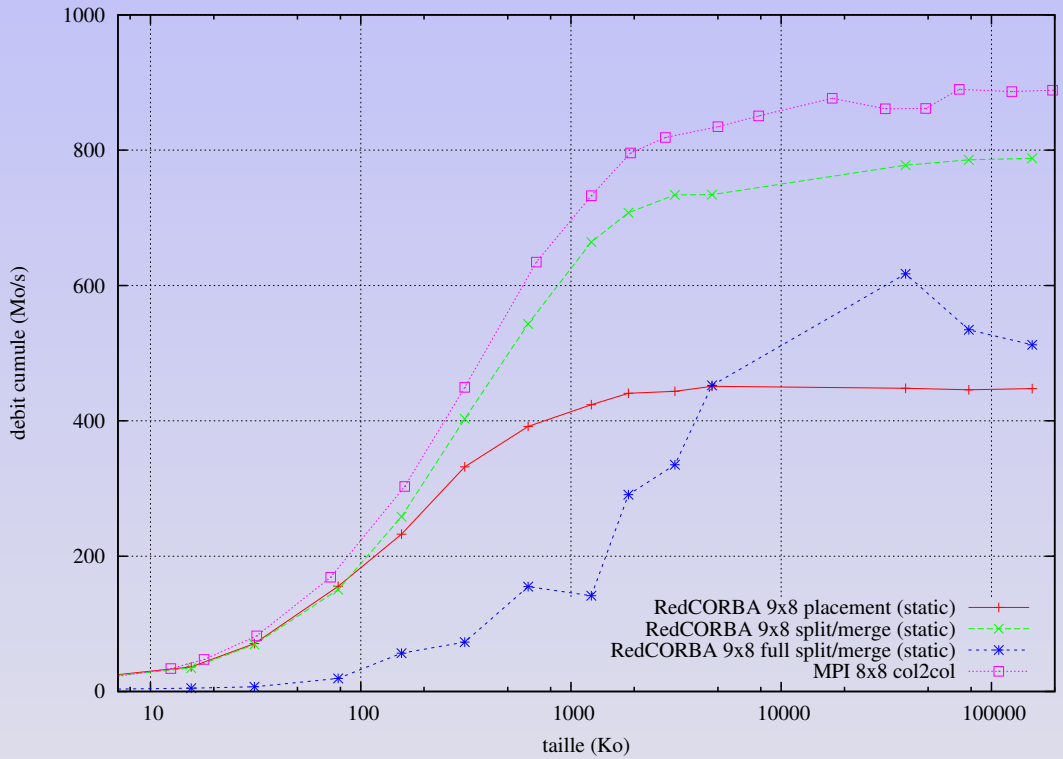
- Field 4x4
- Field 16x16
- Particles 8x8
- Points 8x8
- Points 9x8



Performances RedGRID/RedCORBA

Transfert *MxN*

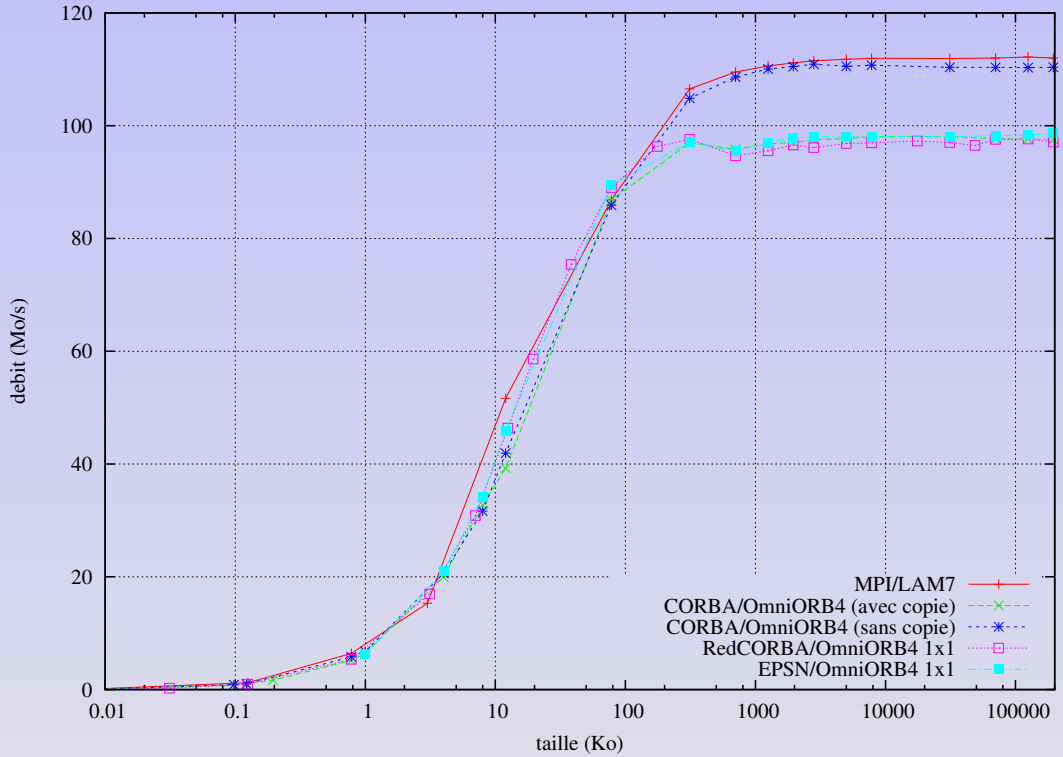
- Field 4x4
- Field 16x16
- Particles 8x8
- Points 8x8
- Points 9x8



