Conception d'un modèle et de *frameworks* de distribution d'applications sur grappes de PCs avec tolérance aux pannes à faible coût

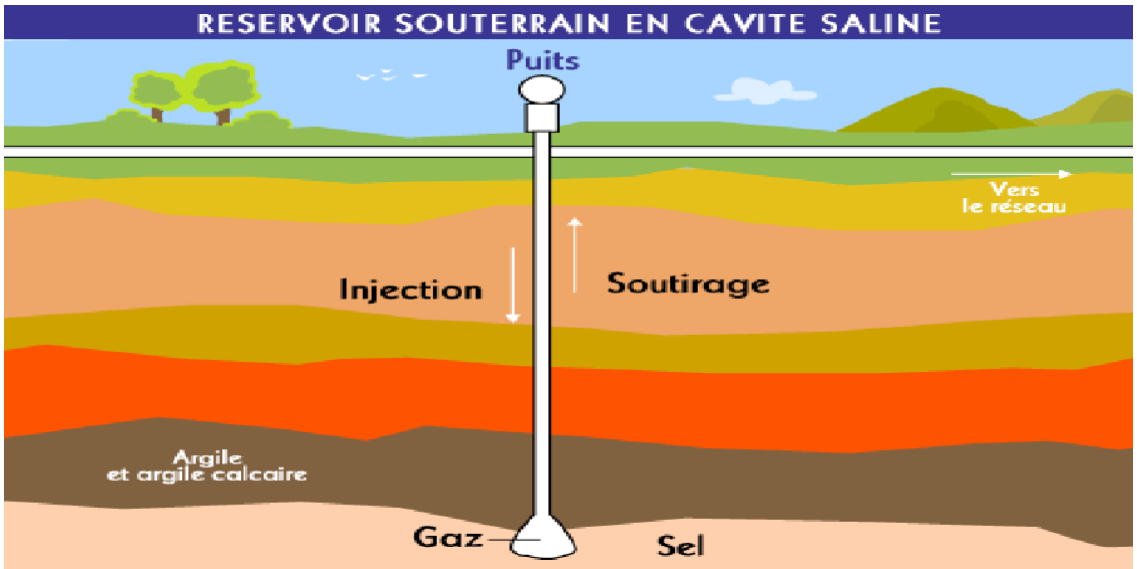
Constantinos Makassikis

Soutenance de thèse
2 février 2011

Applications exigeantes

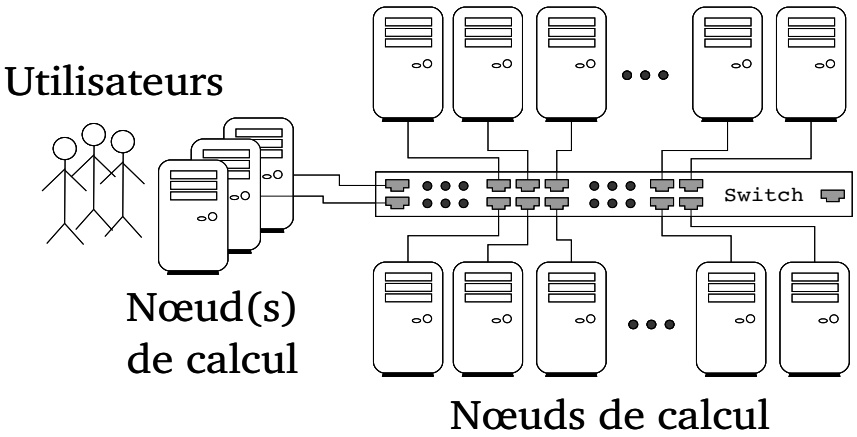
- Besoins accrus en puissance de calcul

  $\longrightarrow$ pour réaliser des simulations plus grandes
- Besoin de respect d'échéances
- Domaines d'application variés :



Optimisation d'actif de stockage de gaz
Application d'EDF R&D et de Supélec

Grappes de PCs



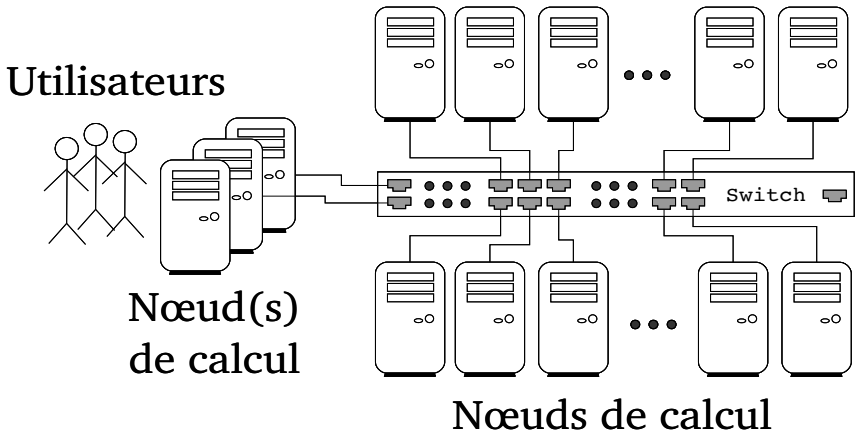
Architecture distribuée

- Répond aux besoins en ressources
  - Nœuds et réseau rapides
  - Composants standard
  - Faible coût
  - Extensibilité

Conséquences

- Fiabilité réduite
  - Pannes de nœuds plus fréquentes
    - $\longrightarrow$ Interruption de l'exécution des applications
    - $\longrightarrow$ Terminaison d'applications longues compromise
    - $\longrightarrow$ Manquement d'échéances
    - $\longrightarrow$ Gaspillage de temps de calcul, d'énergie, et d'argent

Grappes de PCs



Architecture distribuée

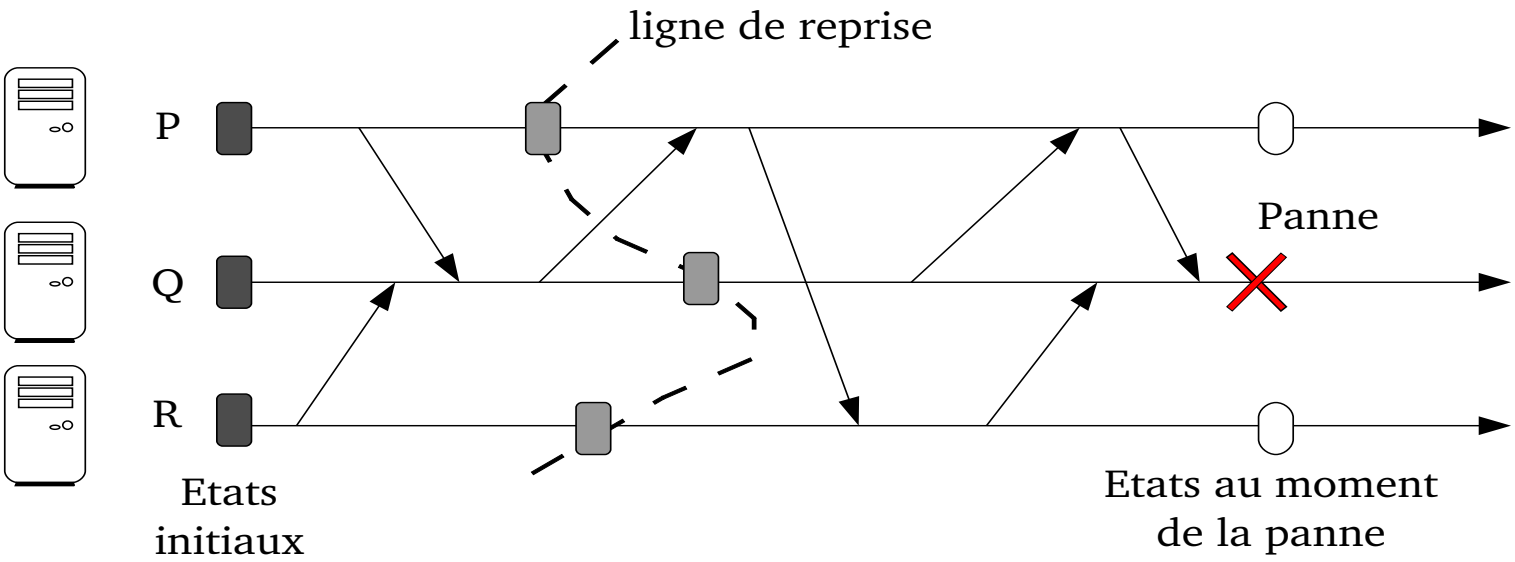
- Répond aux besoins en ressources
  - Nœuds et réseau rapides
  - Composants standard
  - Faible coût
  - Extensibilité

Conséquences

- Fiabilité réduite
  - Pannes de nœuds plus fréquentes
    - $\longrightarrow$ Interruption de l'exécution des applications
    - $\longrightarrow$ Terminaison d'applications longues compromise
    - $\longrightarrow$ Manquement d'échéances
    - $\longrightarrow$ Gaspillage de temps de calcul, d'énergie, et d'argent

Besoin de tolérance aux pannes

Techniques de reprise par retour arrière



Principe

- Sauver et restaurer des **états intermédiaires cohérents**

  $\longrightarrow$ Éviter une reprise depuis le tout début

- Introduit des surcoûts : exécution, reprise, occupation disque

  $\longrightarrow$ Risque toujours présent de manquer une échéance
  $\longrightarrow$ Besoin de minimiser les surcoûts

Dualité des niveaux de mise en œuvre

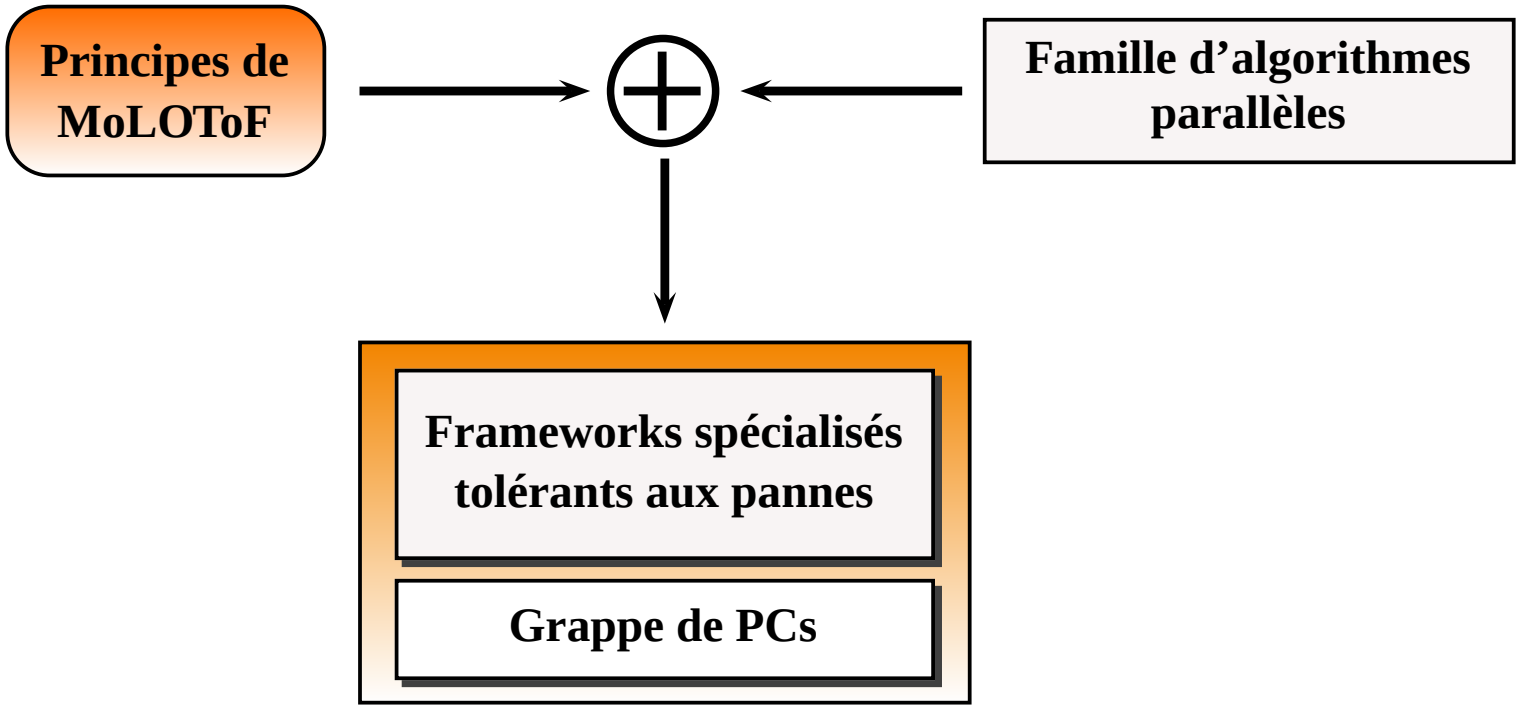
Niveau système

- *Dump* d'octets des processus de la mémoire sur disque
  - Grande transparence – Faible portabilité

Niveau applicatif

- Transformations complexes du code source
  - Faible transparence – Grande portabilité
  - Potentiellement grande efficacité :
    - Exploitation de la sémantique applicative
- Différentes approches :
  - Manuelle
  - Assistée par compilateur
  - Assistée par framework

Approche MoLOToF



- Approche de niveau framework favorisant interactions avec le programmeur et l'environnement
- Points de reprises/sauvegardes de niveau applicatif

Plan

Plan

Définition et objectifs

MoLOToF

- **Mo**del for **L**ow-**O**verhead **T**olerance **o**f **F**aults

Qu'est-ce que MoLOToF ?

