

L'environnement EPSN

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

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

October 4, 2005

- 1 **Modèle**
 - Modèle de description
 - Modèle de pilotage
- 2 La plate-forme EPSN
 - Architecture
 - Fonctionnalités
 - Instrumentation d'une simulation
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

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 propriétés de la simulation qui sont révélées 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 propriétés de la simulation qui sont révélées 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

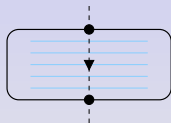
- Flexible
- Les modifications de la description externe ne nécessitent pas une nouvelle instrumentation
- L'instrumentation reste simple
- La généricité permet la gestion du parallélisme et d'opération de haut niveau

- 1 **Modèle**
 - **Modèle de description**
 - Modèle de pilotage
- 2 La plate-forme EPSN
 - Architecture
 - Fonctionnalités
 - Instrumentation d'une simulation
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

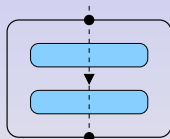
Grphe de Taches Hiérarchiques - HTG

HTG

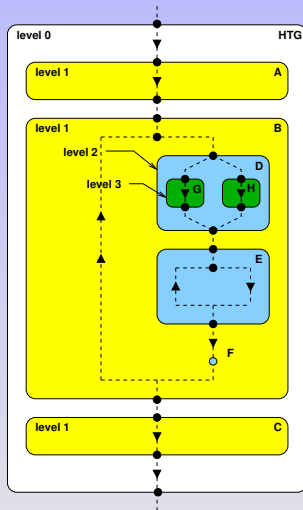
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



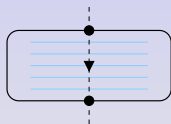
tâche composée



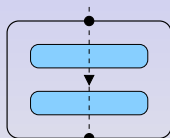
Grphe de Taches Hiérarchiques - HTG

HTG

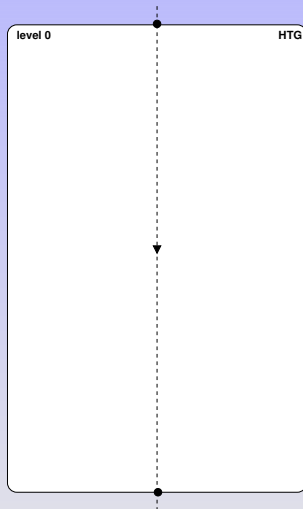
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



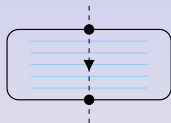
tâche composée



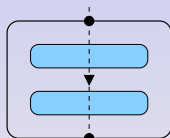
Grphe de Taches Hiérarchiques - HTG

HTG

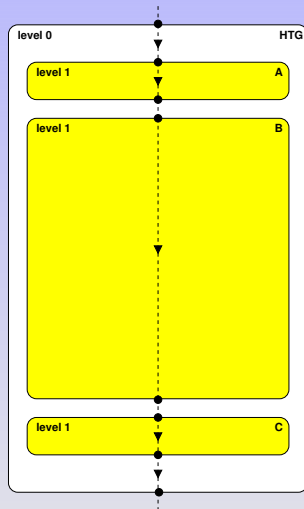
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



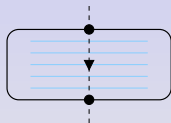
tâche composée



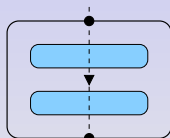
Grphe de Taches Hiérarchiques - HTG

HTG

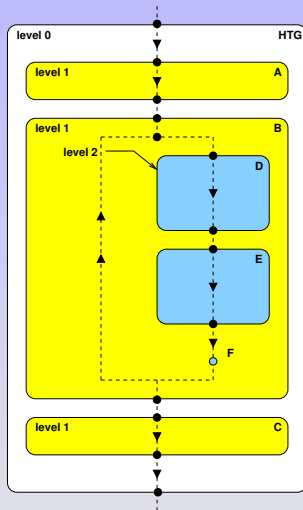
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple



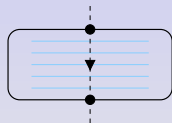
tâche composée



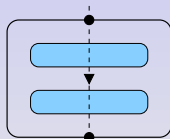
Grphe de Taches Hiérarchiques - HTG

HTG

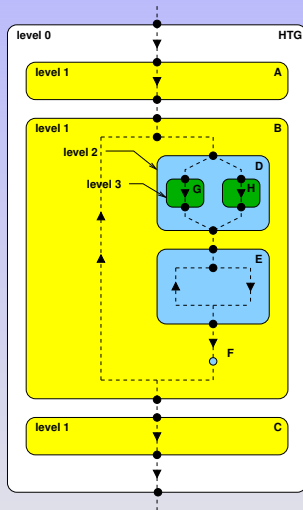
- Graphe dirigé acyclique
- Les noeuds encapsulent du code
- Les noeuds peuvent contenir un HTG
- Les arêtes suivent le flux de contrôle



tâche simple

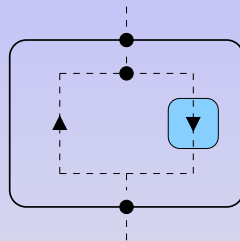


tâche composée



Propriétés

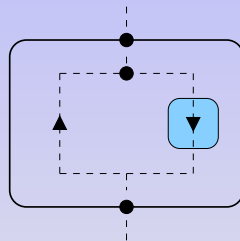
- Tache hiérarchique contenant une seule sous-tache
- La sous-tache peut être parcouru plusieurs fois
- Garantit le caractère acyclique du graphe
- Modélise les structures itératives: `do`, `for`, `while`
- Ne peut modéliser les renvois type `goto`



tâche en boucle

Propriétés

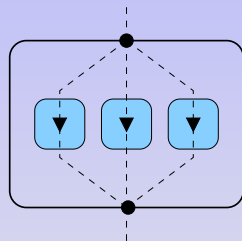
- Tache hiérarchique contenant une seule sous-tache
- La sous-tache peut être parcouru plusieurs fois
- Garantit le caractère acyclique du graphe
- Modélise les structures itératives: `do`, `for`, `while`
- **Ne peut modéliser les renvois type `goto`**



tâche en boucle

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 tâche courante présentent la même cohérence.