Performances EPSN - Recouvrement

1 vers 1

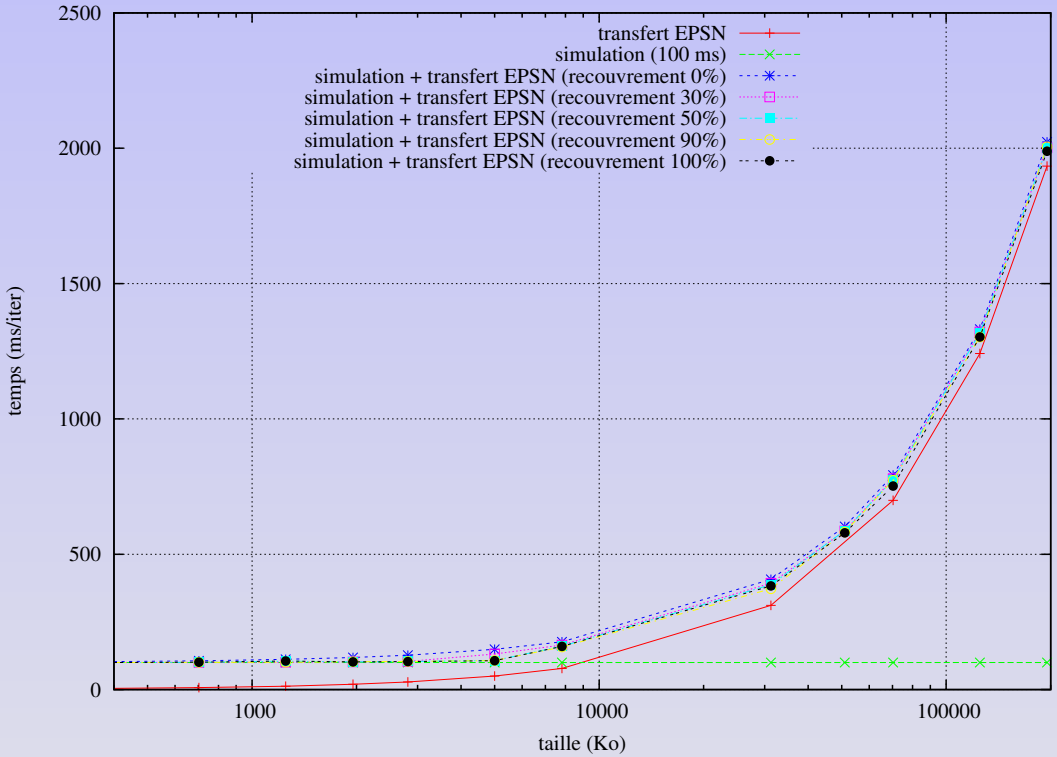
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Recouvrement

1 vers 1

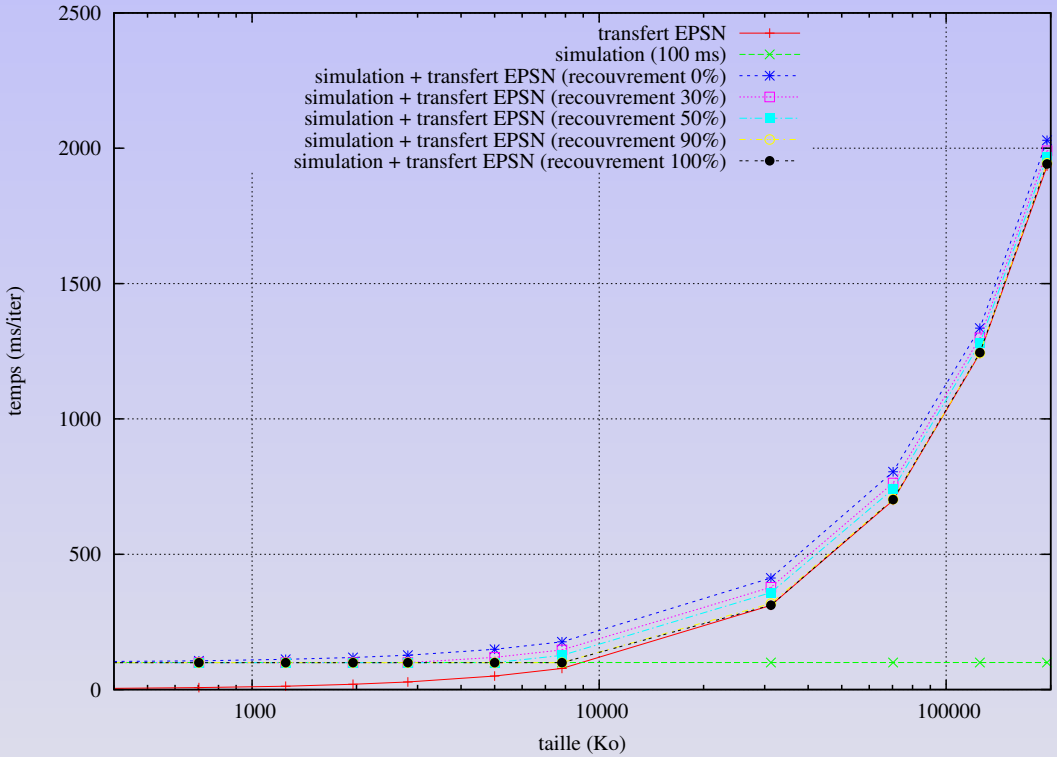
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Recouvrement