- **Ensemble de règles** autour du concept de **squelettes tolérants aux pannes**

Quels sont les objectifs de MoLOToF ?

- Développement d'applications distribuées tolérantes aux pannes **facilité**
- Tolérance aux pannes **efficace** et **portable**

  $\longrightarrow$ Contrer les pannes temporaires/permanentes par arrêt

  $\longrightarrow$ Respecter les échéances au mieux

Squelettes tolérants aux pannes (FT)

- **Concentrer** la tolérance aux pannes aux **parties importantes** de l'application
  - parties **intenses en calculs** → **opérations lourdes**
  - autres opérations → **opérations légères**
- Deux types : séquentiel et parallèle

Exemple de squelettes simples avec des boucles de calcul intensif

```
FT_Seq_Skel
{
  FT_Loop
  {
    calculs()

    checkpoint()
  }
}
```

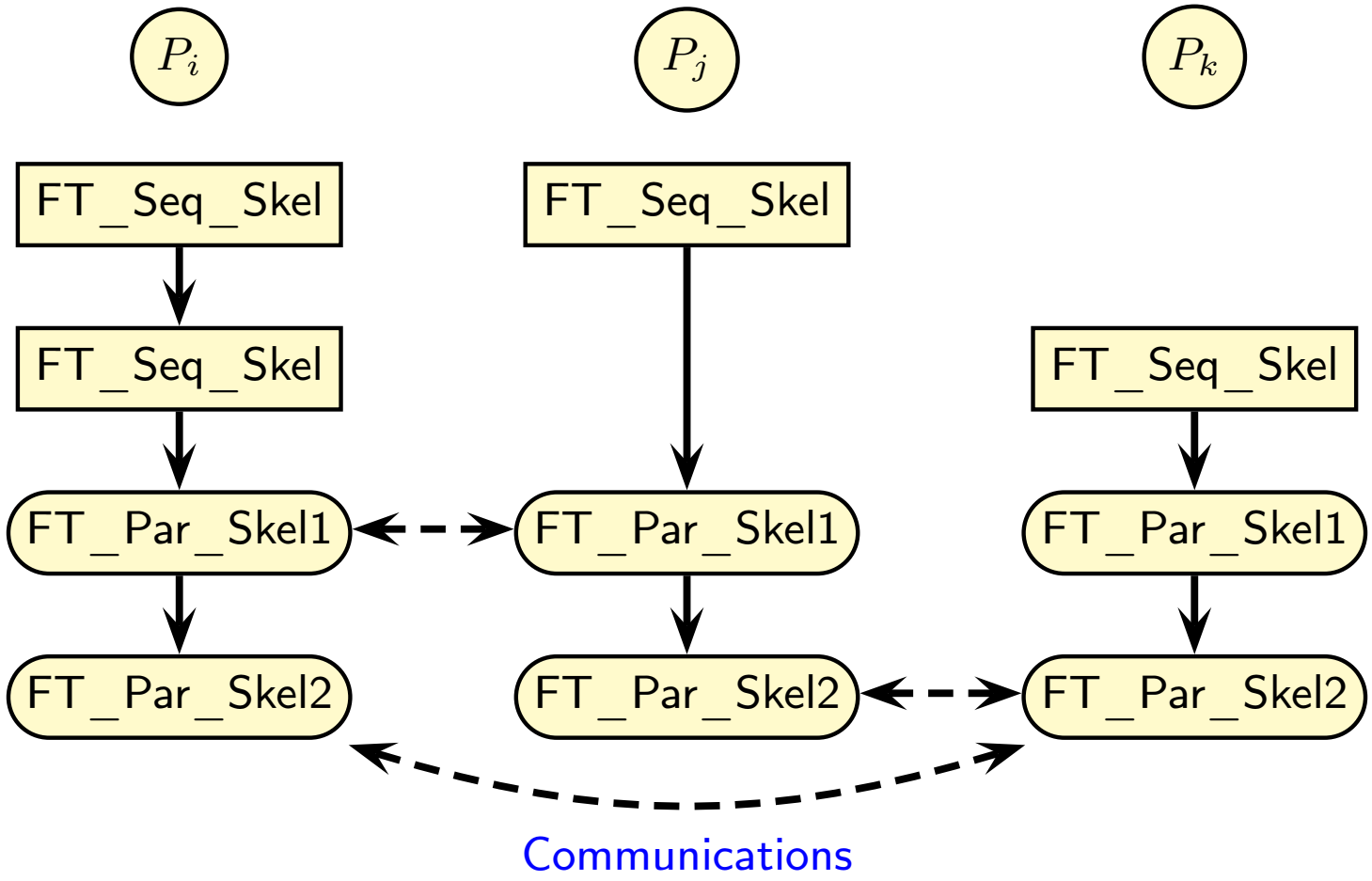
Squelette Séquentiel

```
FT_Par_Skel
{
  FT_Loop
  {
    calculs()
    communications()
    checkpoint()
  }
}
```

Squelette Parallèle

Organisation de l'application

- Une application distribuée comporte plusieurs processus
- Dans MoLOToF, chaque processus est une **succession de squelettes FT**

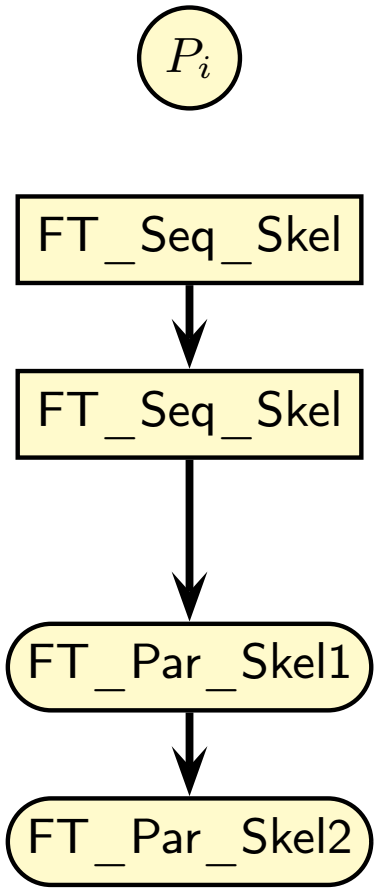


Plan

Sauvegarde

Mode d'exécution normal

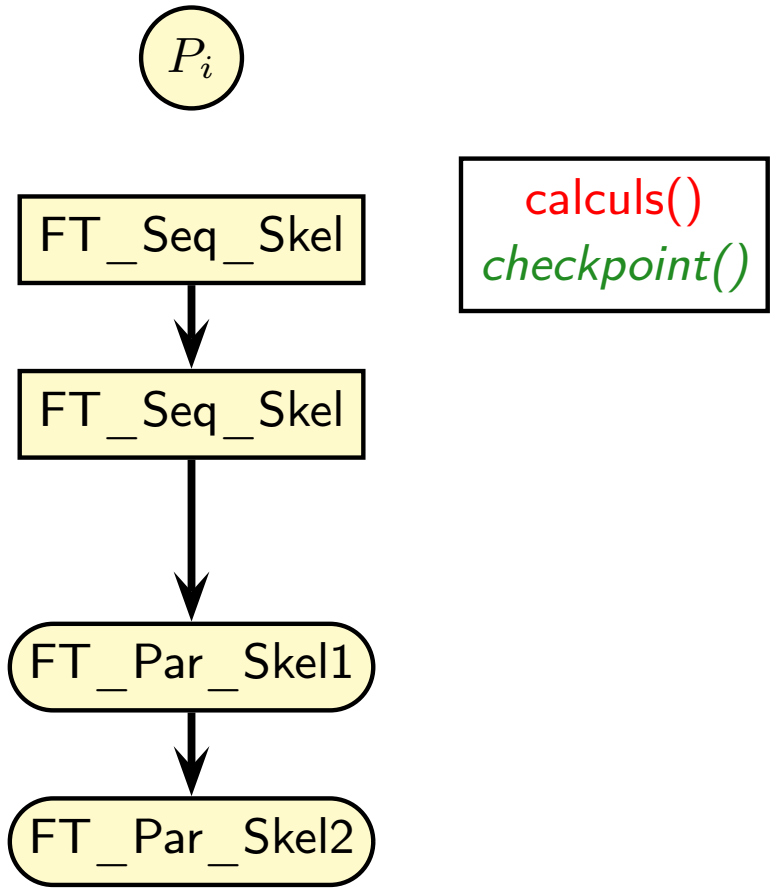
- Exécution du code de l'application de tolérance aux pannes
- Sauvegarde d'un processus
  1. aux points de sauvegarde et si
  2. la condition de sauvegarde est vérifiée



Sauvegarde

Mode d'exécution normal

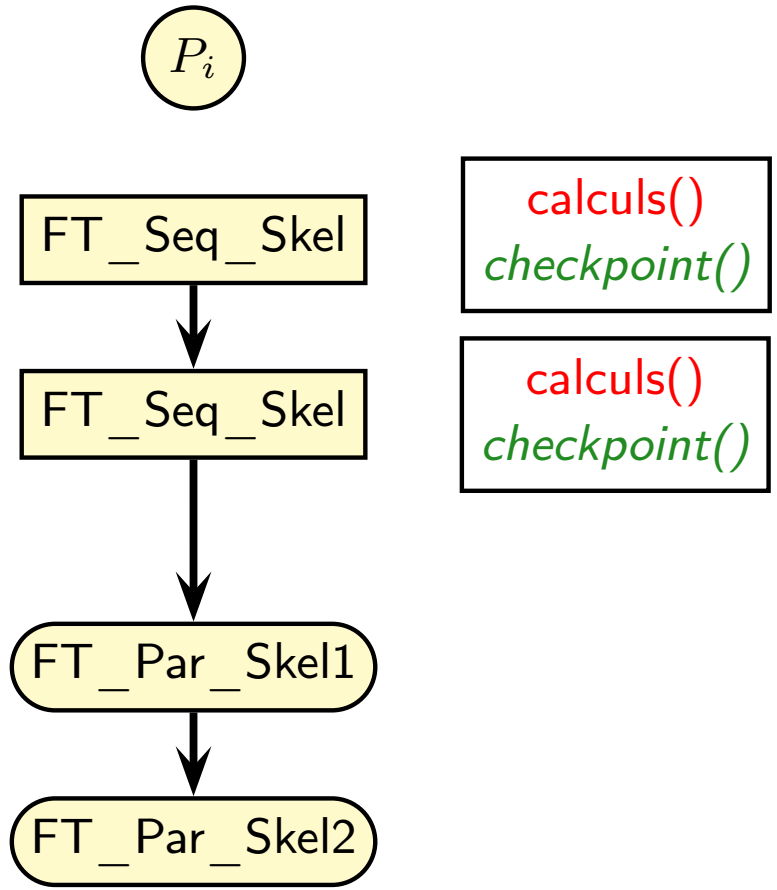
- Exécution du code de l'application de tolérance aux pannes
- Sauvegarde d'un processus
  1. aux points de sauvegarde et si
  2. la condition de sauvegarde est vérifiée