Propriétés pour le parallélisme

- **acyclique**
⇒ *datation* précise de l'évolution des processus
- **hiérarchique**
⇒ 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 tâche courante présentent la même cohérence.

Propriétés pour le parallélisme

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

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

- 1 Modèle
 - Modèle de description
 - **Modèle de pilotage**
- 2 La plate-forme EPSN
 - Architecture
 - Fonctionnalités
 - Instrumentation d'une simulation
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

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

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

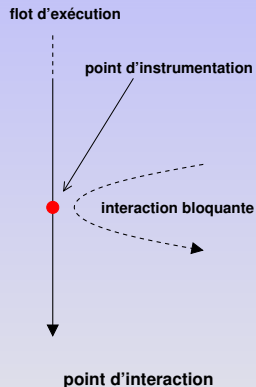
Points et plages d'interaction

Point d'interaction

- Contrôle du flux d'exécution
- Support d'interactions bloquantes
- Correspond à un point d'instrumentation: début/fin de tâche, tâche 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
- ⇒ Permet du recouvrement



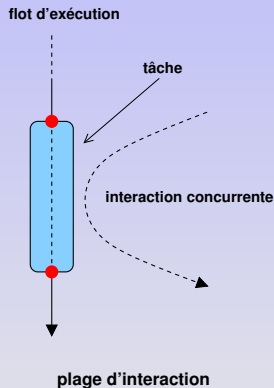
Points et plages d'interaction

Point d'interaction

- Contrôle du flux d'exécution
- Support d'interactions bloquantes
- Correspond à un point d'instrumentation: début/fin de tâche, tâche 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
- ⇒ Permet du recouvrement



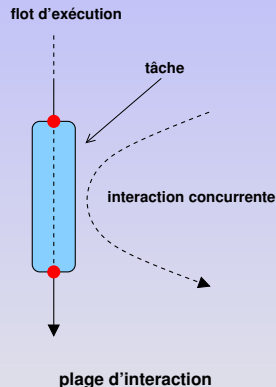
Points et plages d'interaction

Point d'interaction

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

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 ?

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

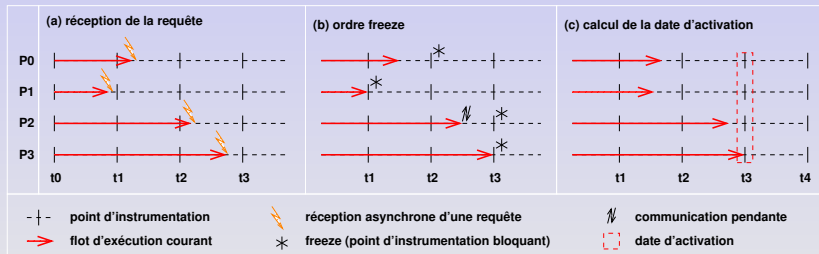
Problème

Comment déterminer une date pour la planification ?

Planification à la volée

Principe

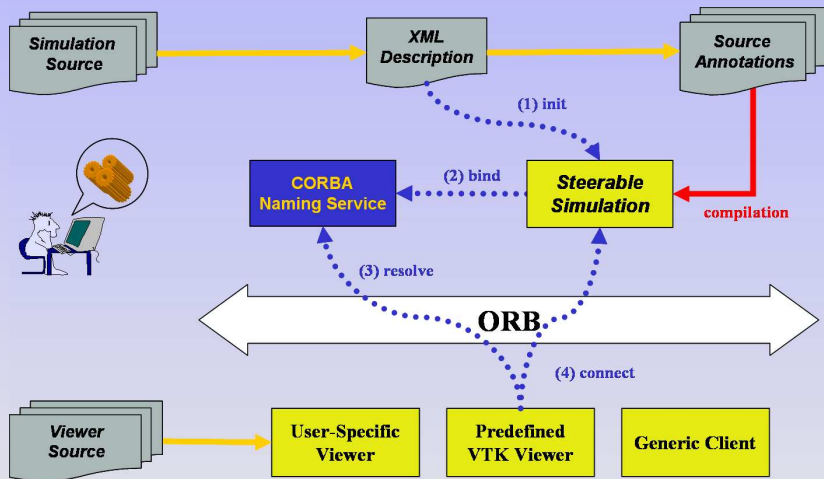
- Déterminer une date commune qu'aucun processus n'a encore dépassé
- Ajouter un critère de validité pour le début du traitement
- Comme la simulation est SPMD on obtient une date de planification valide



- 1 Modèle
 - Modèle de description
 - Modèle de pilotage
- 2 La plate-forme EPSN
 - Architecture
 - Fonctionnalités
 - Instrumentation d'une simulation
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

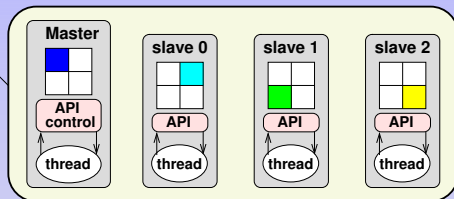
- 1 Modèle
 - Modèle de description
 - Modèle de pilotage
- 2 La plate-forme EPSN
 - **Architecture**
 - Fonctionnalités
 - Instrumentation d'une simulation
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