1 vers 1

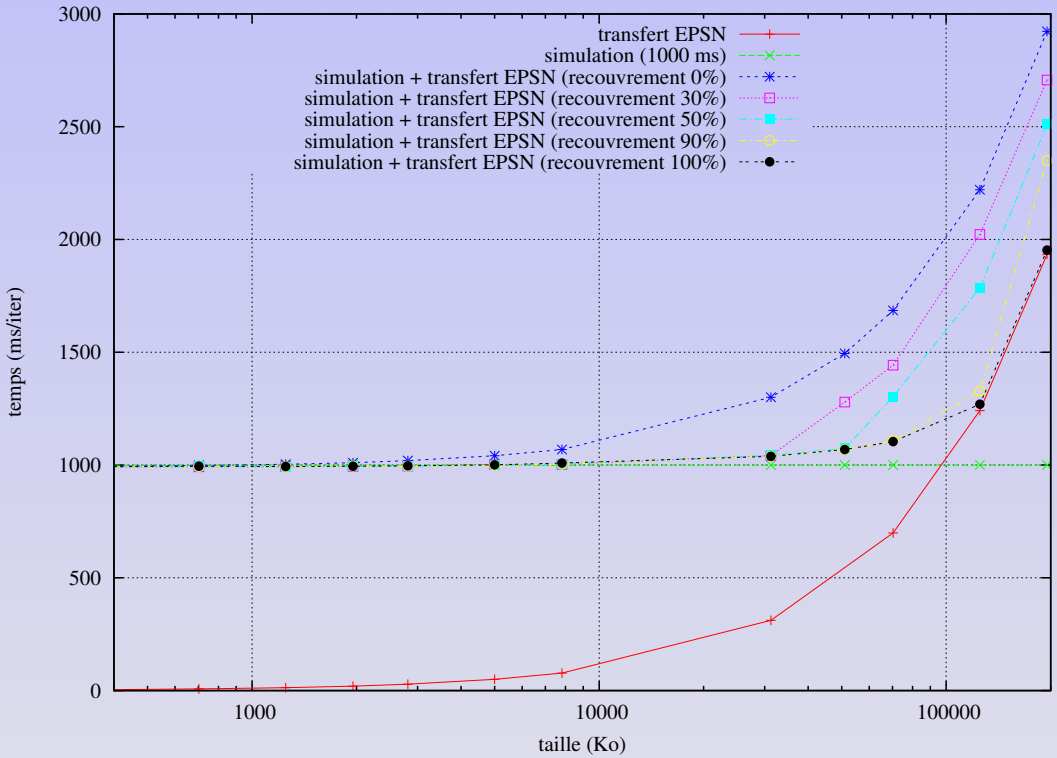
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Recouvrement

1 vers 1

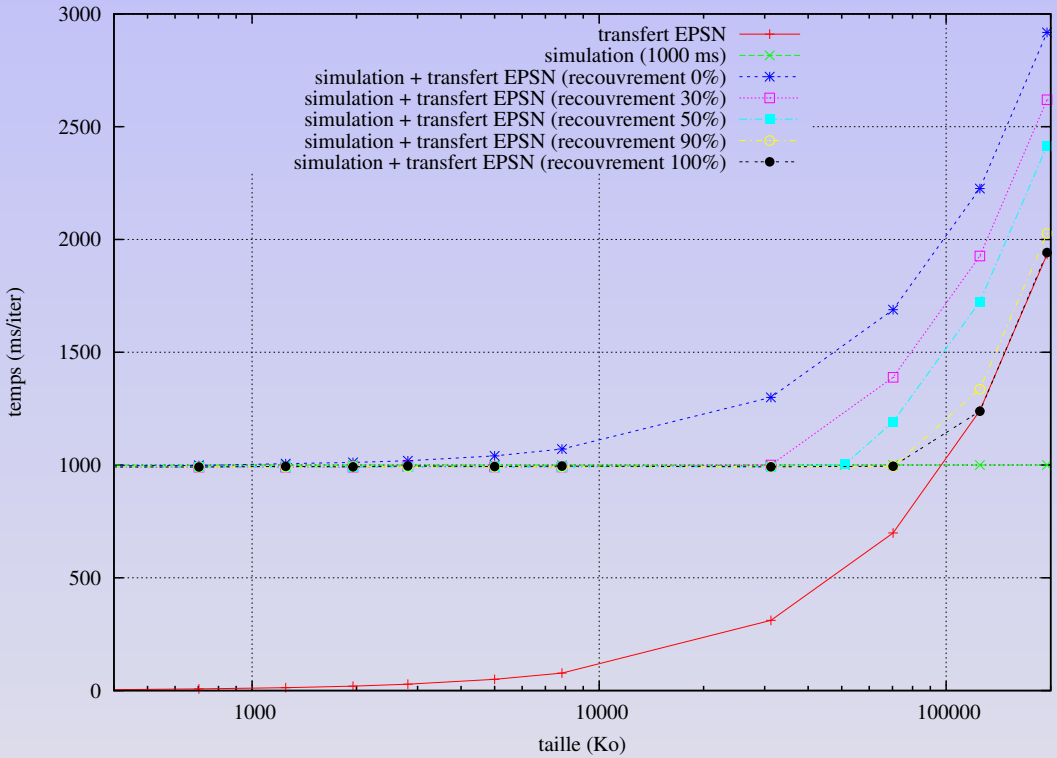
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Recouvrement