Sauvegarde

Mode d'exécution normal

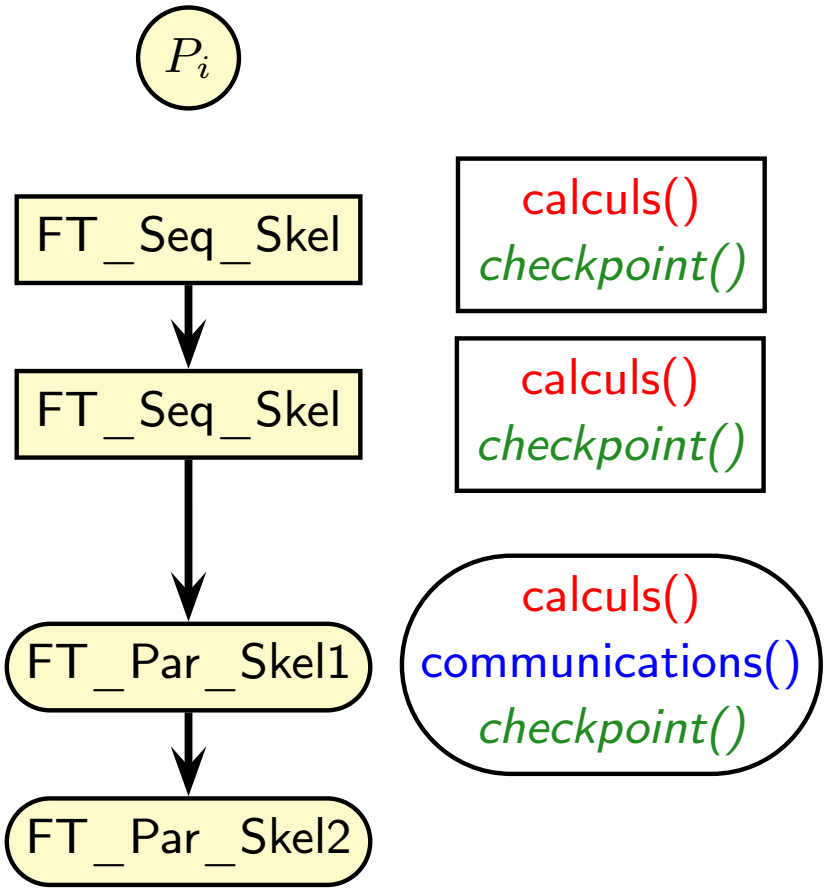
- Exécution du code de l'application de tolérance aux pannes
- Sauvegarde d'un processus
  1. aux points de sauvegarde et si
  2. la condition de sauvegarde est vérifiée



Sauvegarde

Mode d'exécution normal

- Exécution du code de l'application de tolérance aux pannes
- Sauvegarde d'un processus
  1. aux points de sauvegarde et si
  2. la condition de sauvegarde est vérifiée

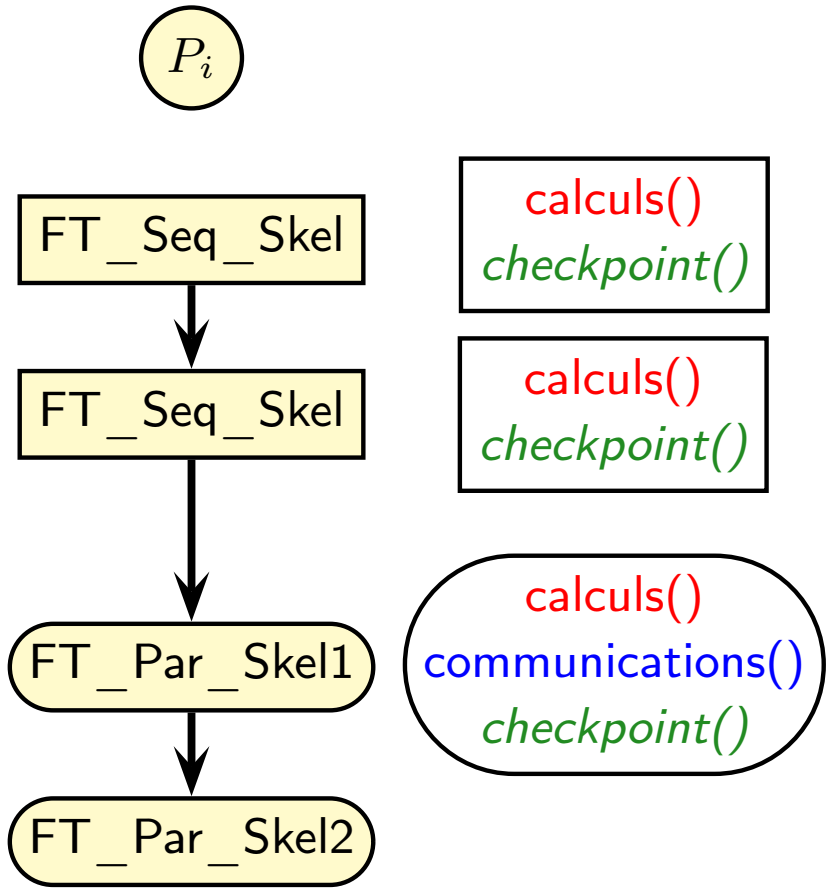


Sauvegarde

Mode d'exécution normal

- Exécution du code de l'application de tolérance aux pannes
- Sauvegarde d'un processus
  1. aux points de sauvegarde et si
  2. la condition de sauvegarde est vérifiée

Sauvegarde de P_i à l'itér. j

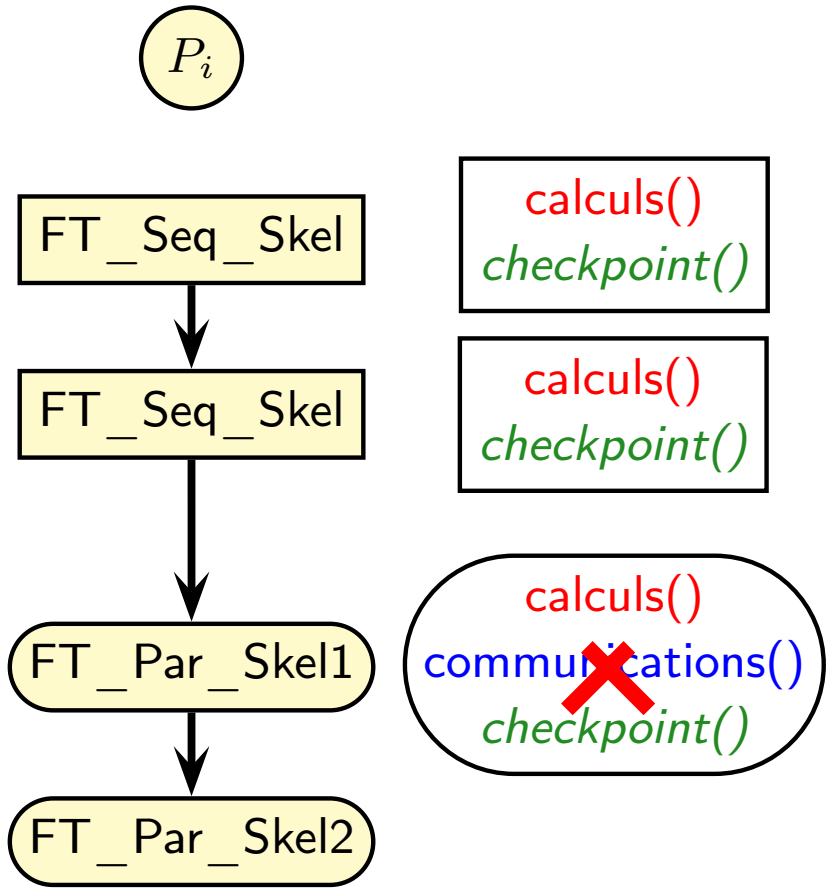


Sauvegarde

Mode d'exécution normal

- Exécution du code de l'application de tolérance aux pannes
- Sauvegarde d'un processus
  1. aux points de sauvegarde et si
  2. la condition de sauvegarde est vérifiée

Sauvegarde de P_i à l'itér. j

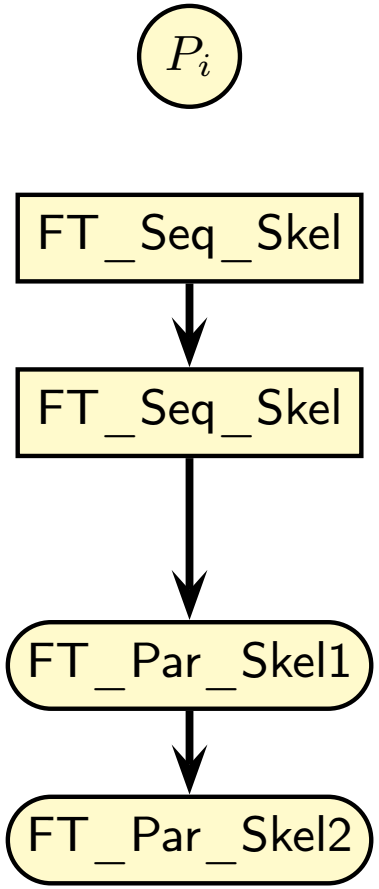


Panne de P_i à l'itér. $j+1$

Reprise

Mode d'exécution de reprise

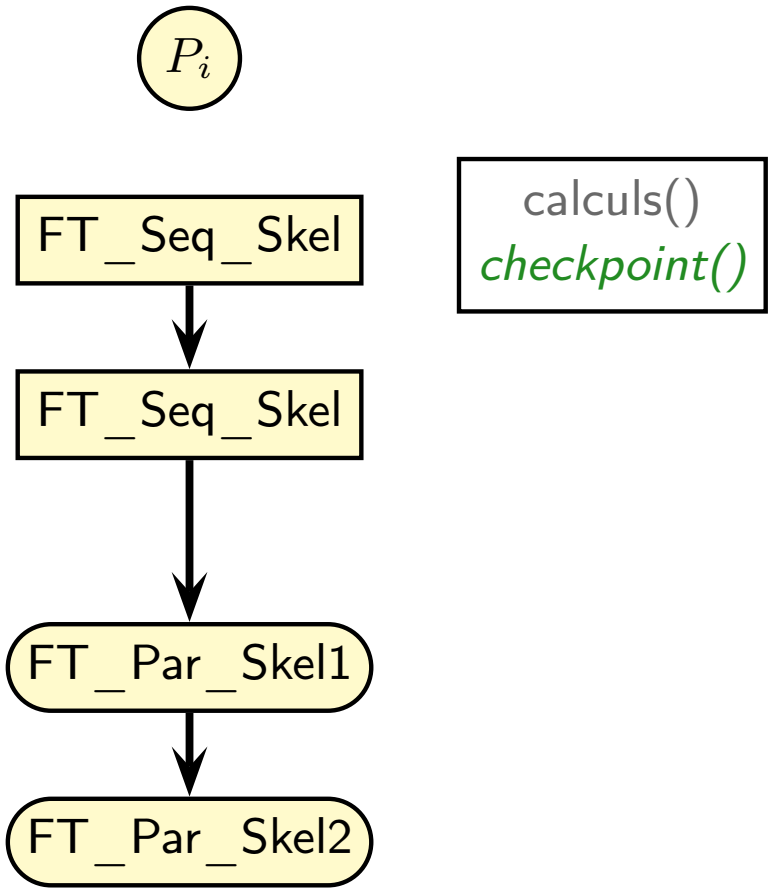
- Détermination de la **ligne de reprise**
- **Réexécution sélective** recouvrement du contexte du processus :
  1. Réexécution d'opérations légères
  2. Omission d'opérations lourdes déjà exécutées



Reprise

Mode d'exécution de reprise

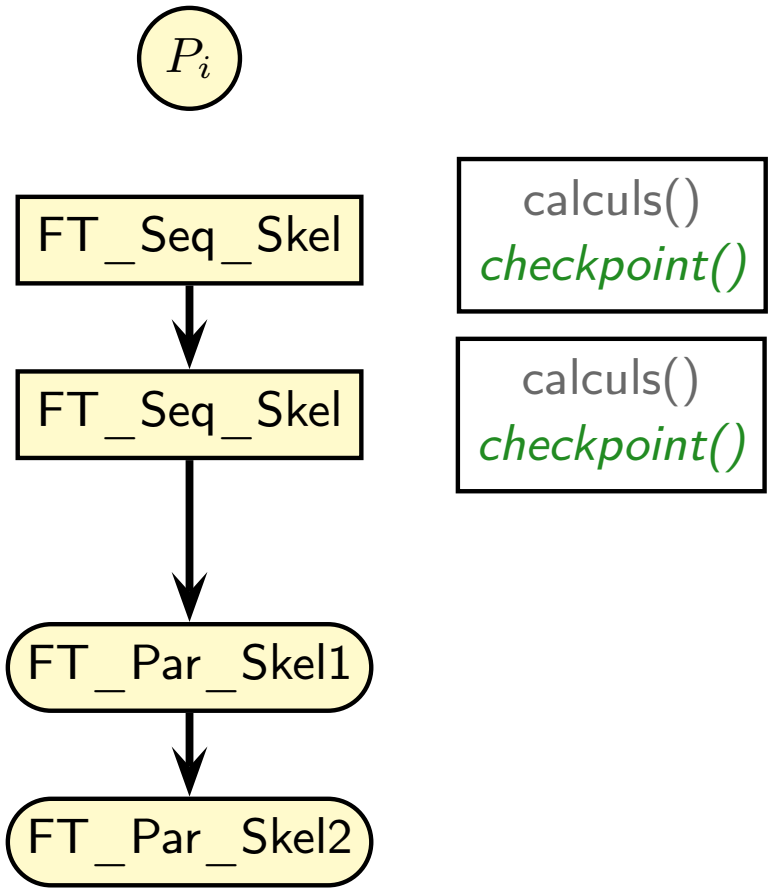
- Détermination de la **ligne de reprise**
- **Réexécution sélective** recouvrement du contexte du processus :
  1. Réexécution d'opérations légères
  2. Omission d'opérations lourdes déjà exécutées