L'intégration

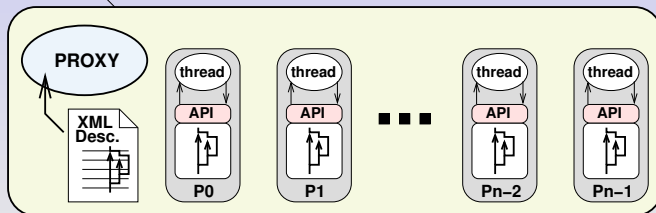


Architecture distribuée

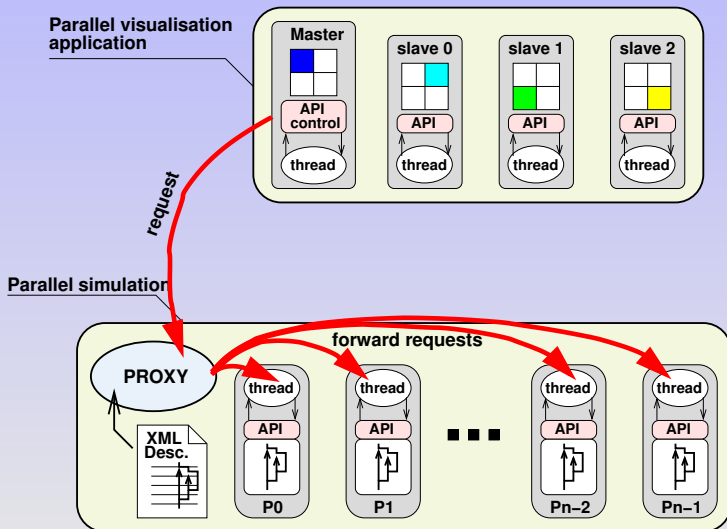
Parallel visualisation application



Parallel simulation

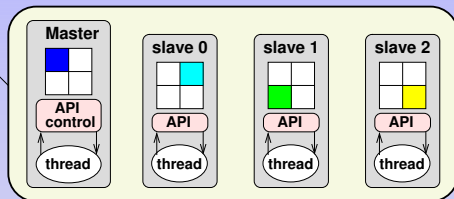


Architecture distribuée

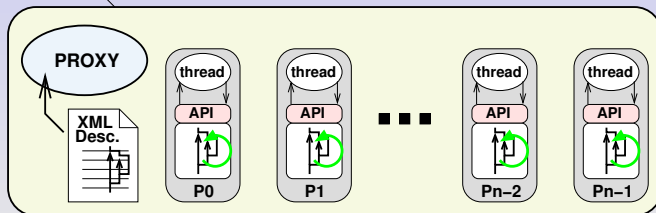


Architecture distribuée

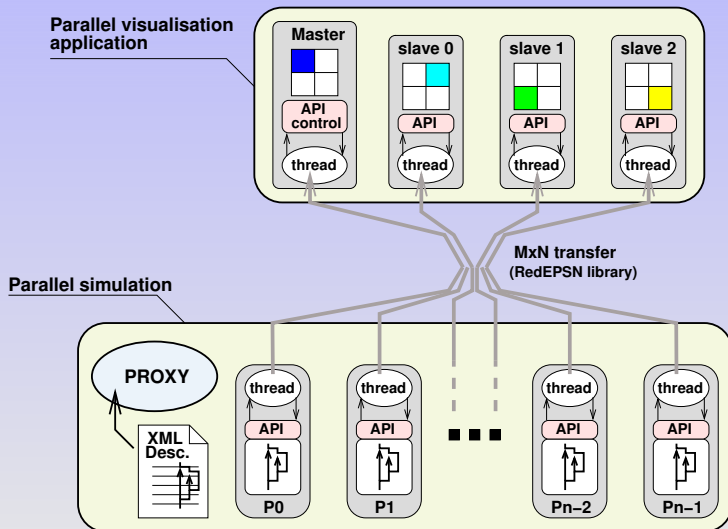
Parallel visualisation application



Parallel simulation



Architecture distribuée

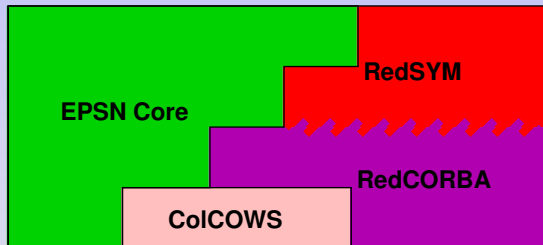


Les bibliothèques associées

Applications de Pilotage

EPSN Front-End + Back-End APIs

RedSYM API



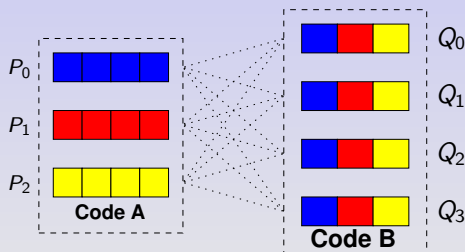
CORBA

Bibliothèque RedSYM

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

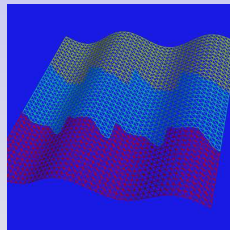
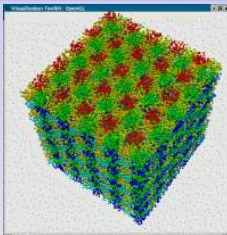
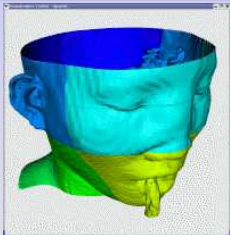
Bibliothèque RedCORBA

- Transfert des blocs de redistribution
- Communication via CORBA



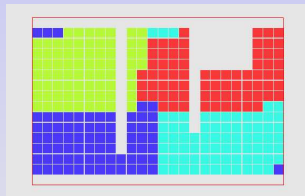
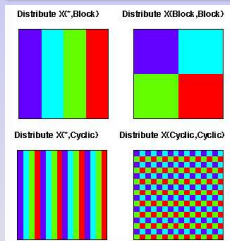
Modèle de données

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



Modèle de distribution

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



- 1 Modèle
 - Modèle de description
 - Modèle de pilotage
- 2 La plate-forme EPSN
 - Architecture
 - **Fonctionnalités**
 - Instrumentation d'une simulation
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

Contrôle

- Stop
- Play
- Step

Accès aux données

- Get
- Put

Déclenchement d'actions

- Déclenchement d'actions

Facilités

- Benchmark
- Date courante
- Listes de requêtes

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

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 tâches

Commandes

- Benchmark
 - récupération ponctuelle des temps d'exécution d'une tâche
 - 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 . . .)