1 vers 1

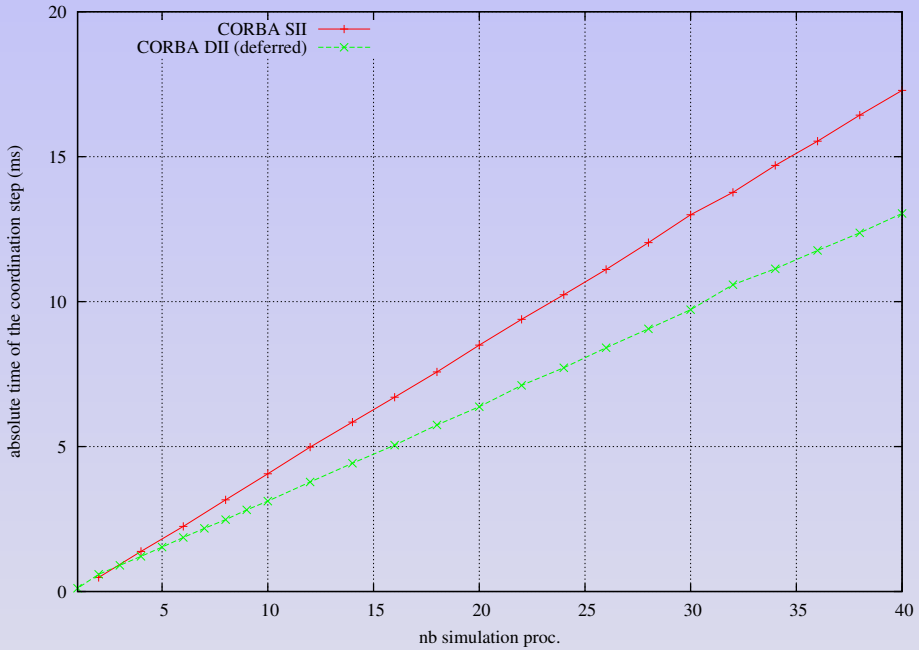
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Lancement de requêtes

transfert des requêtes vers les noeuds

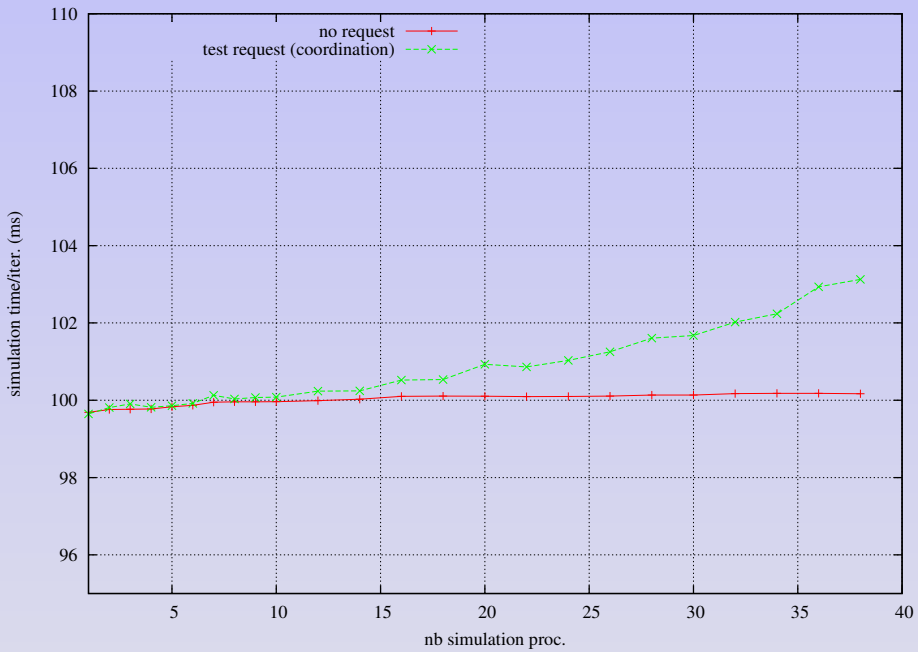
- Linéaire par rapport au nombre de processeur (0.4ms/proc.)
- Recouvert en grande partie par la simulation



Performances EPSN - Lancement de requêtes

transfert des requêtes vers les noeuds

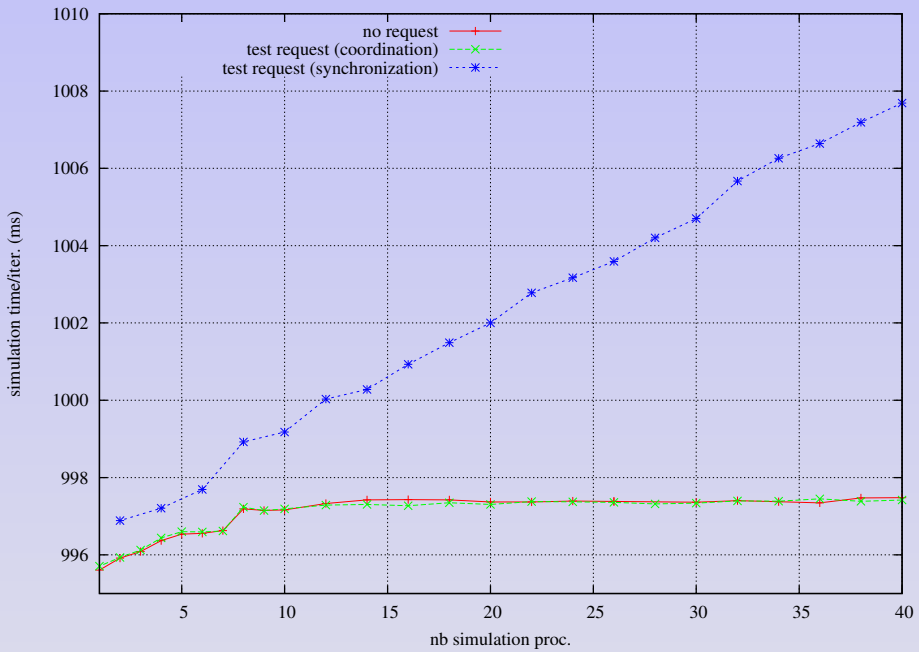
- Linéaire par rapport au nombre de processeur (0.4ms/proc.)
- Recouvert en grande partie par la simulation



