Dataflows scientifiques : introduction, langage et architecture d'exécution

Centre de Données de Strasbourg

26 mars 2004

Jean-Pierre.Matsumoto@inria.fr

Projet Smis - Inria Rocquencourt

Contexte

Chaînes de traitements scientifiques pour l'analyse de données

- Chaîne de traitement ?
  - Unité de traitement = boîte noire définissant
    - Ses données arguments (les entrées)
    - Ses données résultats (les sorties)
  - Mises bout à bout => chaîne de traitements

Contexte (suite)

- Cycle de vie détaillé d'une chaîne de traitement
  1. Conception
     - Spécification
     - Développement
  2. Exécution
     - Instanciation
     - Accès aux données d'entrées et aux programmes exécutables
     - Stockage des résultats
  3. Exploitation des résultats
     - Visualisation / Interprétation des résultats

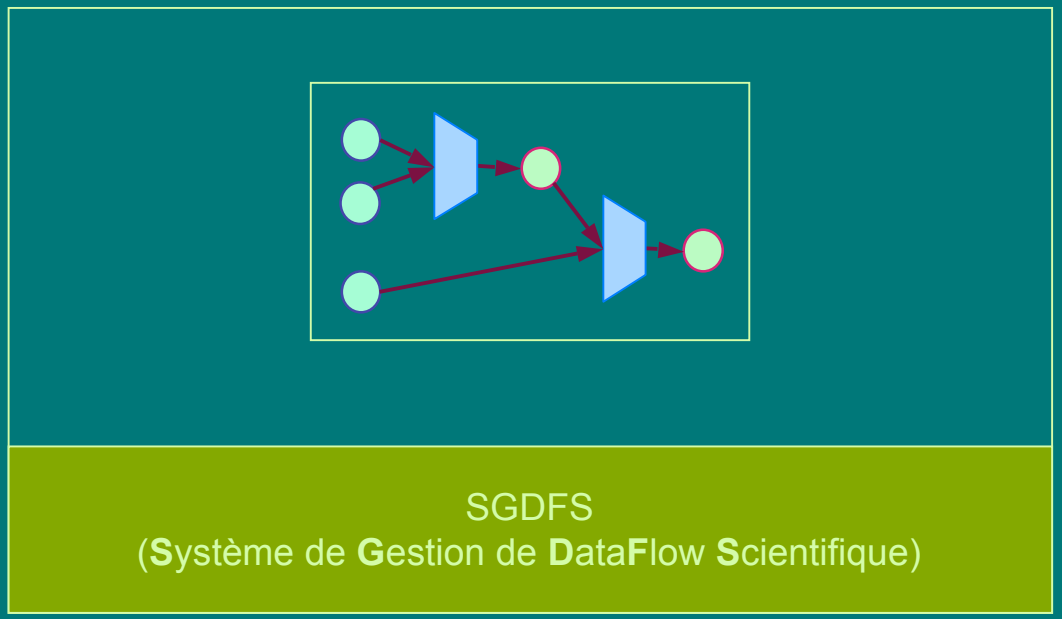
Motivation

- Avantages des chaînes de traitements vs. programmes monolithiques
  - Pendant la phase de conception
    - Spécification formelle plus fine
    - Gains de temps de développement et de maintenance
      - Réutilisation d'unité de traitement
      - Amélioration indépendante des unités
      - Partage d'unités
  - Pendant la phase d'exécution
    - Répartition des calculs grâce à la supervision par un système
    - Stockage automatique et organisé des résultats
    - Données intermédiaires consultables
    - Suivi de l'avancement

Motivation (suite)

- (suite) Avantages des chaînes de traitements vs. programmes monolithiques
  - Pendant la phase d'exploitation des résultats
    - Les résultats sont explicables.
      - Leur hérédité est conservée en leurs attachant la définition de la chaîne de traitement qui les a produit.
    - La définition d'une chaîne de traitement peut être utilisée comme vue conceptuelle des données.