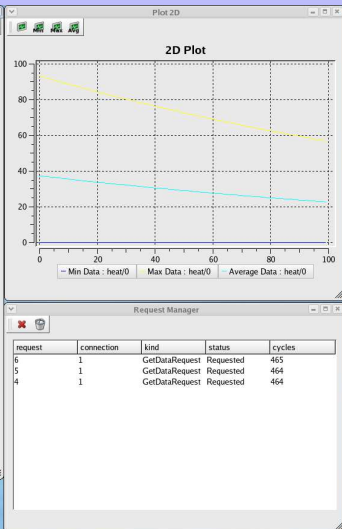
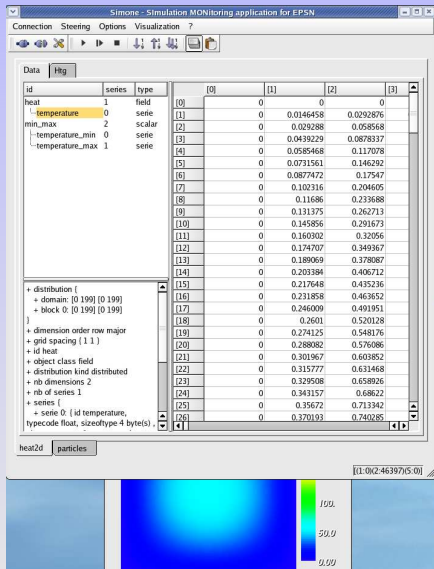
Fonctionnalités XML

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

The screenshot displays the Simone - Simulation MONITORING application for EPSN. It features several windows:

- Main Window:** Shows a task graph on the left with a tree view of tasks (begin, end, children, main, points, iterate, update, swap) and a central diagram of a loop containing 'update' and 'swap' tasks, with an 'after_update' label and a red arrow indicating a flow.
- Plot 2D:** A line graph showing three data series over 100 cycles. The Y-axis ranges from 0 to 100. The series are:
 - Min Data: heat/0 (blue line, constant at 0)
 - Average Data: heat/0 (cyan line, decreasing from ~38 to ~22)
 - Max Data: heat/0 (yellow line, decreasing from ~92 to ~60)
- VTK 2D Field:** A heatmap showing a circular field with a color scale from 0.00 (blue) to 200.0 (red).
- Request Manager:** A table listing requested data requests.

request	connection	kind	status	cycles
6	1	GetDataRequest	Requested	465
5	1	GetDataRequest	Requested	464
4	1	GetDataRequest	Requested	464



Simone - Simulation MONITORING application for EPSN

Connection Steering Options Visualization ?

Data [Htg]

id	series	type	[71]	[72]	[73]	
heat	1	field	[66] 1.5387	44.8677	45.1852	45.491
temperature	0	serie	[67] 1.9424	45.2744	45.5948	45.9034
min_max	2	scalar	[68] 1.3348	45.6698	45.9929	46.3042
temperature_min	0	serie	[69] 1.7159	46.0536	46.3795	46.6934
temperature_max	1	serie	[70] 1.0854	46.4259	46.7544	47.0709
			[71] 1.4434	46.7865	47.1176	47.4366
			[72] 1.7897	47.1354	47.469	47.7903
			[73] 1.1242	47.4724	47.8084	48.1321
			[74] 1.7447	47.7976	48.1358	48.4617
			[75] 1.7578	48.1107	48.4512	48.7792
			[76] 1.0566	48.4117	48.7543	49.0844
			[77] 1.3433	48.7005	49.0452	49.3773
			[78] 1.6179	48.9772	49.3238	49.6578
			[79] 1.8802	49.2415	49.5901	49.9259
			[80] 1.1303	49.4934	49.8438	50.1813
			[81] 1.3681	49.733	50.085	50.4242
			[82] 1.5934	49.96	50.3137	50.6544
			[83] 1.8063	50.1745	50.5297	50.872
			[84] 1.0067	50.3764	50.7331	51.0767
			[85] 1.1945	50.5656	50.9236	51.2686
			[86] 1.3698	50.7422	51.1015	51.4476
			[87] 1.5324	50.906	51.2664	51.6137
			[88] 1.6823	51.057	51.4186	51.7669
			[89] 1.8194	51.1952	51.5578	51.9071
			[90] 1.9439	51.3206	51.684	52.0342
			[91] 1.0555	51.433	51.7973	52.1483
			[92] 1.1543	51.5326	51.8977	52.2493

heat2d particles [L:0;2:46397;S:0]

Plot 2D

2D Plot

Request Manager

request	connection	kind	status	cycles
6	1	GetDataRequest	Requested	465
5	1	GetDataRequest	Requested	464
4	1	GetDataRequest	Requested	464

Simone - Simulation MONITORING application for EPSN

Connection Steering Options Visualization ?

Data Hitg

id	series	type	[71]	[72]	[73]	
heat	1	field	[66] 1.5387	44.8677	45.1852	45.491
temperature	0	serie	[67] 1.9424	45.2744	45.5948	45.9034
min_max	2	scalar	[68] 300	300	300	300
temperature_min	0	serie	[69] 300	300	300	300
temperature_max	1	serie	[70] 300	300	300	300
[71]			300	300	300	300
[72]			300	300	300	300
[73]			300	300	300	300
[74]			300	300	300	300
[75]			300	300	300	300
[76]			300	300	300	300
[77]			300	300	300	300
[78]			300	300	300	300
[79]			300	300	300	300
[80]			300	300	300	300
[81]			300	300	300	300
[82]			300	300	300	300
[83]			300	300	300	300
[84]			300	300	300	300
[85]			300	300	300	300
[86]			300	300	300	300
[87]			300	300	300	300
[88]			1.6823	51.057	51.4186	51.7669
[89]			1.8194	51.1952	51.5578	51.9071
[90]			1.9439	51.3206	51.684	52.0342
[91]			1.0555	51.433	51.7973	52.1483
[92]			1.1543	51.5326	51.8977	52.2493

+ distribution {
 + domain: [0 199] [0 199]
 + block 0: [0 199] [0 199]
 }
 + dimension order row major
 + nb of series 1
 + id heat
 + object class field
 + distribution kind distributed
 + nb dimensions 2
 + nb of series 1
 + series {
 + serie 0: (id temperature,
 typecode float, sizedtype 4 byte(s))
 }

heat2d particles

[L:0;2:46397;S:0]

Plot 2D

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
9	1	PutDataRequest	Complete	1
6	1	GetDataRequest	Requested	465
5	1	GetDataRequest	Requested	464
4	1	GetDataRequest	Requested	464

The screenshot displays the Simone - Simulation MONITORING application for EPSN. It consists of several windows:

- Main Window:** Shows a task graph on the left with a tree view of tasks (hg, points, begin, end, children, main, loop, update, swap) and a central visualization area showing a loop with 'update' and 'swap' tasks and an 'after_update' label.
- Plot 2D:** A line graph showing 'heat/0' data over 100 cycles. It features three lines: Min Data (blue), Max Data (yellow), and Average Data (cyan). The Y-axis ranges from 0 to 100.
- VTK 2D Field:** A 2D heatmap visualization showing a circular gradient from blue (0.00) to red (200.00).
- Request Manager:** A table listing requests with columns for request, connection, kind, status, and cycles.