Performances EPSN - Lancement de requêtes

transfert des requêtes vers les noeuds

- Linéaire par rapport au nombre de processeur (0.4ms/proc.)
- Recouvert en grande partie par la simulation



Perspectives

Principe

- Extension aux simulations MPMD.
- Extension à la grille dans le cadre de Grid5000.
- Intégration dans Paraview.
- Evolution vers le couplage de simulations parallèles.
- Optimisation des communications avec PadiCo.
- Sauvegarde des données accessibles.

Thèse 2005-2008 ou 2009 ... voire plus

Nicolas Richart travaillera sur *la conception et la mise en oeuvre dans un contexte grid d'une plate-forme de pilotage des simulations numériques distribuées.*

Perspectives

Principe

- Extension aux simulations MPMD.
- Extension à la grille dans le cadre de Grid5000.
- Intégration dans Paraview.
- Evolution vers le couplage de simulations parallèles.
- Optimisation des communications avec PadiCo.
- Sauvegarde des données accessibles.

Thèse 2005-2008 ou 2009 . . . voire plus

Nicolas Richart travaillera sur *la conception et la mise en oeuvre dans un contexte grid d'une plate-forme de pilotage des simulations numériques distribuées*.

Perspectives

Principe

- Extension aux simulations MPMD.
- Extension à la grille dans le cadre de Grid5000.
- Intégration dans Paraview.
- Evolution vers le couplage de simulations parallèles.
- Optimisation des communications avec PadiCo.
- Sauvegarde des données accessibles.

Thèse 2005-2008 ou 2009 . . . voire plus

Nicolas Richart travaillera sur *la conception et la mise en oeuvre dans un contexte grid d'une plate-forme de pilotage des simulations numériques distribuées*.

Perspectives

Principe

- Extension aux simulations MPMD.
- Extension à la grille dans le cadre de Grid5000.
- Intégration dans Paraview.
- Evolution vers le couplage de simulations parallèles.
- Optimisation des communications avec PadiCo.
- Sauvegarde des données accessibles.

Thèse 2005-2008 ou 2009 . . . voire plus

Nicolas Richart travaillera sur *la conception et la mise en oeuvre dans un contexte grid d'une plate-forme de pilotage des simulations numériques distribuées*.

Plan

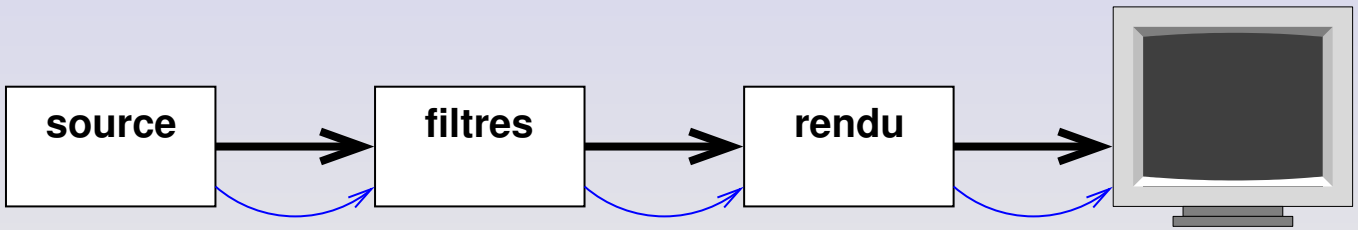
Les systèmes *Event-driven*

Principe

- Modèle *data-flow* basé sur un pipe-line classique
- Une mise-à-jour est lancée automatiquement pour chaque *évènement* dans le pipe-line
  - changement dans les connections du pipe-line
  - modification des paramètres d'un élément
  - modification des données d'une source

Problèmes

⇒ Intégration dans la boucle d'évènement



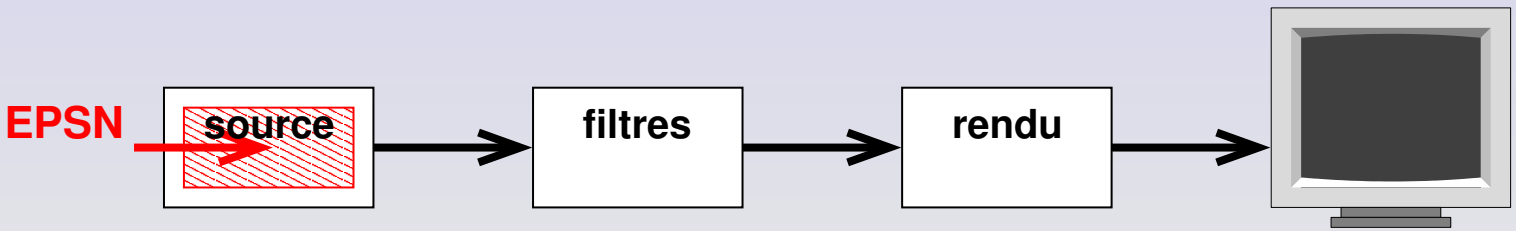
Les systèmes *Event-driven*

Principe

- Modèle *data-flow* basé sur un pipe-line classique
- Une mise-à-jour est lancée automatiquement pour chaque *évènement* dans le pipe-line
  - changement dans les connections du pipe-line
  - modification des paramètres d'un élément
  - modification des données d'une source