Notre approche

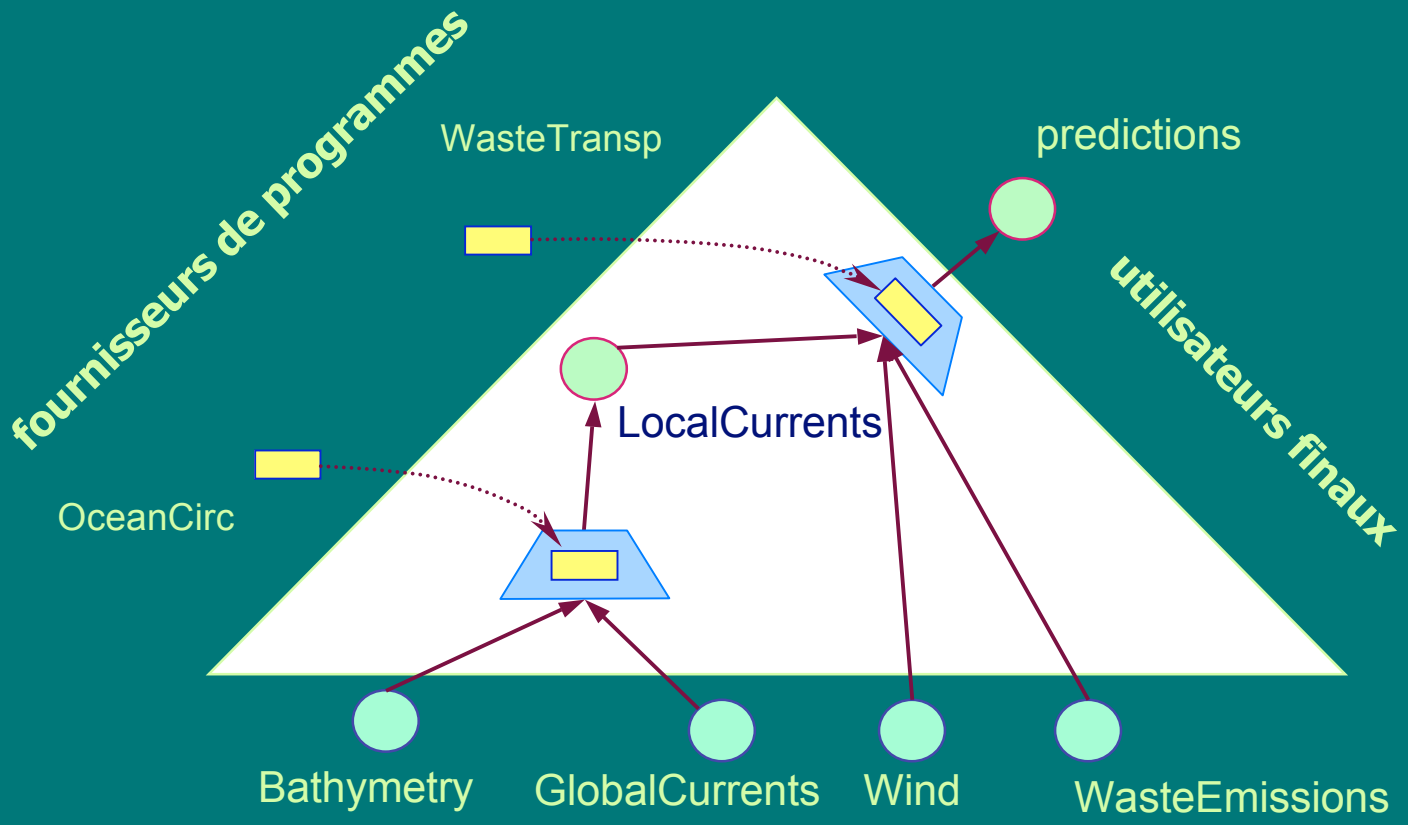


Mise au point du langage de DataFlow Scientifique (DFS)

Conception d'un système

Avec données et programmes distribués sur un réseau !

Exemple concret



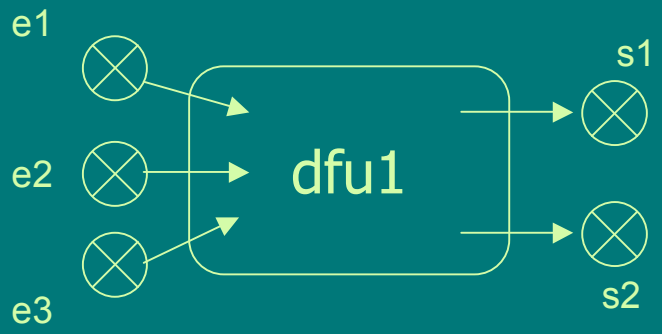
Le langage de Dataflow Scientifique

Base (1/3) : DFU et entité de donnée

DFU = sigle pour **D**ata**F**low **U**nit

1 dataflow unit = 1 traitement

1 DFU est caractérisée par les entités de données qu'elle utilise en entrée (ses arguments) et par les entités de données qu'elle produit (ses résultats)



Syntaxe

```
dfu1(in={e1,e2,e3},out={s1,s2})
```

Sémantique

Dès que e1,e2,e3 sont disponibles, dfu1 s'exécute et produit 1 valeur de s1 et 1 valeur de s2. s1 et s2 sont alors disponibles.