Reprise

Mode d'exécution de reprise

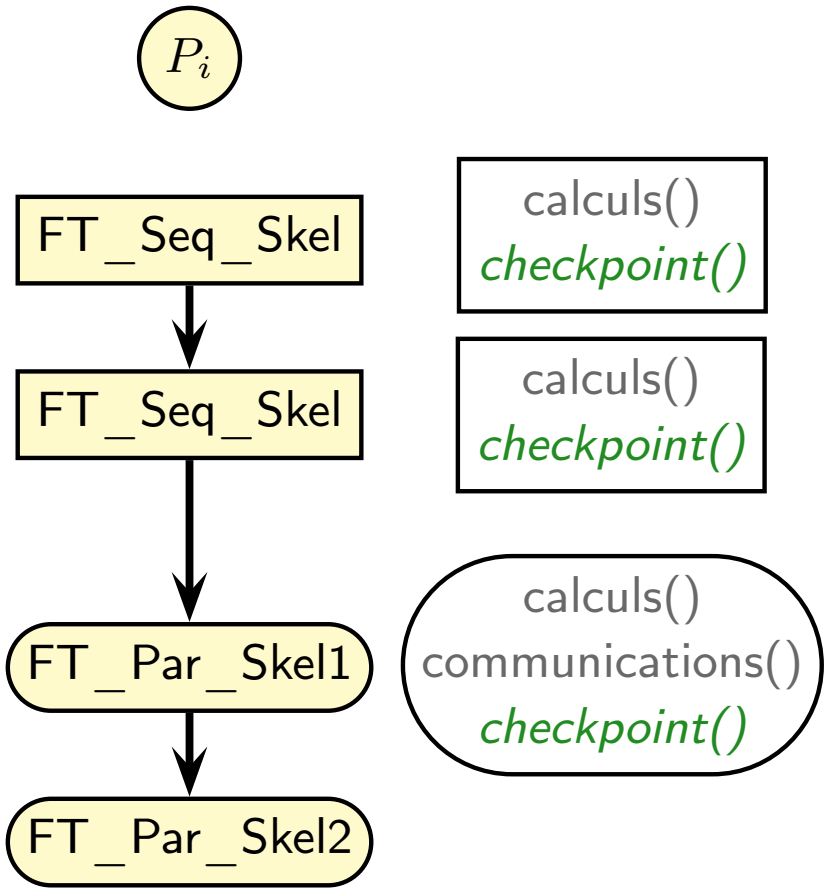
- Détermination de la **ligne de reprise**
- **Réexécution sélective** recouvrement du contexte du processus :
  1. Réexécution d'opérations légères
  2. Omission d'opérations lourdes déjà exécutées



Reprise

Mode d'exécution de reprise

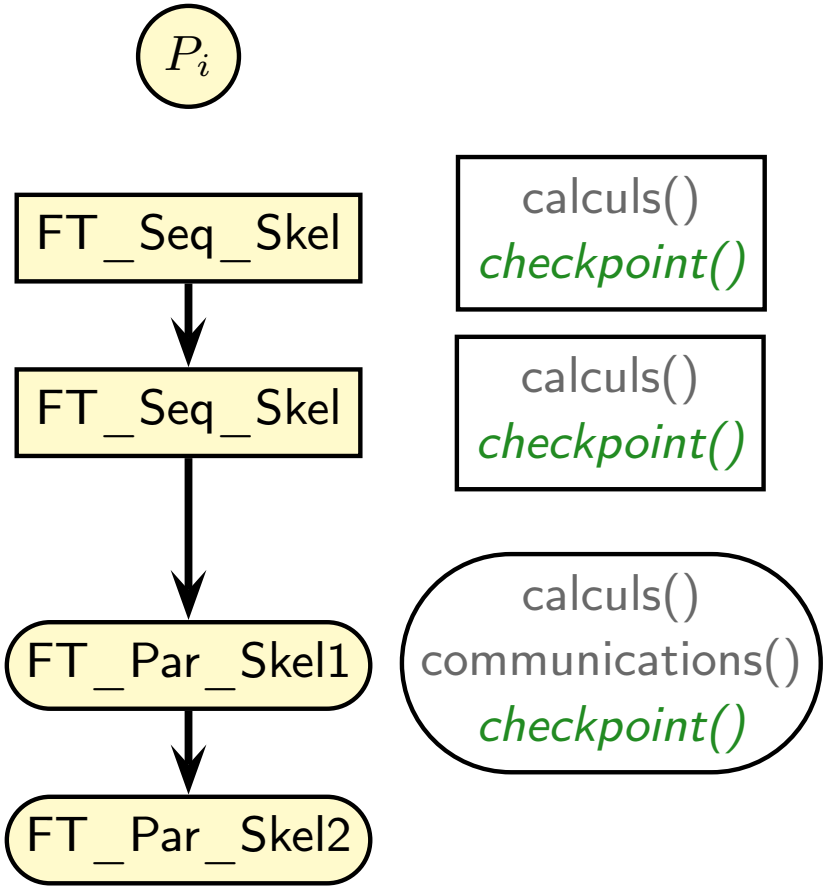
- Détermination de la **ligne de reprise**
- **Réexécution sélective** recouvrement du contexte du processus :
  1. Réexécution d'opérations légères
  2. Omission d'opérations lourdes déjà exécutées



Reprise

Mode d'exécution de reprise

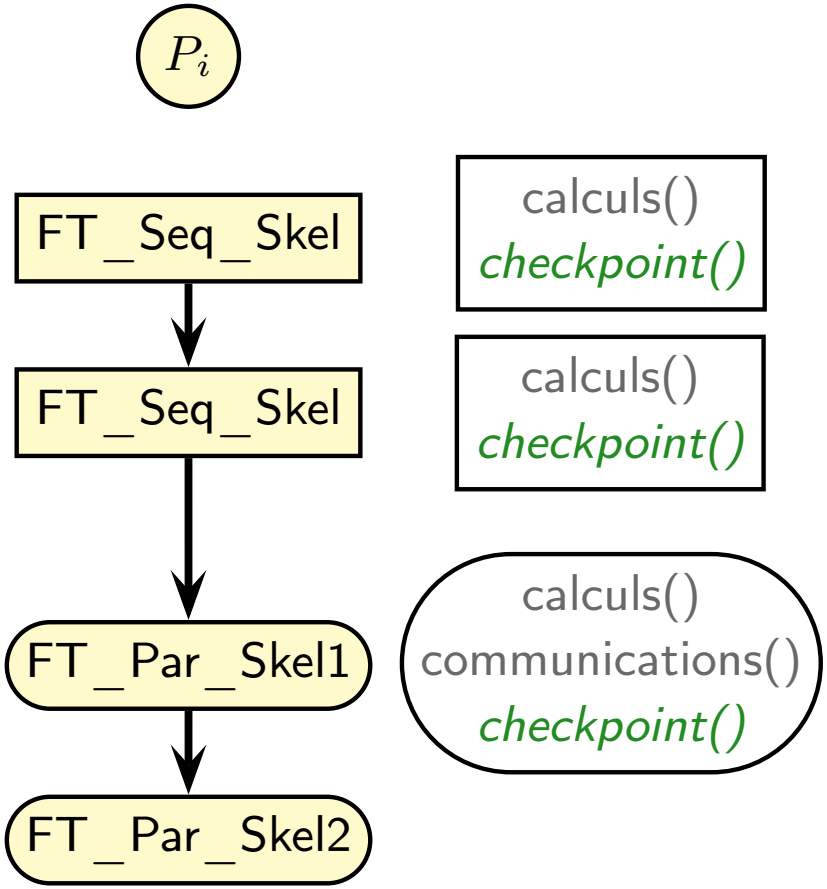
- Détermination de la **ligne de reprise**
- **Réexécution sélective** recouvrement du contexte du processus :
  1. Réexécution d'opérations légères
  2. Omission d'opérations lourdes déjà exécutées
  3. Chargement des données sauvées au bon point de reprise



Reprise

Mode d'exécution de reprise

- Détermination de la **ligne de reprise**
- **Réexécution sélective** recouvrement du contexte du processus :
  1. Réexécution d'opérations légères
  2. Omission d'opérations lourdes déjà exécutées
  3. Chargement des données sauvées au bon point de reprise
  4. Retour en *mode d'exécution normal*



Plan

Collaborations « Programmeur–Framework »

Assistance du programmeur requise

1. Collaboration de **placement**
   - Où placer les squelettes ?
2. Collaboration de **correction** et d'**efficacité**
   - Quelles données inclure dans les sauvegardes ?
3. Collaboration de **fréquence**
   - À quelle fréquence réaliser les sauvegardes ?

Collaborations « Environnement–Framework »

Considération d'informations de FT externes

- Capacité à être **piloté** pour ajuster la FT
- Exemples :
  - Sauvegarde à la demande et modification de la fréquence des sauvegardes
  - Requêtes de l'administrateur / écosystème FT
    (*e.g. : opération de maintenance, prédiction de panne*)

Points clés de MoLOToF

Structure de l'application parallèle

- Concept de squelette tolérant aux pannes
- Opérations lourdes et légères
- Organisation selon des squelettes

Mécanisme de sauvegarde/reprise

- Mode d'exécution normal et de reprise
- Réexécution sélective

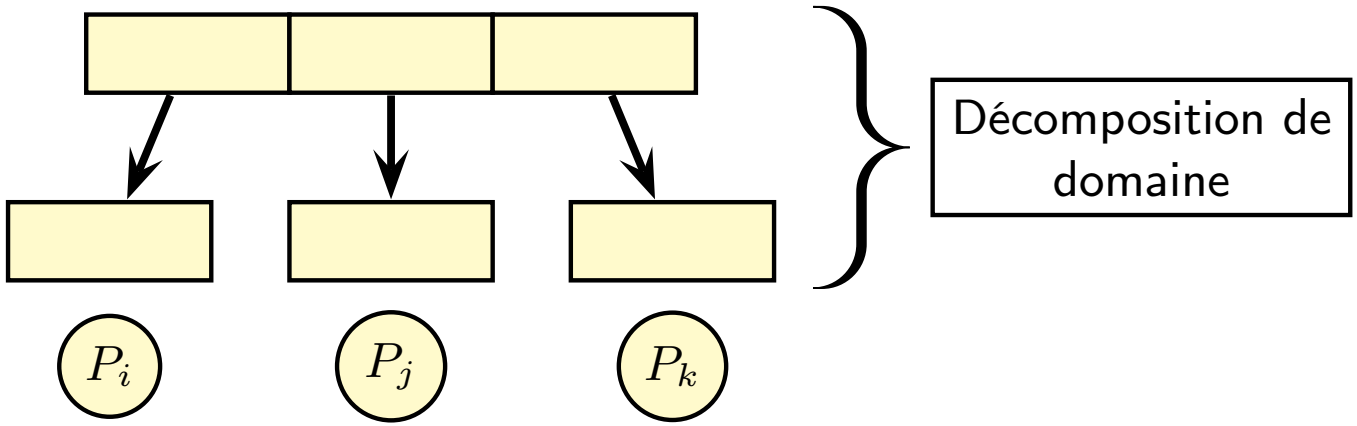
Tolérance aux pannes ajustable

- Collaborations « Programmeur–Framework »
- Collaborations « Environnement–Framework »

Plan

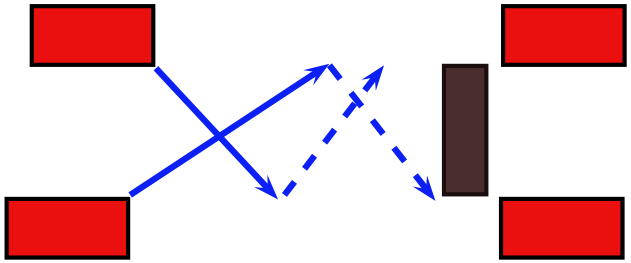
Parallélisation SPMD

- SPMD : **S***ingle* **P***rogram* **M***ultiple* **D***ata*



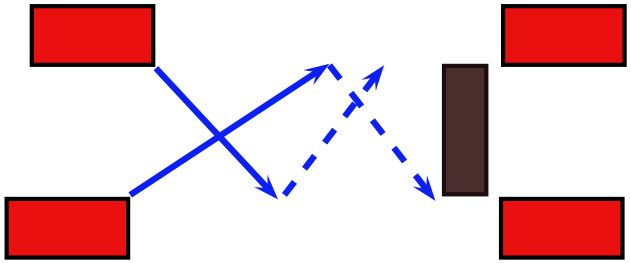
Modèles de calcul distribué : BSP, GReLoSSS, PRO