Problèmes

⇒ Intégration dans la boucle d'évènement



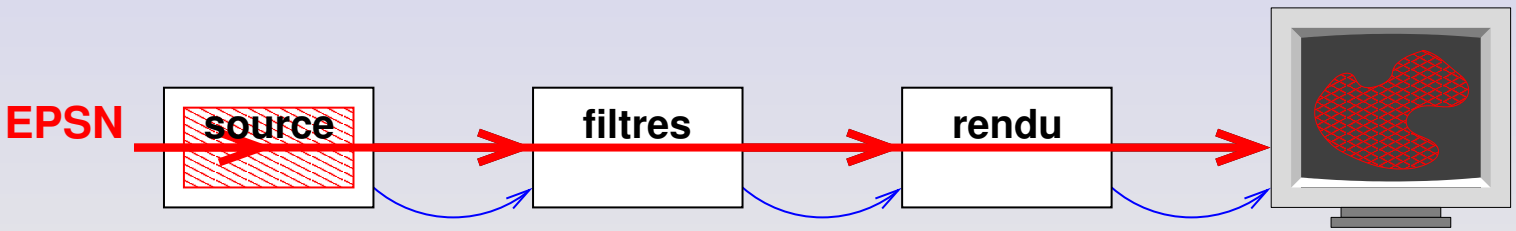
Les systèmes *Event-driven*

Principe

- Modèle *data-flow* basé sur un pipe-line classique
- Une mise-à-jour est lancée automatiquement pour chaque *évènement* dans le pipe-line
  - changement dans les connections du pipe-line
  - modification des paramètres d'un élément
  - modification des données d'une source

Problèmes

⇒ Intégration dans la boucle d'évènement



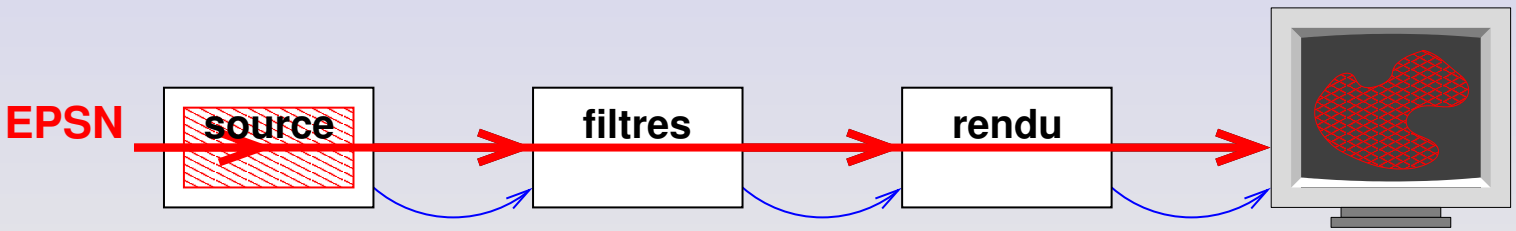
Les systèmes *Event-driven*

Principe

- Modèle *data-flow* basé sur un pipe-line classique
- Une mise-à-jour est lancée automatiquement pour chaque *évènement* dans le pipe-line
  - changement dans les connections du pipe-line
  - modification des paramètres d'un élément
  - modification des données d'une source

Problèmes

⇒ Intégration dans la boucle d'évènement



Les systèmes *Demand-driven*

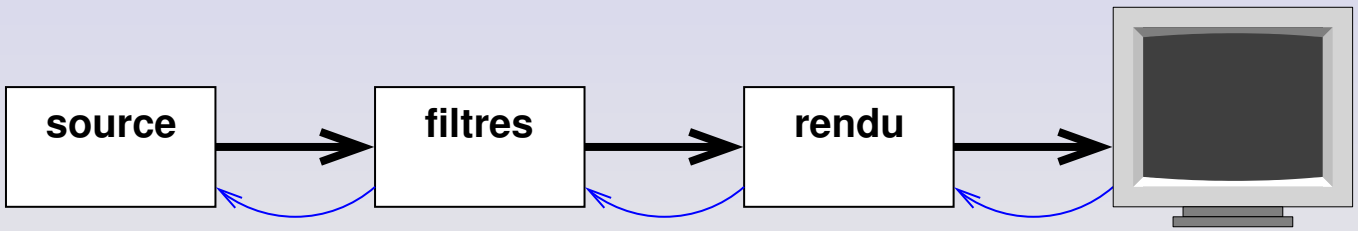
Principe

- Modèle *data-flow* basé sur un pipe-line classique
- L'utilisateur peut demander explicitement une mise-à-jour
- Les mises-à-jours automatique ne sont commandées que par les interactions utilisateur