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

The screenshot displays the Simone - Simulation Monitoring application for EPSN, which is divided into several panels:

- Task Graph (Left):** A hierarchical tree view showing the simulation's structure. The root is 'hsg' (task), which contains 'begin' (BeginPoint) and 'end' (EndPoint). It has two children: 'main' (loop) and 'update' (task). 'main' contains 'begin' (BeginPoint), 'iterate' (IteratePoint), and 'end' (EndPoint). 'update' contains 'begin' (BeginPoint), 'end' (EndPoint), and 'swap' (task). 'swap' contains 'points' (task). A central diagram shows a loop with 'update' and 'swap' tasks, with an arrow indicating a flow from 'update' to 'swap' and back to 'update'.
- 2D Plot (Top Right):** A line graph titled '2D Plot' showing data over 100 cycles. The y-axis ranges from 0 to 250. Three data series are plotted: 'Min Data: heat/0' (blue line, constant at 0), 'Max Data: heat/0' (yellow line, decreasing from ~100 to ~60), and 'Average Data: heat/0' (cyan line, decreasing from ~40 to ~30).
- VTK 2D Field (Bottom Left):** A 2D heatmap showing a localized high-temperature region (red/yellow) in the center of a blue field. A color scale on the right ranges from 0.00 to 200.0. The coordinates are [1:0](2:46531)(3:0).
- Request Manager (Bottom Right):** A table listing simulation requests.

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

The image displays the Simone - Simulation Monitoring application for EPSN, showing four main components:

- Task Graph (Left):** A hierarchical tree view of tasks. The 'update' task is highlighted in yellow. Below the tree, parameters are listed:


```
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
```
- 2D Plot (Top Right):** A line graph titled '2D Plot' showing 'heat/0' over 100 cycles. The y-axis ranges from 0 to 250. Three data series are shown: Min Data (blue), Max Data (yellow), and Average Data (cyan). The Max Data starts at approximately 100 and decreases to about 60, then spikes to 250 at the end. The Average Data starts at approximately 40 and decreases to about 30. The Min Data remains near 0.

Cycle	Min Data (heat/0)	Average Data (heat/0)	Max Data (heat/0)
0	0	40	100
20	0	35	80
40	0	32	70
60	0	30	65
80	0	28	60
100	0	30	250
- VTK 2D Field (Bottom Left):** A 2D heatmap showing a localized high-temperature region (yellow/red) in the center of a blue field. A color scale on the right ranges from 0.00 to 200.0.

Color	Temperature
Blue	0.00
Cyan	50.0
Green	100.0
Yellow	150.0
Red	200.0
- Request Manager (Bottom Right):** A table showing the status of requests.

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

The screenshot displays the Simone - Simulation Monitoring application for EPSN, which is used for monitoring and visualizing simulation tasks.

Task Graph (Left Panel): A hierarchical tree view shows the task structure. The root task is 'hsg' (task), which contains 'begin' (BeginPoint) and 'end' (EndPoint) markers, and a 'children' list. The 'children' list includes a 'main' loop (loop) and an 'update' task (task). The 'update' task has its own 'begin' (BeginPoint), 'end' (EndPoint), and 'children' list, which includes 'points' (points), 'swap' (task), and 'after_update' (task). A diagram below the tree shows a loop with 'update' and 'swap' tasks, with an arrow indicating the flow from 'update' to 'swap' and back to 'update'.

2D Plot (Top Right Panel): A line graph titled '2D Plot' showing the evolution of 'heat/0' over 100 cycles. The y-axis ranges from 0 to 250. Three data series are shown: 'Min Data : heat/0' (blue line, constant at 0), 'Max Data : heat/0' (yellow line, decreasing from ~90 to ~60), and 'Average Data : heat/0' (cyan line, decreasing from ~35 to ~25). A sharp spike in the Max Data series occurs at the end of the simulation.

VTK 2D Field (Bottom Left Panel): A visualization of the 'heat2d' field, showing a circular region of high temperature (red/yellow) in the center of a blue field. The color scale ranges from 0.00 to 200.0.

Request Manager (Bottom Right Panel): A table showing the status of various requests:

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

The image displays the Simone - Simulation Monitoring application for EPSN, showing several interconnected windows:

- Main Window:** Displays a task graph on the left and a central visualization area. The graph shows a hierarchy of tasks: 'hsg' (task) contains 'begin' (BeginPoint) and 'end' (EndPoint); 'children' contains 'main' (loop); 'main' contains 'begin' (BeginPoint), 'iterate' (IteratePoint), and 'end' (EndPoint); 'children' contains 'update' (task), which includes 'begin' (BeginPoint), 'end' (EndPoint), and 'swap' (task); 'swap' contains 'points'.
- 2D Plot:** A line graph titled '2D Plot' showing 'heat/0' data over 100 cycles. The y-axis ranges from 0 to 250. Three lines are plotted: 'Min Data' (blue), 'Max Data' (yellow), and 'Average Data' (cyan). The Max Data line starts at approximately 90 and gradually decreases to about 60, then spikes sharply to 230 at cycle 95. The Average Data line starts at approximately 40 and remains relatively stable around 30. The Min Data line is constant at 0.
- VTK 2D Field:** A heatmap window titled 'VTK 2D Field' showing a circular field of values. The color scale ranges from 0.00 (blue) to 200.0 (red). The field shows a bright green/yellow spot in the center, indicating high values, surrounded by a blue region.
- Request Manager:** A table window showing the status of various requests.

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

The screenshot displays the Simone - Simulation MONITORING application for EPSN. The interface is divided into several panels:

- Task Graph (Left):** A hierarchical tree view showing the simulation structure. The 'update' task is highlighted in yellow. Below the tree, parameters are listed: `id : update`, `time_sensor_frequency : 100`, `nb points : 2`, and `nb child tasks : 0`.
- Task Execution Diagram (Center):** A flow diagram showing the execution of the 'update' task. It includes sub-tasks 'update' (red box), 'swap' (blue box), and 'after_update' (black box), connected by arrows in a loop.
- 2D Plot (Top Right):** A line graph titled '2D Plot' showing 'heat/0' data over 100 cycles. The y-axis ranges from 0 to 250. Three lines are plotted: 'Min Data : heat/0' (blue), 'Max Data : heat/0' (yellow), and 'Average Data : heat/0' (cyan). The 'Max Data' line shows a sharp spike near the end of the simulation.
- VTK 2D Field (Bottom Left):** A visualization of a 2D field with a color scale from 0.00 (blue) to 200.0 (red). A bright spot is visible in the center of the field.
- Request Manager (Bottom Right):** A table listing simulation requests.