Modèles de calcul distribué : BSP, GReLoSSS, PRO

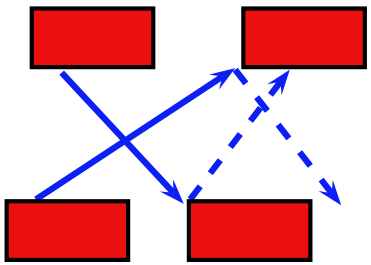


BSP : **B***ulk* **S***ynchronous* **P***arallel*

Modèles de calcul distribué : BSP, GReLoSSS, PRO

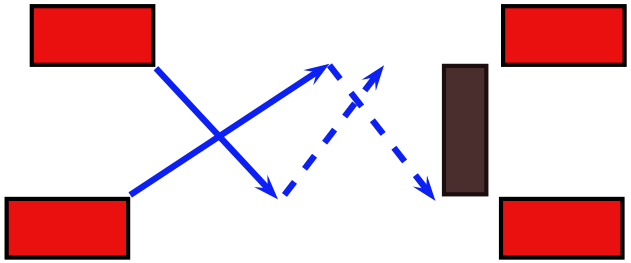


BSP : Bulk Synchronous Parallel

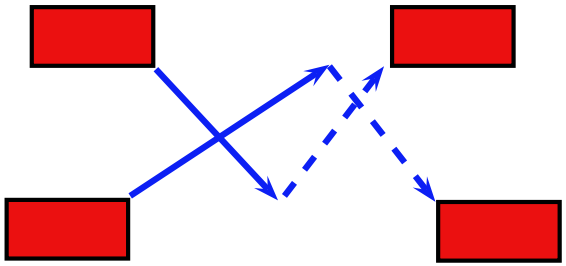


PRO : Parallel Ressource-Optimal

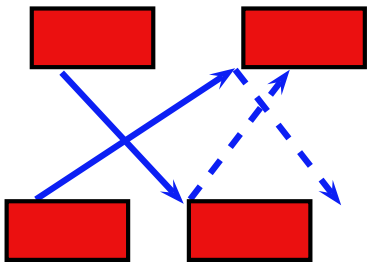
Modèles de calcul distribué : BSP, GReLoSSS, PRO



BSP : **B***ulk* **S***ynchronous* **P***arallel*

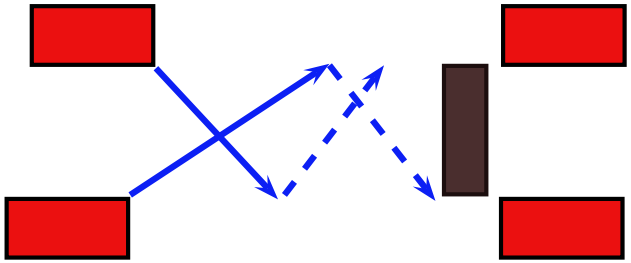


GReLoSSS : **G***lobally* **Re***laxed* **Lo***cally* **S***trict* **S***ynchronization* **S***PMD*

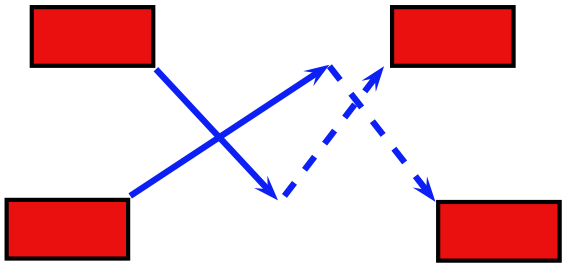


PRO : **P***arallel* **R***essource-***O***ptimal*

Modèles de calcul distribué : BSP, GReLoSSS, PRO



BSP : **B***ulk* **S***ynchronous* **P***arallel*

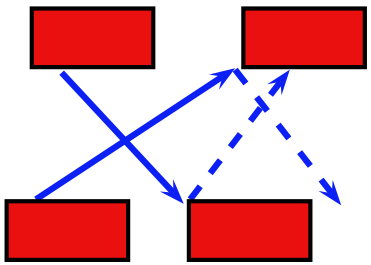


GReLoSSS : **G***lobally* **Re***laxed* **Lo***cally* **S***trict* **S***ynchronization* **S***PMD*

Profiter de la synchronisation relachée

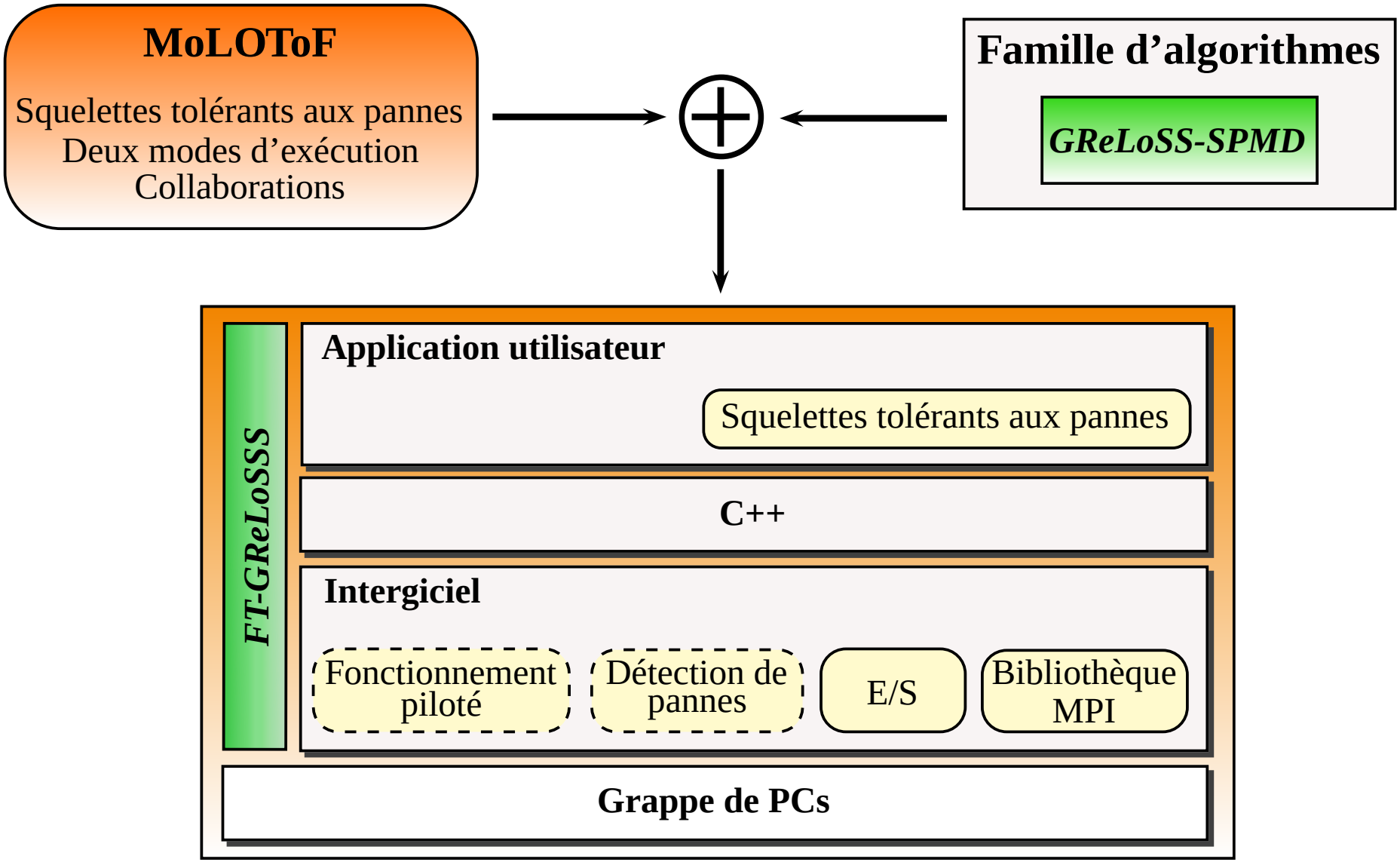
Evite la présence de message pendant la phase de calcul . . .

. . . très utile pour la réalisation de points de reprise



PRO : **P***arallel* **R***essource-***O***ptimal*

Architecture du framework



Squelettes et modèle de parallélisation

```
FT_Par_Skel
{
  FT_Loop
  {
    calculs()
    communications()
    swap()
    checkpoint()
  }
}
```