Notation

```
s1 = dfu1|s1(e1,e2,e3)
s2 = dfu1|s2(e1,e2,e3)
```

Base (2/3) : types d'entités de données

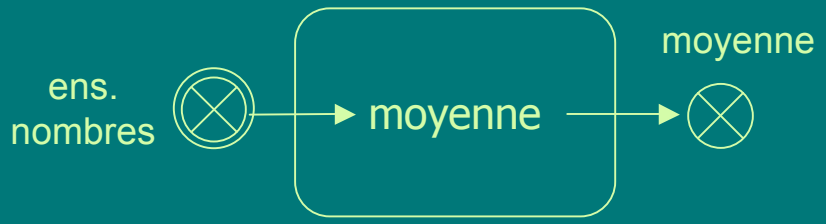
3 types de base :

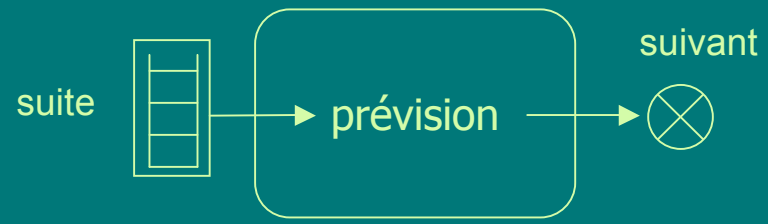
séquence

ensemble

atomique

Exemples :





Base (3/3) : datalinks

1 *datalink* = un lien orienté entre deux entités de données

L'origine du datalink est appelé *entité de donnée source*

La cible du datalink est appelée *entité de donnée destination*

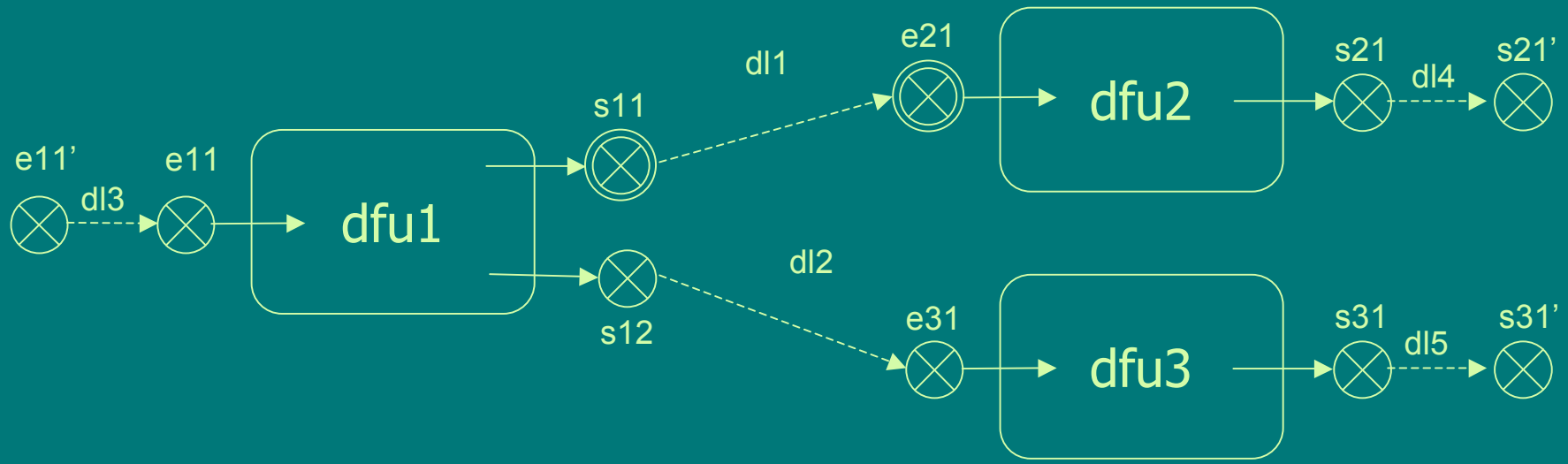
Syntaxe

```
dl(src=s1,dest=e2)
```

Sémantique

Dès que s1 est disponible, le datalink dl fait effet et s2 prend la valeur de s1. s2 est alors disponible.