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

The screenshot displays the Simone - Simulation MONITORING application for EPSN. The main window is divided into several panels:

- Task Graph:** A hierarchical tree on the left shows the task structure. The 'update' task is highlighted in yellow. A central diagram shows a loop with 'update' and 'swap' tasks, with an arrow indicating the flow from 'update' to 'swap' and back to 'update'.
- 2D Plot:** A line graph titled '2D Plot' showing 'heat/0' over time. The y-axis ranges from 0 to 250, and the x-axis from 0 to 100. A yellow line shows a sharp peak around x=85, reaching approximately 230. A cyan line shows a steady decline from about 80 to 20. A blue line remains near 0.
- VTK 2D Field:** A heatmap showing a circular distribution of 'heat/0' values, with a color scale from 0.00 (blue) to 200.0 (red).
- Request Manager:** A table listing requests with their connection, kind, status, and cycles.

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

The screenshot displays the Simone - Simulation MONITORING application for EPSN. The interface is divided into several panels:

- Task Graph (Left):** A hierarchical tree view showing the simulation structure. The 'update' task is highlighted in yellow. Below the tree, simulation parameters are listed:


```
id : update
time_sensor_frequency : 100
nb points : 2
nb child tasks : 0
```
- 2D Plot (Top Right):** A line graph titled '2D Plot' showing the evolution of 'heat/0' over 100 cycles. The y-axis ranges from 0 to 250. Three data series are shown: Min Data (blue), Max Data (yellow), and Average Data (cyan). The Max Data shows a sharp peak of approximately 230 at cycle 50.

Cycle	Min Data (heat/0)	Max Data (heat/0)	Average Data (heat/0)
0	~25	~75	~50
20	~25	~65	~50
40	~25	~65	~50
50	~25	~230	~50
60	~25	~80	~50
80	~25	~60	~50
100	~25	~60	~50
- VTK 2D Field (Bottom Left):** A visualization of the 'heat2d' field, showing a circular gradient from blue (low temperature) to red (high temperature). A color scale on the right ranges from 0.00 to 200.0.

Temperature	Color
0.00	Blue
50.0	Cyan
100.0	Green
150.0	Yellow
200.0	Red
- Request Manager (Bottom Right):** A table listing simulation requests.

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

The screenshot displays the Simone - Simulation Monitoring application for EPSN. The main window is divided into several panels:

- Task Graph:** A hierarchical tree on the left shows the task structure. The 'update' task is highlighted in yellow. A central diagram shows a loop with 'update' and 'swap' tasks, with an arrow indicating the flow from 'update' to 'swap' and back to 'update'.
- 2D Plot:** A line graph titled '2D Plot' showing 'heat/0' data. The y-axis ranges from 0 to 250, and the x-axis from 0 to 100. A yellow line shows a sharp peak at approximately x=50, reaching a value of about 230. A legend at the bottom indicates 'Min Data : heat/0', 'Max Data : heat/0', and 'Average Data : heat/0'.
- VTK 2D Field:** A 2D heatmap showing a circular distribution of values. A color scale on the right ranges from 0.00 (blue) to 200.0 (red). The center of the field is bright red, indicating high values.
- Request Manager:** A table listing requests with columns for request ID, connection, kind, status, and cycles.

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

- 1 Modèle
 - Modèle de description
 - Modèle de pilotage
- 2 La plate-forme EPSN
 - Architecture
 - Fonctionnalités
 - **Instrumentation d'une simulation**
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

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>
```

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

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

Insertion des points d'instrumentation

Code instrumenté

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

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();
...
```


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'accessibilité

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'accessibilité

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>
    </loop>
  </htg>
</simulation>
```

Configuration de la simulation

XML

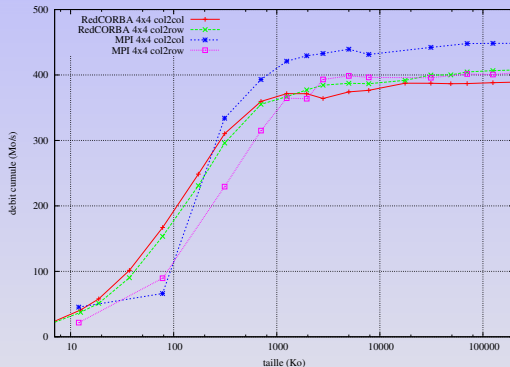
```
<simulation id="particles">
  <data>
    <points id="particles" dimension="3" />
  </data>
  <htg auto-start="false" >
    <loop id="main" synchronize="no" >
      <task id="body" synchronize="no" />
      <task id="acceleration" synchronize="no"/>
      <task id="move" synchronize="no">
       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

- 1 Modèle
 - Modèle de description
 - Modèle de pilotage
- 2 La plate-forme EPSN
 - Architecture
 - Fonctionnalités
 - Instrumentation d'une simulation
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

Performances RedGRID/RedCORBA

Transfert $M \times N$

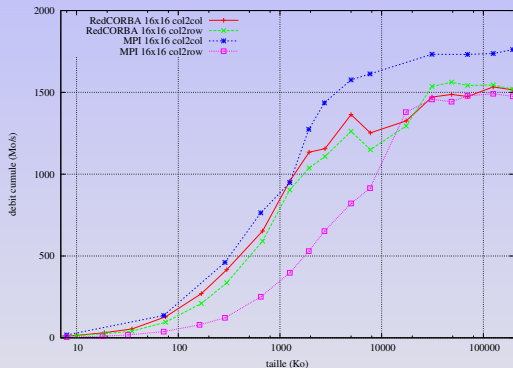
- Field 4x4
- Field 16x16
- Particles 8x8
- Points 8x8
- Points 9x8