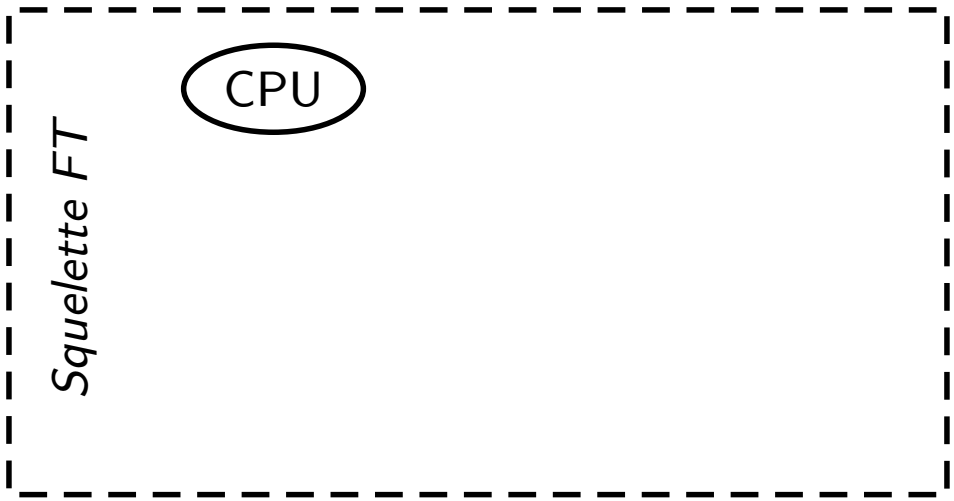
Nombre de super-étapes connu/inconnu

Squelettes et modèle de parallélisation

1 Calculs

```
FT_Par_Skel
{
  FT_Loop
  {
    calculs()
    communications()
    swap()
    checkpoint()
  }
}
```

Nombre de super-étapes connu/inconnu

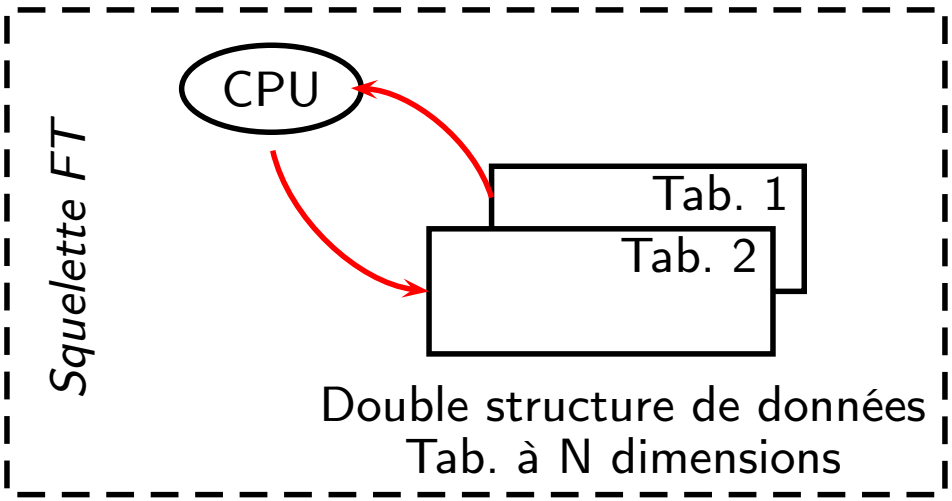


Squelettes et modèle de parallélisation

1 Calculs

```
FT_Par_Skel
{
  FT_Loop
  {
    calculs()
    communications()
    swap()
    checkpoint()
  }
}
```

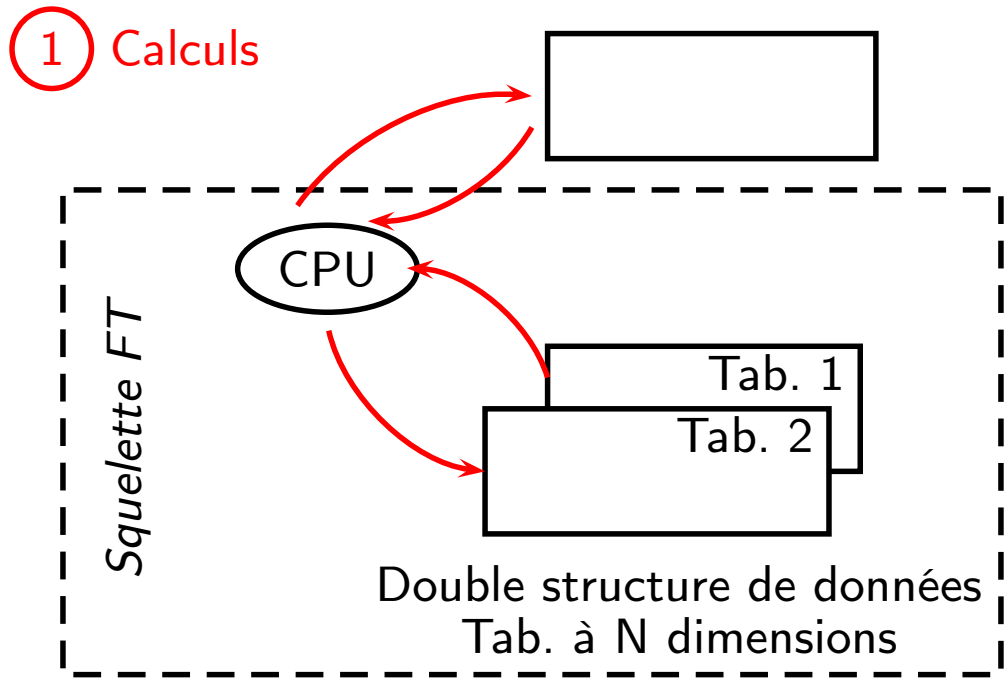
Nombre de super-étapes connu/inconnu



Squelettes et modèle de parallélisation

```
FT_Par_Skel
{
  FT_Loop
  {
    calculs()
    communications()
    swap()
    checkpoint()
  }
}
```

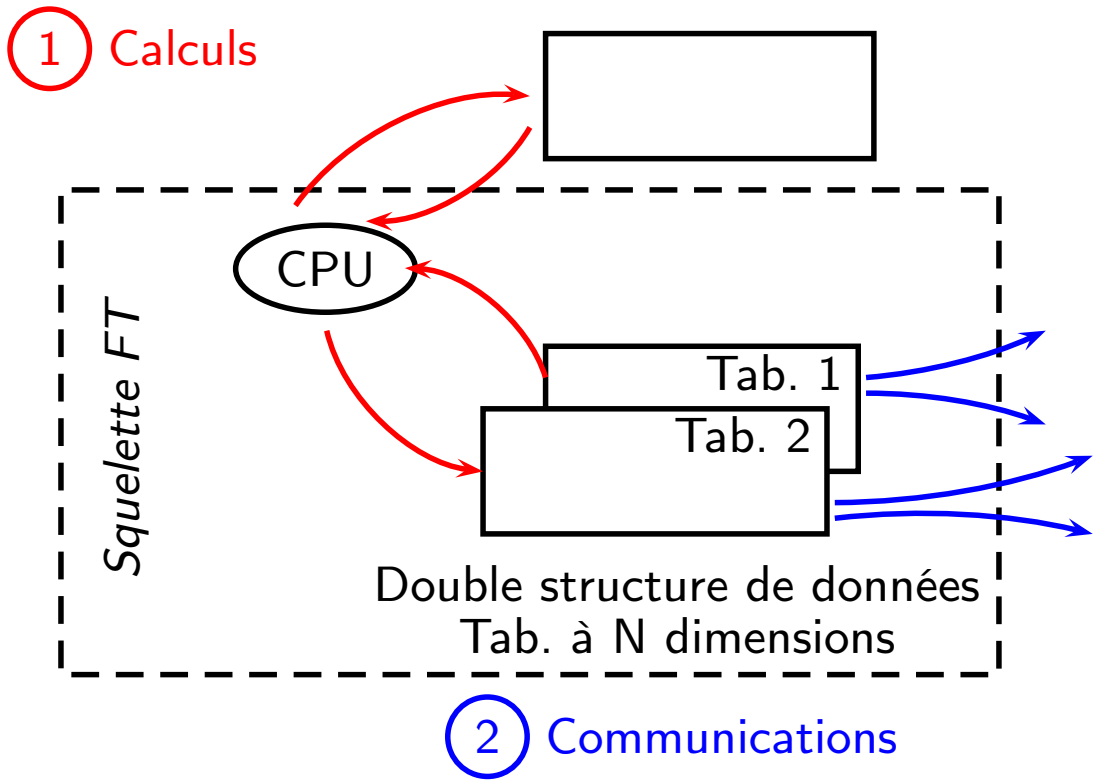
Nombre de super-étapes connu/inconnu



Squelettes et modèle de parallélisation

```
FT_Par_Skel
{
  FT_Loop
  {
    calculs()
    communications()
    swap()
    checkpoint()
  }
}
```

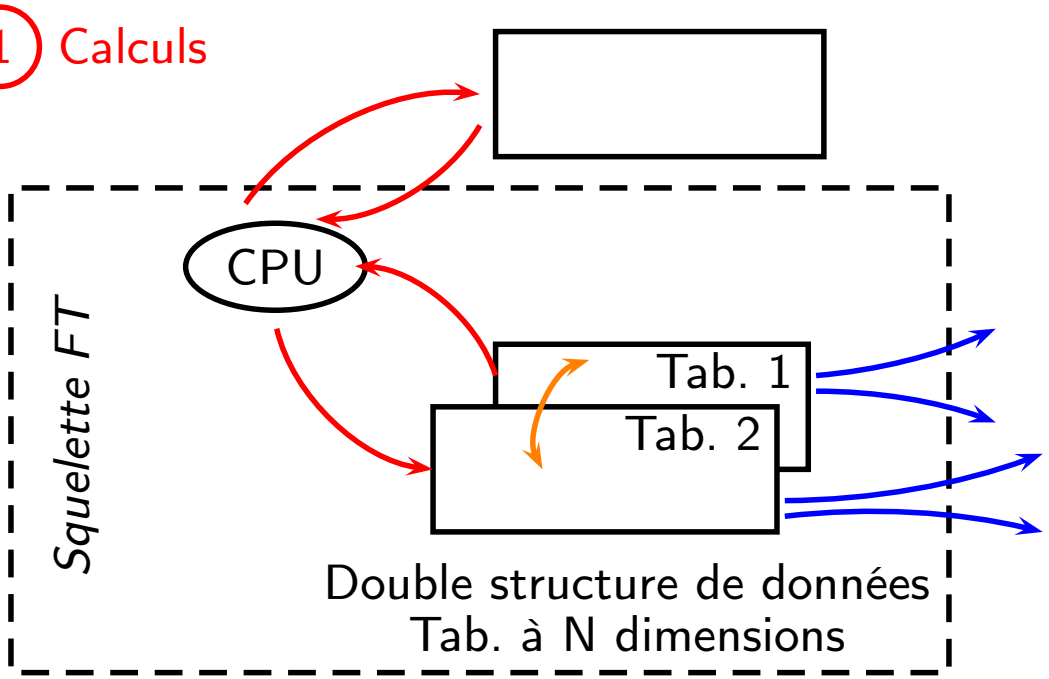
Nombre de super-étapes connu/inconnu



Squelettes et modèle de parallélisation

```
FT_Par_Skel
{
  FT_Loop
  {
    calculs()
    communications()
    swap()
    checkpoint()
  }
}
```

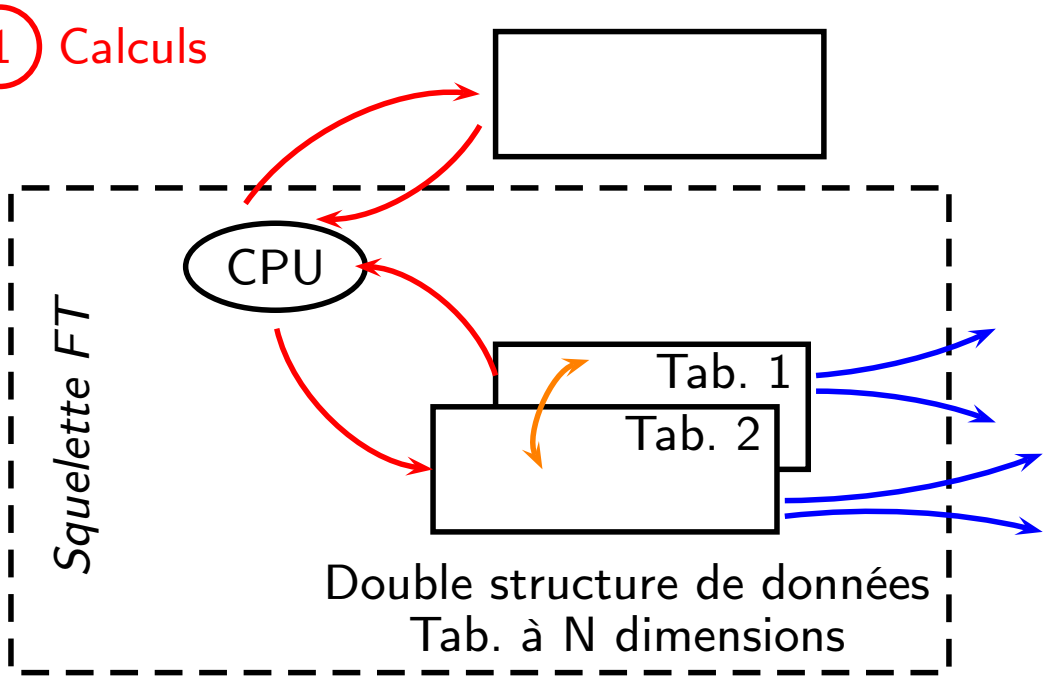
Nombre de super-étapes connu/inconnu



Squelettes et modèle de parallélisation

```
FT_Par_Skel
{
  FT_Loop
  {
    calculs()
    communications()
    swap()
    checkpoint()
  }
}
```

Nombre de super-étapes connu/inconnu



Fonctionnalités de tolérance aux pannes (FT-GReLoSSS)

Protocole de cohérence

- Protocole avec coordination sans échange de messages
  - SPMD assure la réalisation des sauvegardes au même point
  - FT-GReLoSSS assure l'absence de messages
    $\longrightarrow$ Protocole simple, efficace et extensible

Collaborations « Programmeur – Framework »

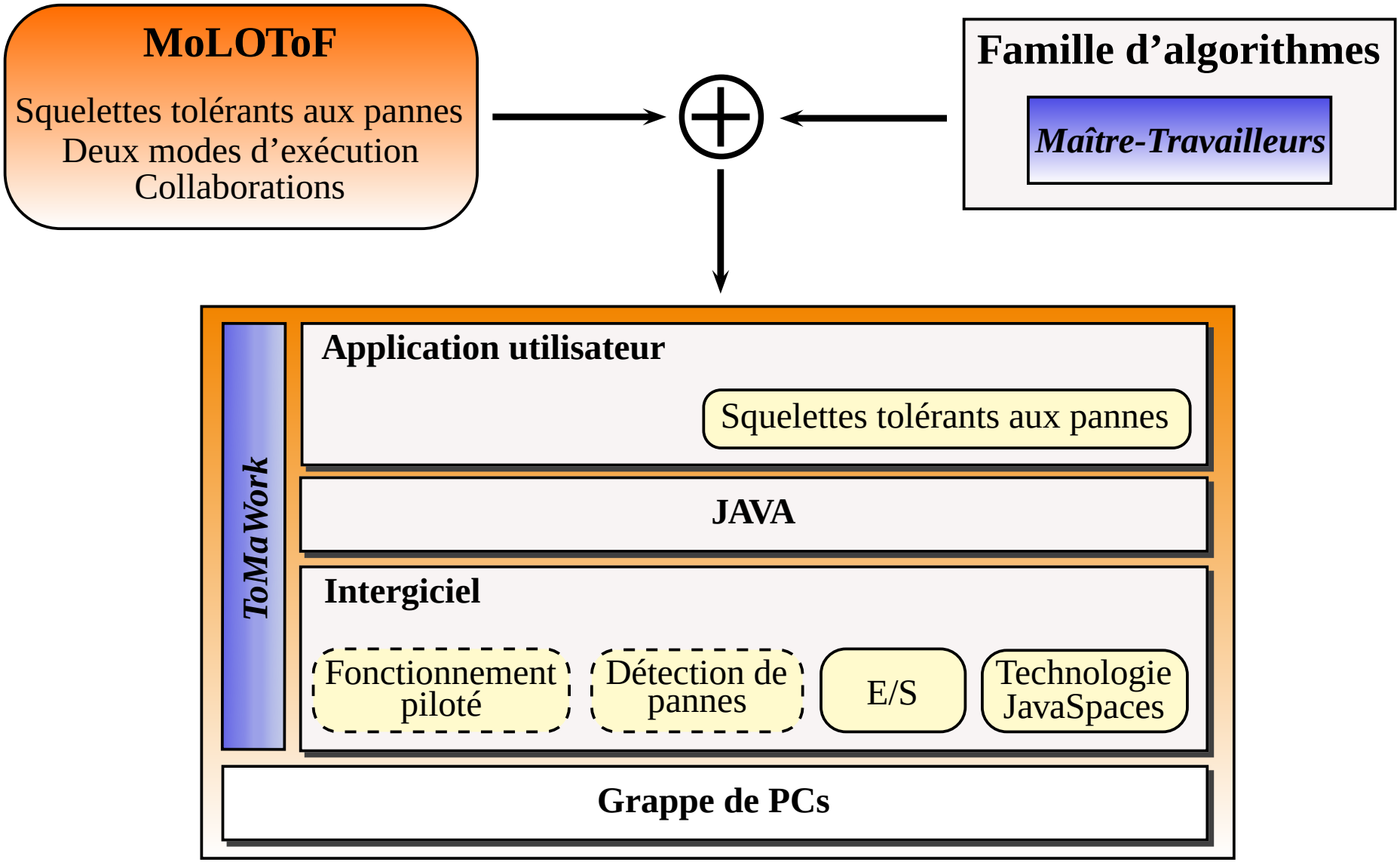
- Via interface fournie par les squelettes
  - Sélection des données à sauver
  - Établissement de la fréquence des sauvegardes