Premier exemple

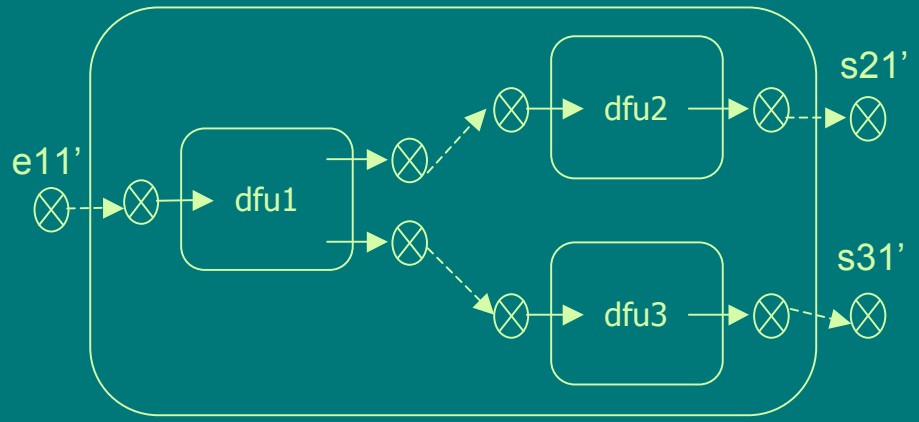



```
s21' = s21                              car dl4(src=s21,dest=s21')
     = dfu2|s21(e21)                    car dfu2(in={e21},out={s21})
     = duf2|s21(s11)                    car dl1(src=s11,dest=e21)
     = dfu2|s21(dfu1|s11(e11))          car dfu1(in={e11},out={s11,s12})
     = dfu2|s21(dfu1|s11(e11'))         car dl3(src=e11',dest=e11)
```

Par un raisonnement analogue : `s31' = dfu3|s31(dfu1|s12(e11'))`

Encapsulation

Une *encapsulation* sert à abstraire un section d'un dataflow.



Syntaxe

```
dfu4(dfu_encap={dfu1,dfu2,dfu3},
     dl_encap={dl1,dl2},
     in={e11},
     out={s21,s31})
```

Support du calcul itératif, pourquoi ?

- Les chaînes de traitements scientifiques utilisent souvent des processus itératifs s'appliquant sur des séquences
  - temporelles
  - spatiales
- Support du calcul itératif au niveau du langage
  - Sort les itérations des programmes
    - + fine granularité de spécif
    - + de répartition de charge possible

Extension du modèle

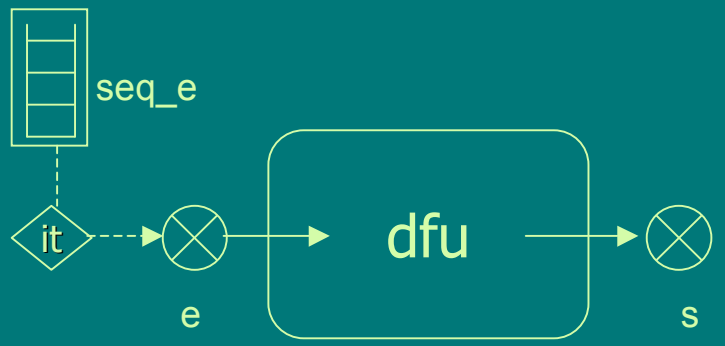
Nouvelle Notation

Si d est de type ensemble ou atomique :

d[i] est la valeur de d au moment de l'itération i

Si d est de type séquence :

d[i] est i-ème valeur de la séquence d



Exemple

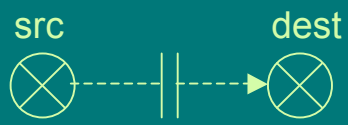
Pour tout i,

e[i]=seq_e[i]

s[i]=dfu|s(seq_e[i])