Problèmes

⇒ Demande explicite de mise à jour

⇒ Intégration dans la boucle d'évènement



Les systèmes *Demand-driven*

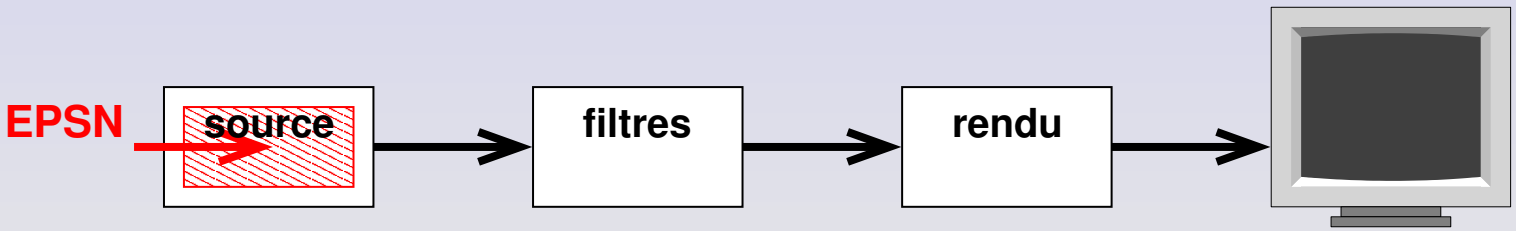
Principe

- Modèle *data-flow* basé sur un pipe-line classique
- L'utilisateur peut demander explicitement une mise-à-jour
- Les mises-à-jours automatique ne sont commandées que par les interactions utilisateur

Problèmes

⇒ Demande explicite de mise à jour

⇒ Intégration dans la boucle d'évènement



Les systèmes *Demand-driven*

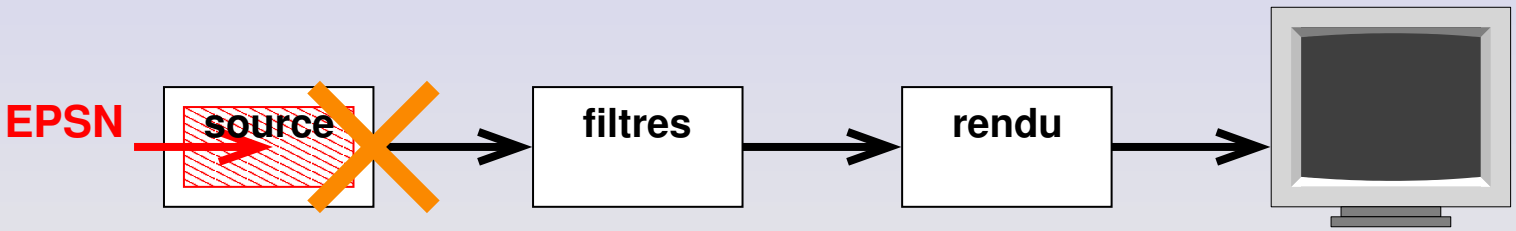
Principe

- Modèle *data-flow* basé sur un pipe-line classique
- L'utilisateur peut demander explicitement une mise-à-jour
- Les mises-à-jours automatique ne sont commandées que par les interactions utilisateur

Problèmes

⇒ Demande explicite de mise à jour

⇒ Intégration dans la boucle d'évènement



Les systèmes *Demand-driven*

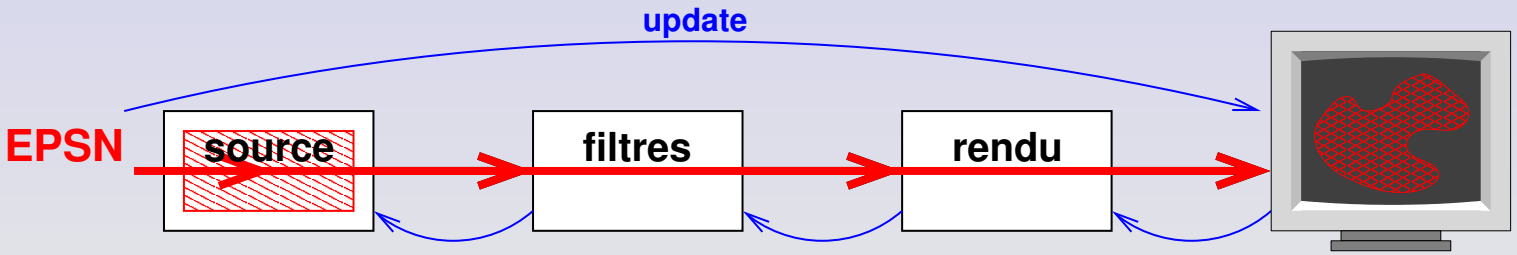
Principe

- Modèle *data-flow* basé sur un pipe-line classique
- L'utilisateur peut demander explicitement une mise-à-jour
- Les mises-à-jours automatique ne sont commandées que par les interactions utilisateur