Performances RedGRID/RedCORBA

Transfert $M \times N$

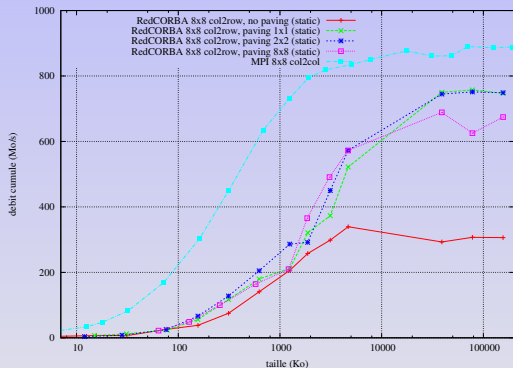
- Field 4x4
- Field 16x16
- Particles 8x8
- Points 8x8
- Points 9x8



Performances RedGRID/RedCORBA

Transfert $M \times N$

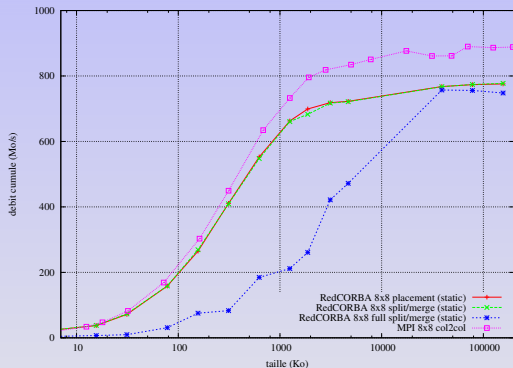
- Field 4x4
- Field 16x16
- **Particles 8x8**
- Points 8x8
- Points 9x8



Performances RedGRID/RedCORBA

Transfert $M \times N$

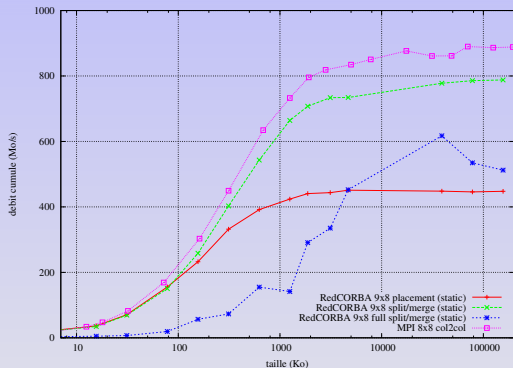
- Field 4x4
- Field 16x16
- Particles 8x8
- **Points 8x8**
- Points 9x8



Performances RedGRID/RedCORBA

Transfert $M \times N$

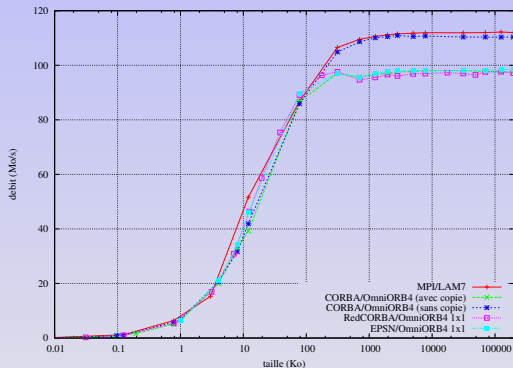
- Field 4x4
- Field 16x16
- Particles 8x8
- Points 8x8
- **Points 9x8**



Performances EPSN - Recouvrement

1 vers 1

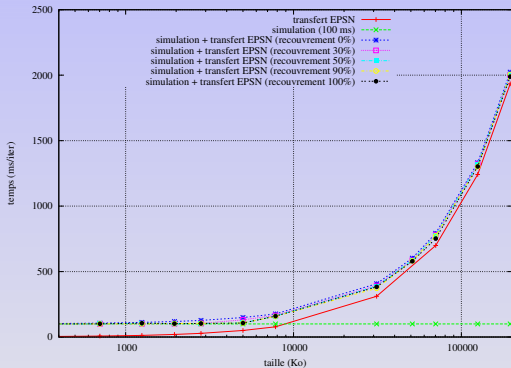
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Recouvrement

1 vers 1

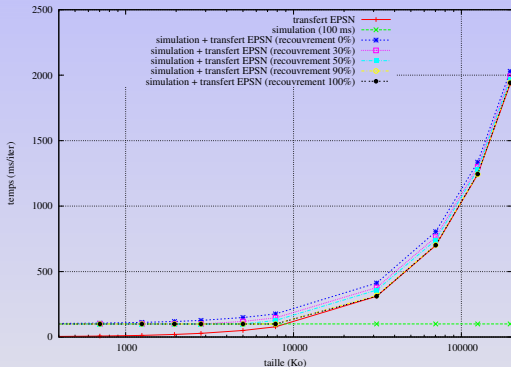
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Recouvrement

1 vers 1

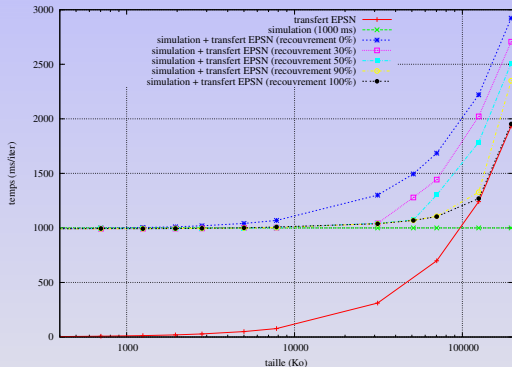
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Recouvrement

1 vers 1

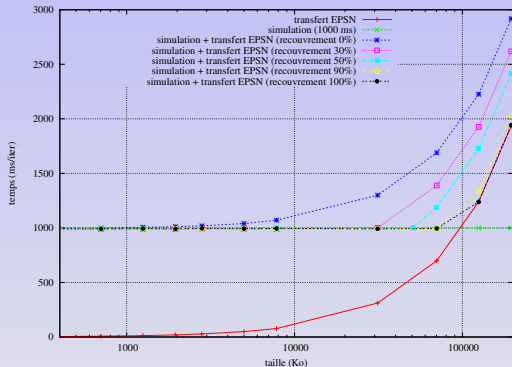
- transfert
- 100 ms sur mono-pro
- 100 ms sur bi-pro
- 1 s sur mono-pro
- 1 s sur bi-pro