Collaborations « Environnement – Framework »

- Protocole original pour *fonctionnement piloté*
  - Fondé sur l'organisation en squelettes de l'application
  - Mise en œuvre en cours

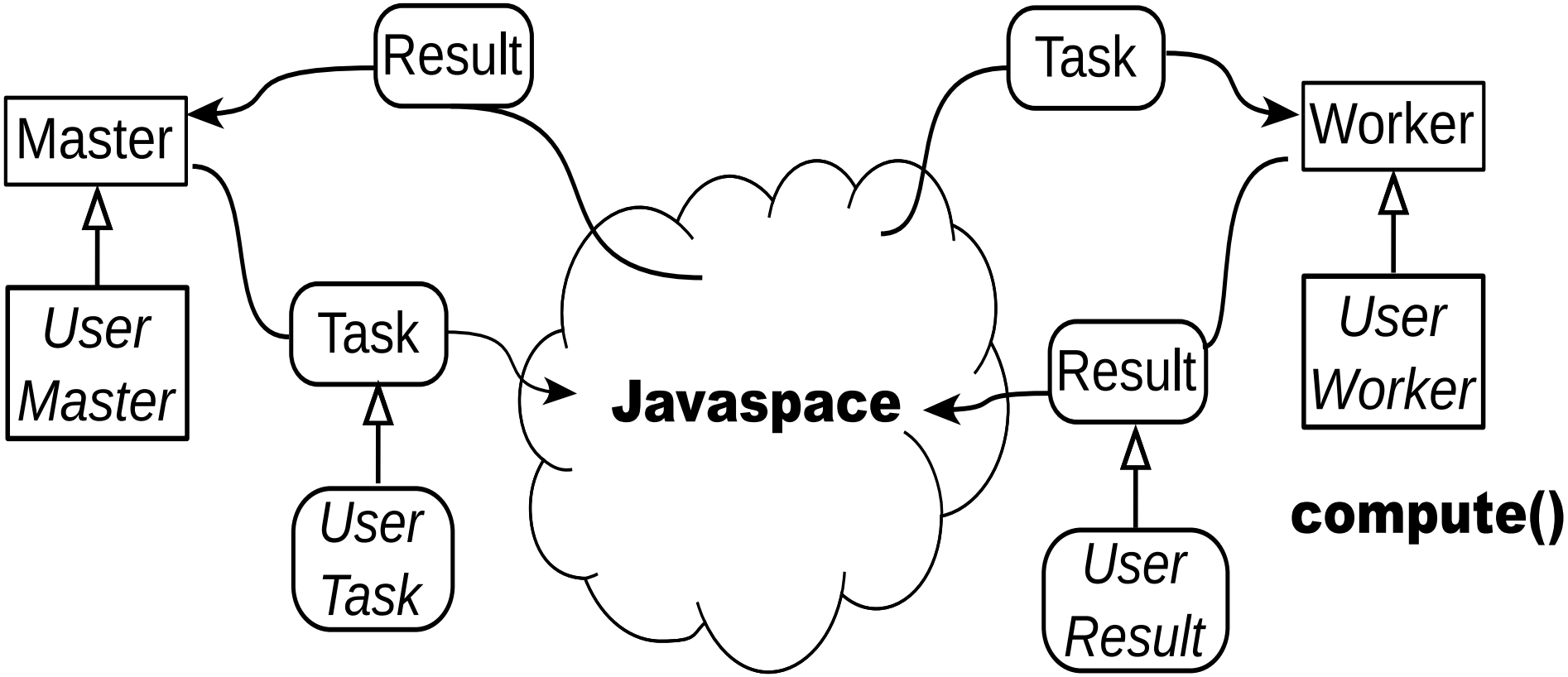
Plan

Architecture du framework



Modèle de parallélisation

checkTermination(): True/False



Fonctionnalités de tolérance aux pannes (ToMaWork)

Protocole de cohérence

- Maintient à tout moment la cohérence entre Maître, Javaspace et Travailleurs
- Combinaison de sauvegardes locales et de transactions

Collaborations « Programmeur – Framework »

- (Dés)activation des sauvegardes et/ou des transactions
- Introduction de sauvegardes intermédiaires dans la méthode `compute`

Solution de tolérance aux pannes par défaut
dont peuvent bénéficier les applications immédiatement

Plan

Plan

Critères d'évaluation

- Correction de la tolérance aux pannes
- Facilité de développement
- Performances
  - Sans tolérance aux pannes
  - Avec tolérance aux pannes $\longrightarrow$ avec sauvegardes
    - Sans panne
    - Avec panne

Plan

Méthodologie

- **Métrique** : nombre de lignes de code source logiques
- **Comparaison** : versions de l'application avec et sans framework

Résultats pour FT-GReLoSSS

	Sans framework	Avec framework	Surcoût absolu	Surcoût relatif (%)
Matmult	168	186	+18 *lignes*	+10.7
Jacobi	179	190	+11 *lignes*	+6.1

- Surcoûts acceptables :
  1. Inclusion de la tolérance aux pannes
  2. Présence d'instructions à faible complexité algorithmique
  3. Pour un même schéma avec des calculs plus complexes, le nombre de lignes reste faible

Résultats pour FT-GReLoSSS

	Sans framework	Avec framework	Surcoût absolu	Surcoût relatif (%)
Matmult	168	186	+18 *lignes*	+10.7
Jacobi	179	190	+11 *lignes*	+6.1

- Surcoûts acceptables :
  1. Inclusion de la tolérance aux pannes
  2. Présence d'instructions à faible complexité algorithmique
  3. Pour un même schéma avec des calculs plus complexes, le nombre de lignes reste faible

Résultats pour FT-GReLoSSS

	Sans framework	Avec framework	Surcoût absolu	Surcoût relatif (%)
Matmult	168	186	+18 *lignes*	+10.7
Jacobi	179	190	+11 *lignes*	+6.1

- Surcoûts acceptables :
  1. Inclusion de la tolérance aux pannes
  2. Présence d'instructions à faible complexité algorithmique
  3. Pour un même schéma avec des calculs plus complexes, le nombre de lignes reste faible

Résultats pour ToMaWork

	Sans framework	Avec framework	Surcoût absolu	Surcoût relatif (%)
NQueens	408	308	-100 *lignes*	-24.5

- Gain très intéressant :
  1. Inclusion d'une tolérance aux pannes directement opérationnelle
  2. Absence de code lié aux Javaspaces
     - Découverte de Javaspace
     - Écritures et retraits concurrents (tâches et résultats)
     - Code de gestion de terminaison « propre » de l'application

  $\longrightarrow$ simplification du code source par diminution du code lié aux Javaspaces
  $\longrightarrow$ -100 *lignes* sur cette petite application

Résultats pour ToMaWork

	Sans framework	Avec framework	Surcoût absolu	Surcoût relatif (%)
NQueens	408	308	-100 *lignes*	-24.5

- Gain très intéressant :
  1. Inclusion d'une tolérance aux pannes directement opérationnelle
  2. Absence de code lié aux Javaspaces
     - Découverte de Javaspace
     - Écritures et retraits concurrents (tâches et résultats)
     - Code de gestion de terminaison « propre » de l'application

  $\longrightarrow$ simplification du code source par diminution du code lié aux Javaspaces
  $\longrightarrow$ -100 *lignes* sur cette petite application

Résultats pour ToMaWork

	Sans framework	Avec framework	Surcoût absolu	Surcoût relatif (%)
NQueens	408	308	-100 *lignes*	-24.5

- Gain très intéressant :
  1. Inclusion d'une tolérance aux pannes directement opérationnelle
  2. Absence de code lié aux Javaspaces
     - Découverte de Javaspace
     - Écritures et retraits concurrents (tâches et résultats)
     - Code de gestion de terminaison « propre » de l'application

  $\longrightarrow$ simplification du code source par diminution du code lié aux Javaspaces
  $\longrightarrow$ -100 *lignes* sur cette petite application

Plan

Environnement expérimental I

Plate-forme d'essai

- *Intercell* la grappe de 256 PCs à Supélec (4 Go, 1 Gigabit Ethernet)

Application benchmark : *Matmult*

- Produit de matrices denses en parallèle sur un anneau de PCs : $C = A \times B$

Taille matrice individuelle	16384×16384	32768×32768	65536×65536
Taille totale de l'application en RAM	$\sim$ 6 Go	$\sim$ 24 Go	$\sim$ 96 Go
Taille totale d'une sauvegarde par FT-GReLoSSS	$\sim$ 4 Go	$\sim$ 16 Go	$\sim$ 70 Go

Environnement expérimental I

Plate-forme d'essai

- *Intercell* la grappe de 256 PCs à Supélec (4 Go, 1 Gigabit Ethernet)

Application benchmark : *Matmult*

- Produit de matrices denses en parallèle sur un anneau de PCs : $C = A \times B$

Taille matrice individuelle	16384×16384	32768×32768	65536×65536
Taille totale de l'application en RAM	$\sim$ 6 Go	$\sim$ 24 Go	$\sim$ 96 Go
Taille totale d'une sauvegarde par FT-GReLoSSS	$\sim$ 4 Go	$\sim$ 16 Go	$\sim$ 70 Go

Sauvegardes plus légères grâce aux collaborations Programmeur–Framework

Environnement expérimental II

Comparaison de réalisation : niveau système et applicatif

- **FT-GReLoSSS** with Open MPI 1.3.3 (OMPI FT-GReLoSSS)
- **LAM/MPI** 7.1.4 (LAM/MPI)
- **DMTCP** r481 with Open MPI 1.3.3 (DMTCP OMPI)

Autres réalisations testées

- **Open MPI** FT non entièrement opérationnelle au moment des tests
- **MPICH-V** non maintenu et incompatible avec nos installations Linux

Sans tolérance aux pannes

Taille des matrices	Nombre de nœuds	T_{exec} (secondes)		Framework FT-GReLoSSS Surcoût relatif (%)
		OMPI	OMPI FT-GReLoSSS	
16384×16384	4	2027	2027	0.0
	8	1025	1027	0.3
	16	522	526	0.7
	32	274	277	0.9
32768×32768	32	2107	2113	**0.3**
	64	1094	1103	**0.8**
	128	597	609	**1.9**
	256	352	362	**3.0**
65536×65536	64	8405	8439	0.4
	128	4444	4469	0.6
	256	2406	2445	1.6