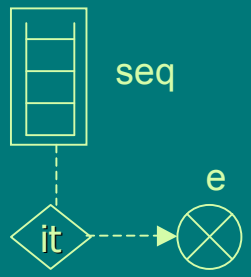
Types de datalink supplémentaires

4 nouveaux types



à effet retardé

```
dest[i]:=src[i-1]
```

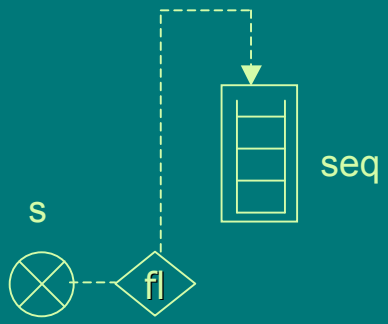



```
e[i]:=seq[i]
```

lien itérateur

à effet cumulatif

```
dest[i]:=Union(
src[i],src[i-1], …, src[1])
```



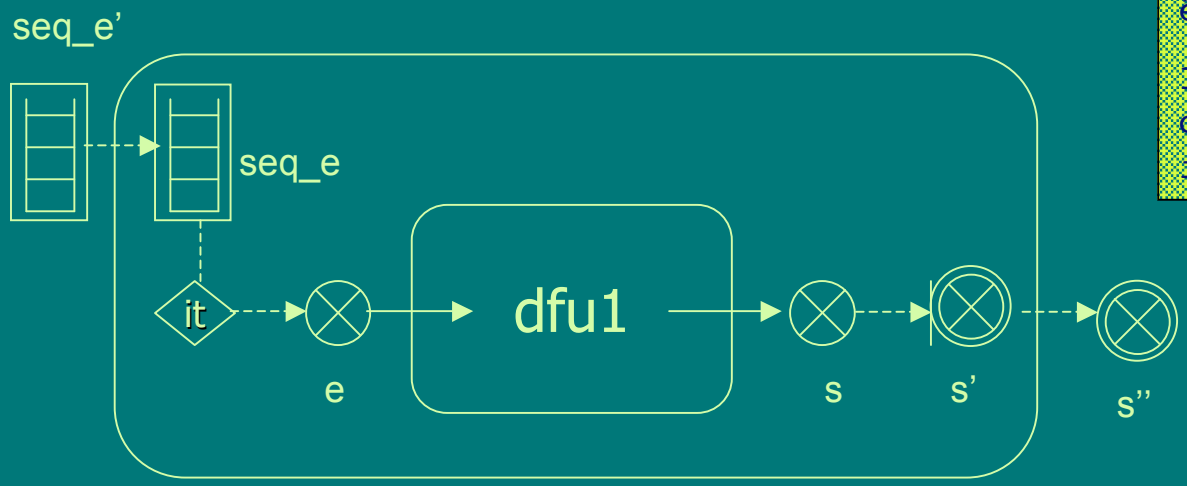

```
seq[i]:=s[i]
```

lien « remplir »

Encapsulation & calcul itératif

1 lien itérateur => 1 encapsulation

- pour définir la porté du calcul itératif

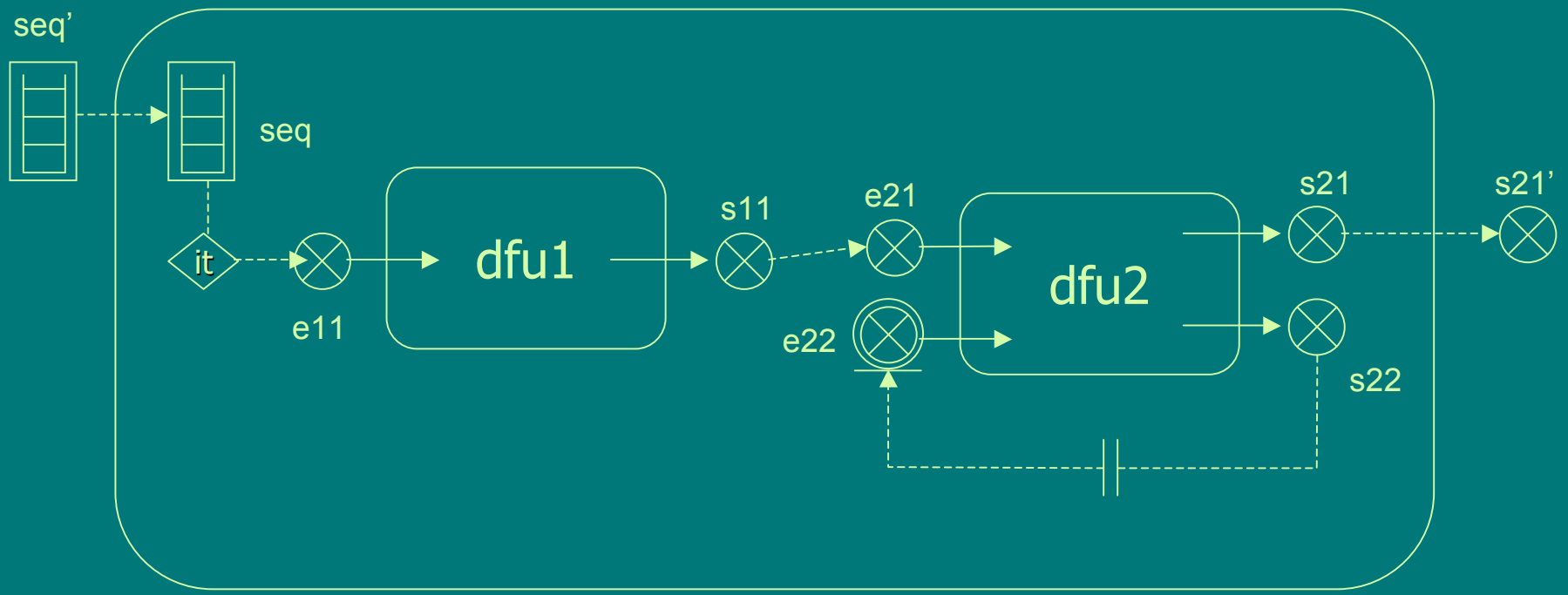


Les datalinks coupés par l'encapsulation doivent être de type simple.