Problèmes

⇒ Demande explicite de mise à jour

⇒ Intégration dans la boucle d'évènement



Les systèmes *Demand-driven*

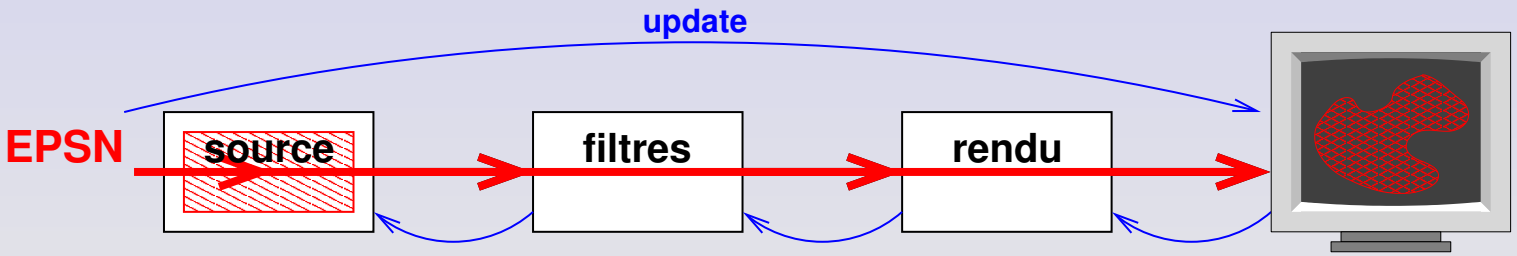
Principe

- Modèle *data-flow* basé sur un pipe-line classique
- L'utilisateur peut demander explicitement une mise-à-jour
- Les mises-à-jours automatique ne sont commandées que par les interactions utilisateur

Problèmes

⇒ Demande explicite de mise à jour

⇒ Intégration dans la boucle d'évènement



Intégration à la boucle d'évènements

Intégration à la boucle d'évènements

- Accès sérialisé au données de la source (lock/unlock)
- Signalement des modifications (modified)

Demande explicite de mise à jour

- Les systèmes de visualisation sont rarement *thread-safe*
- Utilisation d'évènements spécifique au système
- Utilisation d'un *timer* ou d'une fonction *idle*

Source de données pour VTK

RedSYM vs. VTK

- Même système de données
  - objets complexes avec des séries de données associées
  - séries $\leftrightarrow$ listes de *tuples* à plusieurs composantes homogènes
- Partage de la mémoire ou recopie
- Equivalence :
  - champs $\longrightarrow$ `vtkImageData`
  - points $\longrightarrow$ `vtkPoints`
  - maillages $\longrightarrow$ `vtkUnstructuredGrid`
  - séries $\longrightarrow$ `vtkDataArray`

En cours

$\Rightarrow$ Ecritures de sources RedGRID pour VTK

Source de données pour VTK

RedSYM vs. VTK

- Même système de données
  - objets complexes avec des séries de données associées
  - séries $\leftrightarrow$ listes de *tuples* à plusieurs composantes homogènes
- Partage de la mémoire ou recopie
- Equivalence :
  - champs $\longrightarrow$ `vtkImageData`
  - points $\longrightarrow$ `vtkPoints`
  - maillages $\longrightarrow$ `vtkUnstructuredGrid`
  - séries $\longrightarrow$ `vtkDataArray`

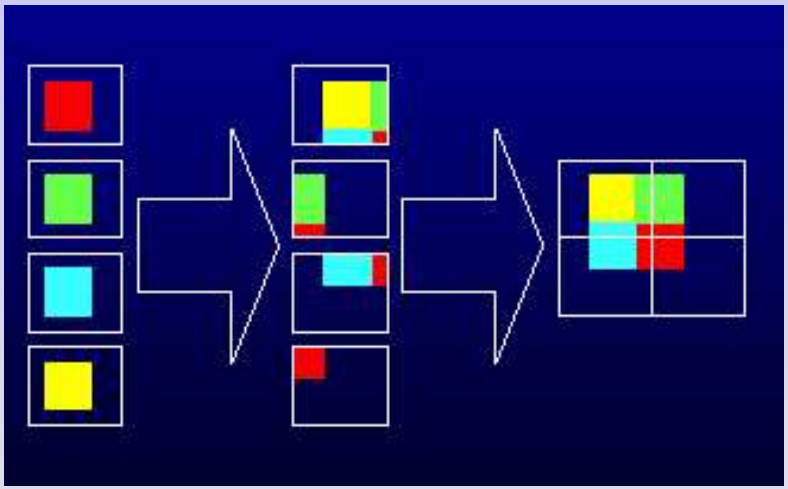
En cours

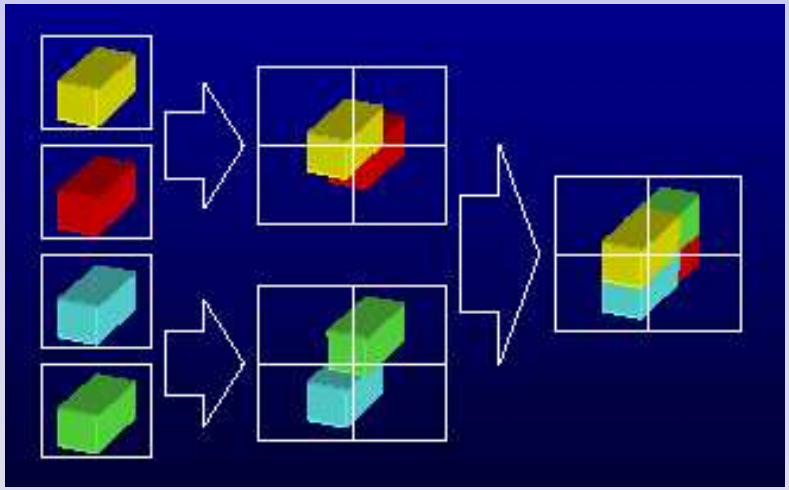
$\Rightarrow$ Ecritures de sources RedGRID pour VTK

Système de visualisation parallèle

Différentes techniques

- Réplication des données
- Sort-first : transfert des primitives graphiques
- Sort-last : transfert et composition des images





Intégration dans un système de visualisation parallèle

Principe

- Architecture maître / esclaves
- Données distribuées
- Pipe-lines parallèles