Performances EPSN - Recouvrement

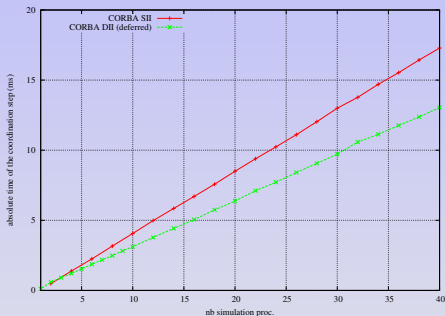
1 vers 1

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



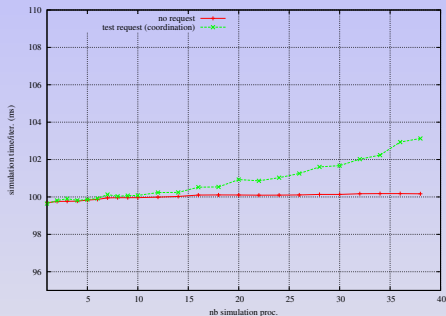
transfert des requêtes vers les noeuds

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



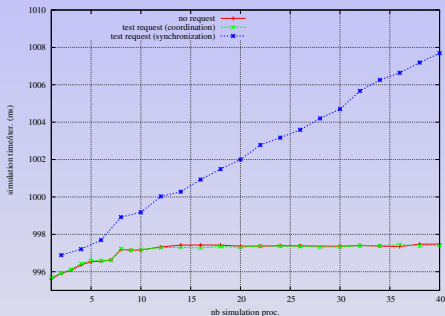
transfert des requêtes vers les noeuds

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



transfert des requêtes vers les noeuds

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



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 à voir 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.*

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.*

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 ... voir 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.*

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.*

- 1 Modèle
 - Modèle de description
 - Modèle de pilotage
- 2 La plate-forme EPSN
 - Architecture
 - Fonctionnalités
 - Instrumentation d'une simulation
 - Resultats et perspectives
- 3 EPSN et visualisation parallèle

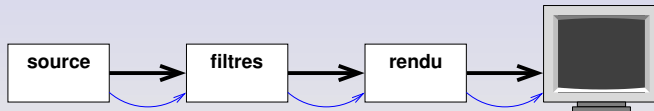
Les systèmes *Event-driven*

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



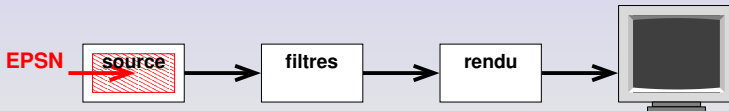
Les systèmes *Event-driven*

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



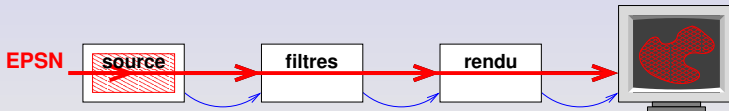
Les systèmes *Event-driven*

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



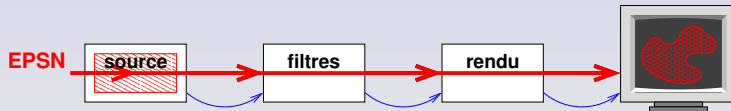
Les systèmes *Event-driven*

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



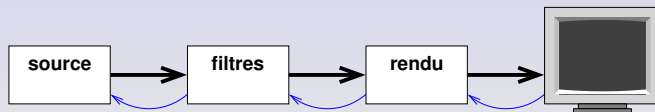
Les systèmes *Demand-driven*

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



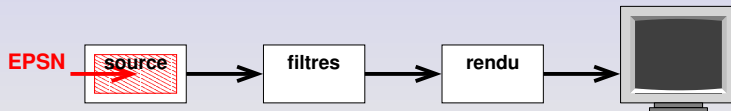
Les systèmes *Demand-driven*

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



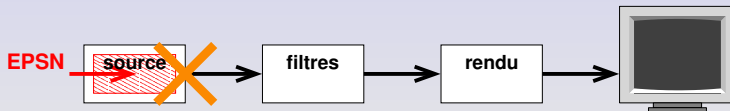
Les systèmes *Demand-driven*

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



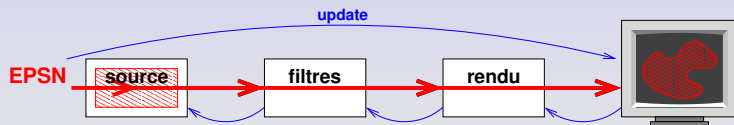
Les systèmes *Demand-driven*

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



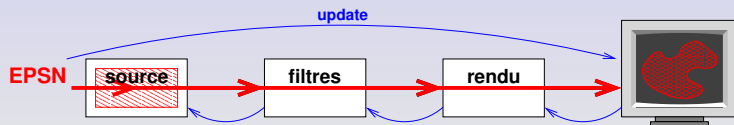
Les systèmes *Demand-driven*

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 aux 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 \rightarrow `vtkImageData`
 - points \rightarrow `vtkPoints`
 - maillages \rightarrow `vtkUnstructuredGrid`
 - séries \rightarrow `vtkDataArray`

En cours

\Rightarrow Ecritures de sources RedGRID pour VTK

Source de données pour VTK

RedSYM vs. VTK

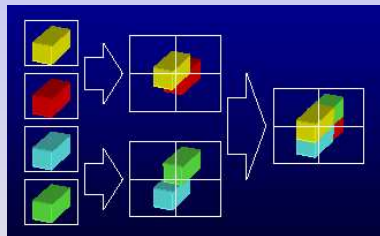
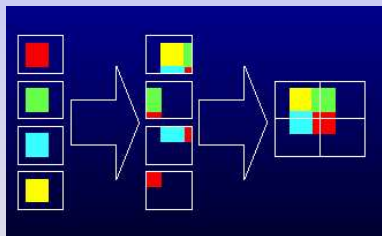
- Même système de données
 - objets complexes avec des séries de données associées
 - séries \leftrightarrow listes de *tuples* à plusieurs composantes homogènes
- Partage de la mémoire ou recopie
- Equivalence :
 - champs \rightarrow `vtkImageData`
 - points \rightarrow `vtkPoints`
 - maillages \rightarrow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

En cours

\Rightarrow Ecritures de sources RedGRID pour VTK

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