Ils ne font effet qu'au début et à la fin du calcul itératif.

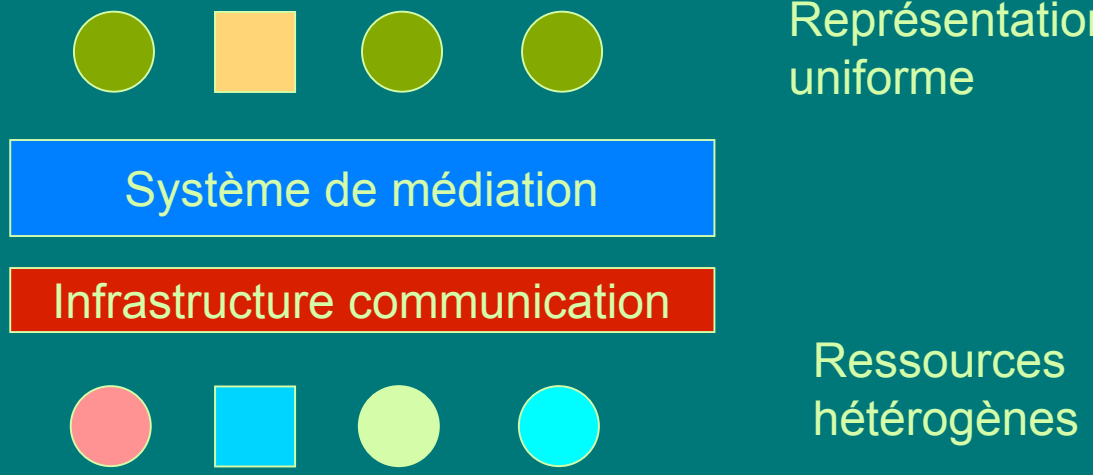
Pour toute la séquence seq_e', une seule valeur de s'' est produite

Deuxième exemple



Architecture Système de Gestion de DataFlow Scientifique + Système de Médiation

Qu'est-ce qu'un système de médiation ?

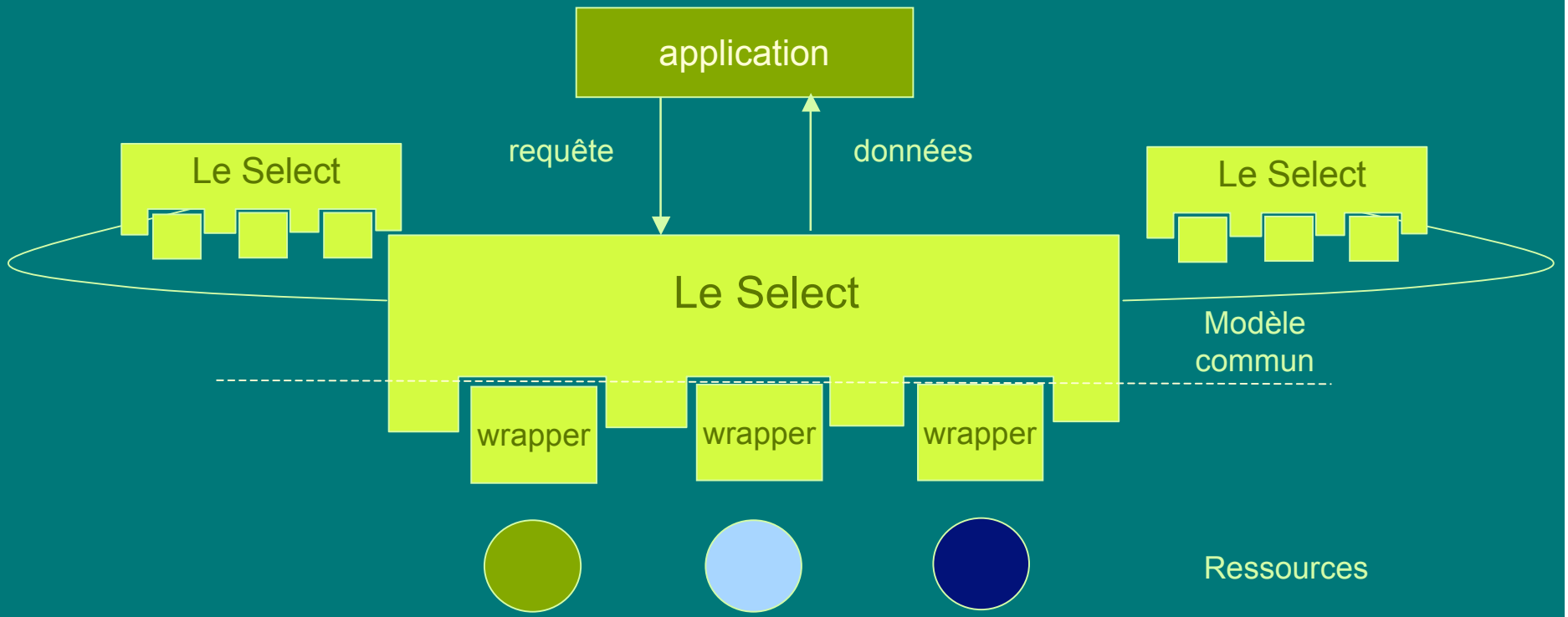


- Les données au niveau uniformes sont virtuelles, non matérialisées (!= dépôt de données)
- Le système de médiation abstrait également la répartition sur le réseau des ressources

Avantages des médiateurs dans le contexte scientifique

- Relatifs aux données
  - Données distribuées interopérables
  - Langage de haut-niveau pour l'interrogation
  - Fraîcheur des données
  - Optimisation de l'accès aux données
- Relatifs au programmes
  - Uniformisation de l'interface des programmes
  - Appel à distance de programme préconfiguré
- Relatifs à l'administration
  - Autonomie des sources de données
  - Faible coût d'administration (contrôle distribué)

Le Select – Un système de médiation



Le modèle de données commun

- Le modèle de donnée de LeSelect est relationnel
  - Les données sont organisées en tables contenant des lignes et des colonnes
- Types de données qui peuvent être mis dans une colonne
  - Types basiques : integer, string, dates, etc.
  - Types non structurés, larges : images, cartes, ...
    - Ces types sont qualifiés par une étiquette (le type mime)
    - Exemple: une image au format GIF a le type mime image/gif

Publication de données – Exemple

blob = Binary Large Object

Table Bathym

Id	COORDRECT	MAP
bathy01	(155,4940);(158,4941)	blob
bathy02	(156,4941);(156,4943)	blob
…		

Text Wrapper

Fichier bathym.txt

```
bathy01,155,4940,158,4941,/maps/bathy/01.map
bathy02,158,4941,159,4943,/maps/bathy/02.map
…
```

Publication de données

- Les données sont traduites en tables
  - par un adaptateur de données écrit par le « publisher »
  - des adaptateurs génériques existants et configurables peuvent être utilisés

- Que doit faire l'adaptateur de données ?
  - donner l'accès aux schémas des données
  - accéder aux données dans leur format natif de stockage
  - traduire les données au format relationnel
  - donner l'accès aux données traduites par une interface commune

Publication de programme - exemple

LeSelect program OceanCirc (input A: table[map_bathym];
input B: table[map_global_currents];
parameter grid_Res: double precision):
output: local_currents[Id, map_local_currents]

Program Wrapper

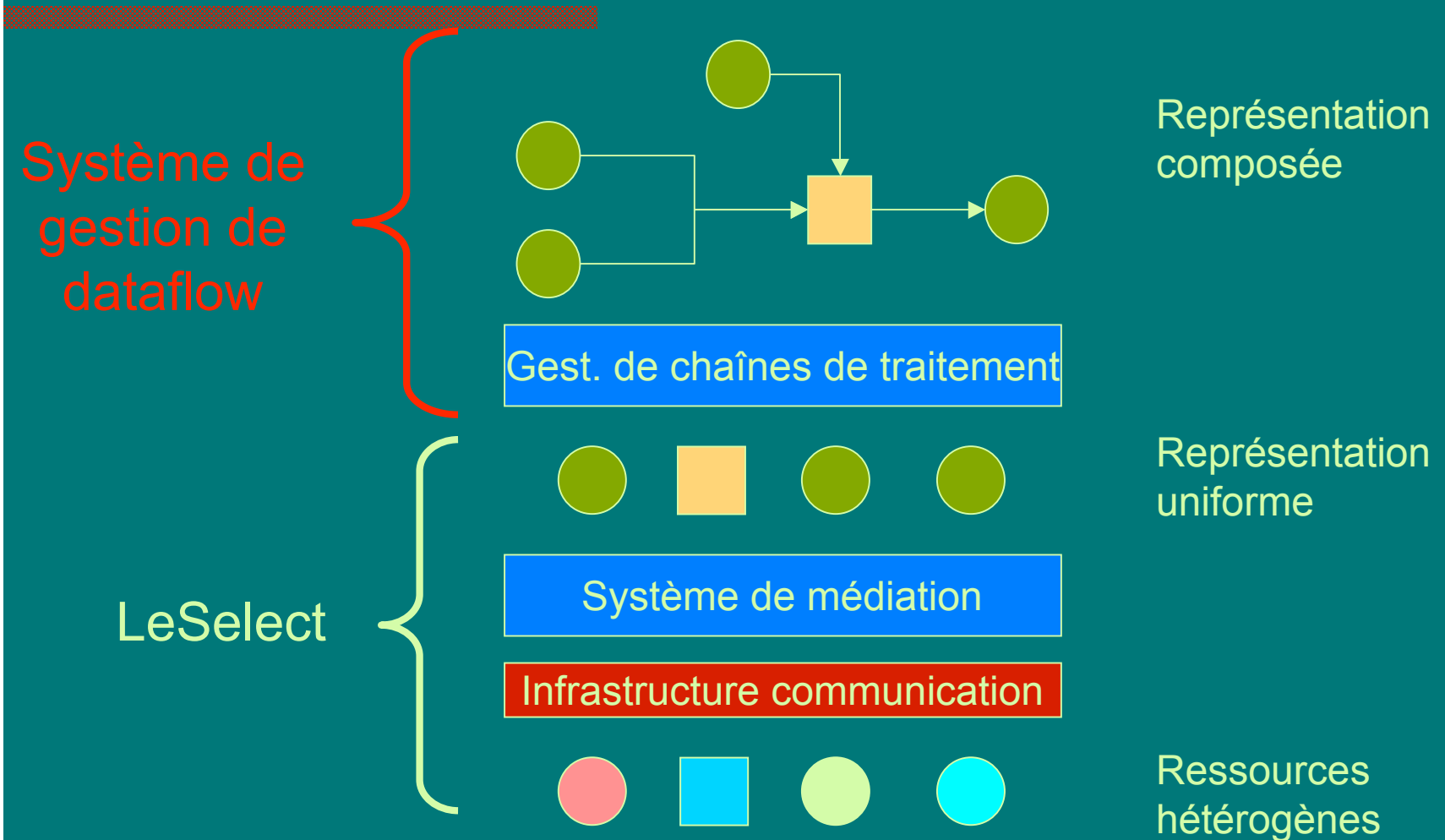
Program OceanCirc ({map_bathym}, {map_global_currents},
grid_Res: float):
{map_local_currents}

Modèle mathématique qui calcule les courants locaux à partir de courants globaux et de données de bathymétrie locale

Publication de programme

- Les programmes sont publiés via un adaptateur de programme qui :
  - Invoque les programmes de manière asynchrone
  - Obtient les données d'entrée à partir de serveurs LeSelect éventuellement distants
  - Donne l'accès aux résultats par un adaptateur de données
- Les adaptateurs de programme s'occupent aussi :
  - Spécifier le schéma des données d'entrées/sorties en tables
  - Informe LeSelect du coût d'exécution du programme
- Les adaptateurs de prog. sont écrits par les « publishers » :
  - Mais des adaptateurs génériques existent et peuvent être réutilisés

Complémentarité Médiation / Dataflow



Dataflow scientifique

Publication de programmes scientifiques pour un dataflow

- Au niveau hétérogène
  - 1 prog. en ligne de commande
  - manipulant des fichiers

+ adaptateur LeSelect =

- Au niveau uniforme
  - 1 prog. LeSelect
  - manipulant des tables

+ fichier de configuration Dataflow =

- Au niveau dataflow
  - 1 DFU
  - manipulant des entités de données

Publication de données scientifiques pour un dataflow

- Au niveau hétérogène
  - Des données textuelles ou des données structurées ou des fichiers binaires etc.
  - Difficilement interopérables

+ adaptateur LeSelect =

- Au niveau uniforme
  - Tables relationnelles
  - Interrogeables en SQL

+ fichier de configuration Dataflow =

- Au niveau dataflow
  - Une entité de données

Perspectives

- Implémentation d'un dataflow avec l'Institut de Recherche pour le Développement (délimitation de territoires de villages en Afrique)
- Conception / Implémentation d'un dataflow avec la Maison de la Télédétection (gestion des digues françaises)
- Conception / Implémentation d'un portail web pour l'ACI Neurobase (partage de traitements et de données en Neuroimagerie médicale)
- Migration des données et chaînes de traitements du projet Clime de l'Inria vers LeSelect + Dataflow
- Distribution du système de gestion de dataflow
  http://www-smis.inria.fr/~jpmat/dataflow/

FIN