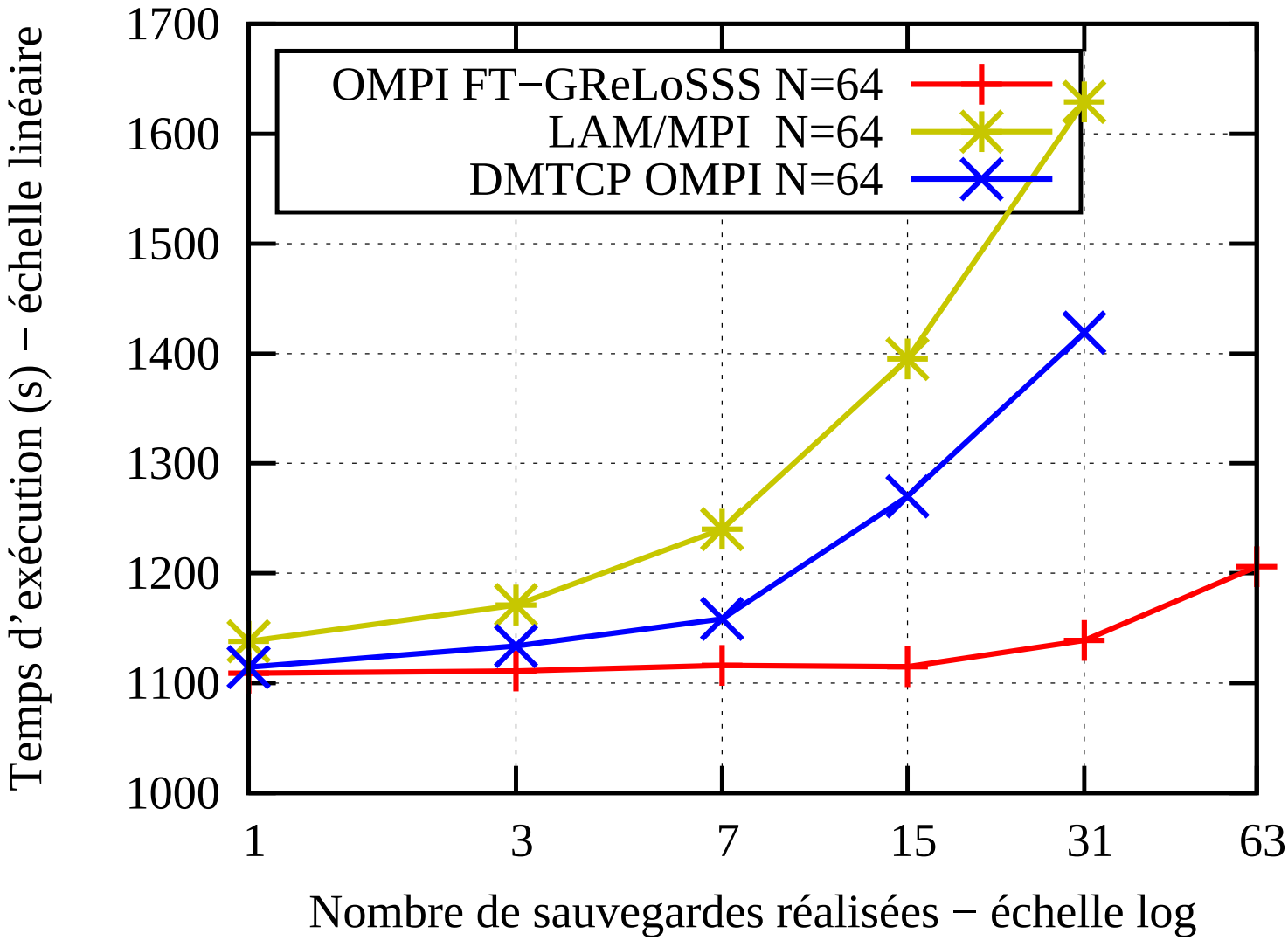
Sans tolérance aux pannes

Taille des matrices	Nombre de nœuds	T_{exec} (secondes)		Framework FT-GReLoSSS Surcoût relatif (%)
		OMPI	OMPI FT-GReLoSSS	
16384×16384	4	2027	2027	0.0
	8	1025	1027	0.3
	16	522	526	0.7
	32	274	277	0.9
32768×32768	32	2107	2113	**0.3**
	64	1094	1103	**0.8**
	128	597	609	**1.9**
	256	352	362	**3.0**
65536×65536	64	8405	8439	0.4
	128	4444	4469	0.6
	256	2406	2445	1.6

Surcoûts faibles $<4\%$

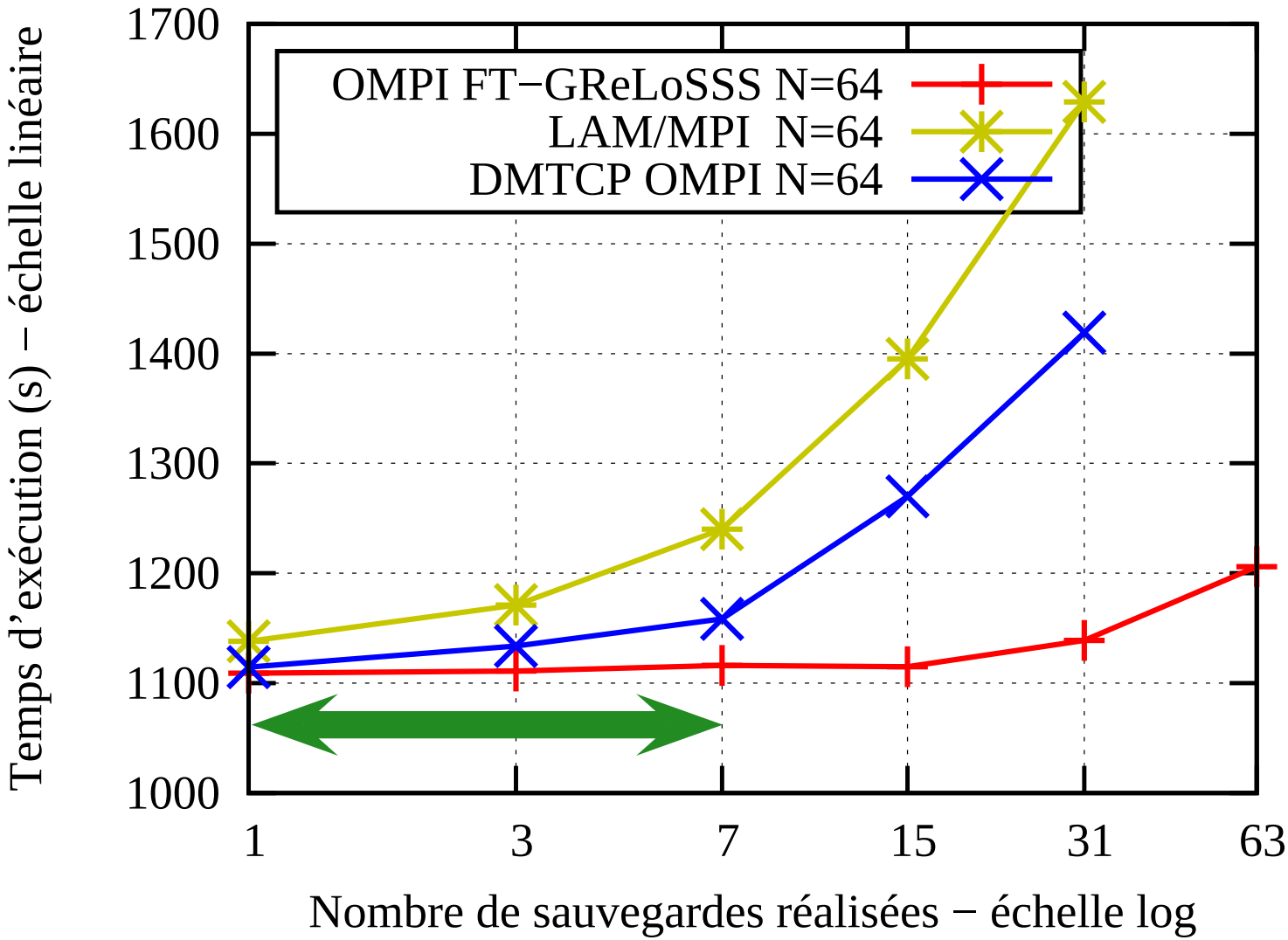
Avec sauvegardes – Sans panne

- 32768×32768 (24 Go) - 64 Nœuds



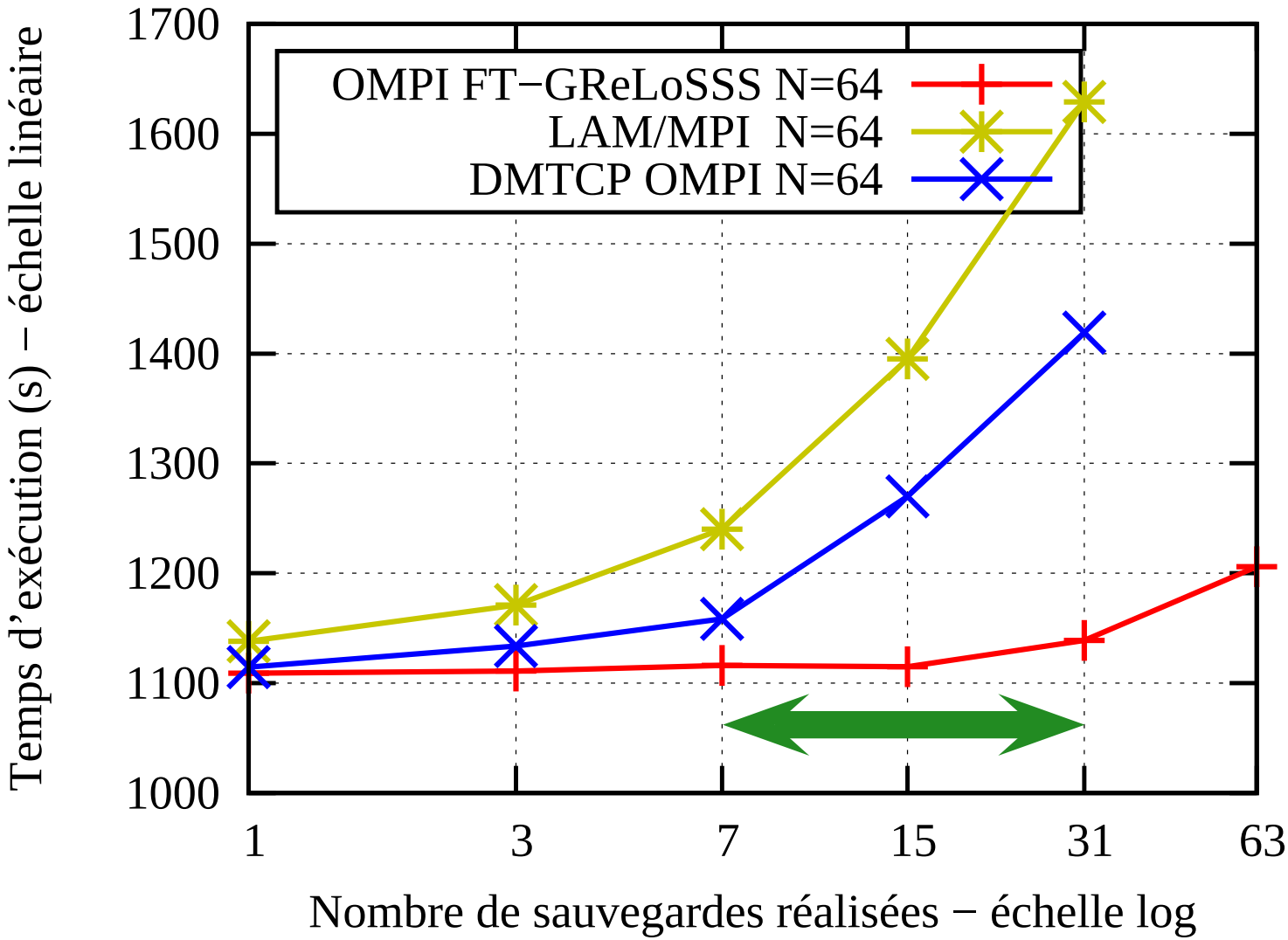
Avec sauvegardes – Sans panne

- 32768×32768 (24 Go) - 64 Nœuds



Avec sauvegardes – Sans panne

- 32768×32768 (24 Go) - 64 Nœuds



Avec sauvegardes – Une panne

Surcoût de reprise

- $T_{surcout\_reprise} = T_{detection\_panne} + T_{reprise}$

Expérience

- Évaluation de $T_{reprise}$ après une seule panne (absence de détection et relance auto)
- Mesures de $T_{reprise}$ plus difficiles que prévu
  - Cadre distribué
  - Hétérogénéité des niveaux (système vs. applicatif)

Résultats

- Surcoûts négligeables pour LAM/MPI et FT-GReLoSSS
- Reprise impossible de DMTCP sur *Intercell*

Avec sauvegardes – Une panne

Surcoût de reprise

- $T_{surcout\_reprise} = T_{detection\_panne} + T_{reprise}$

Expérience

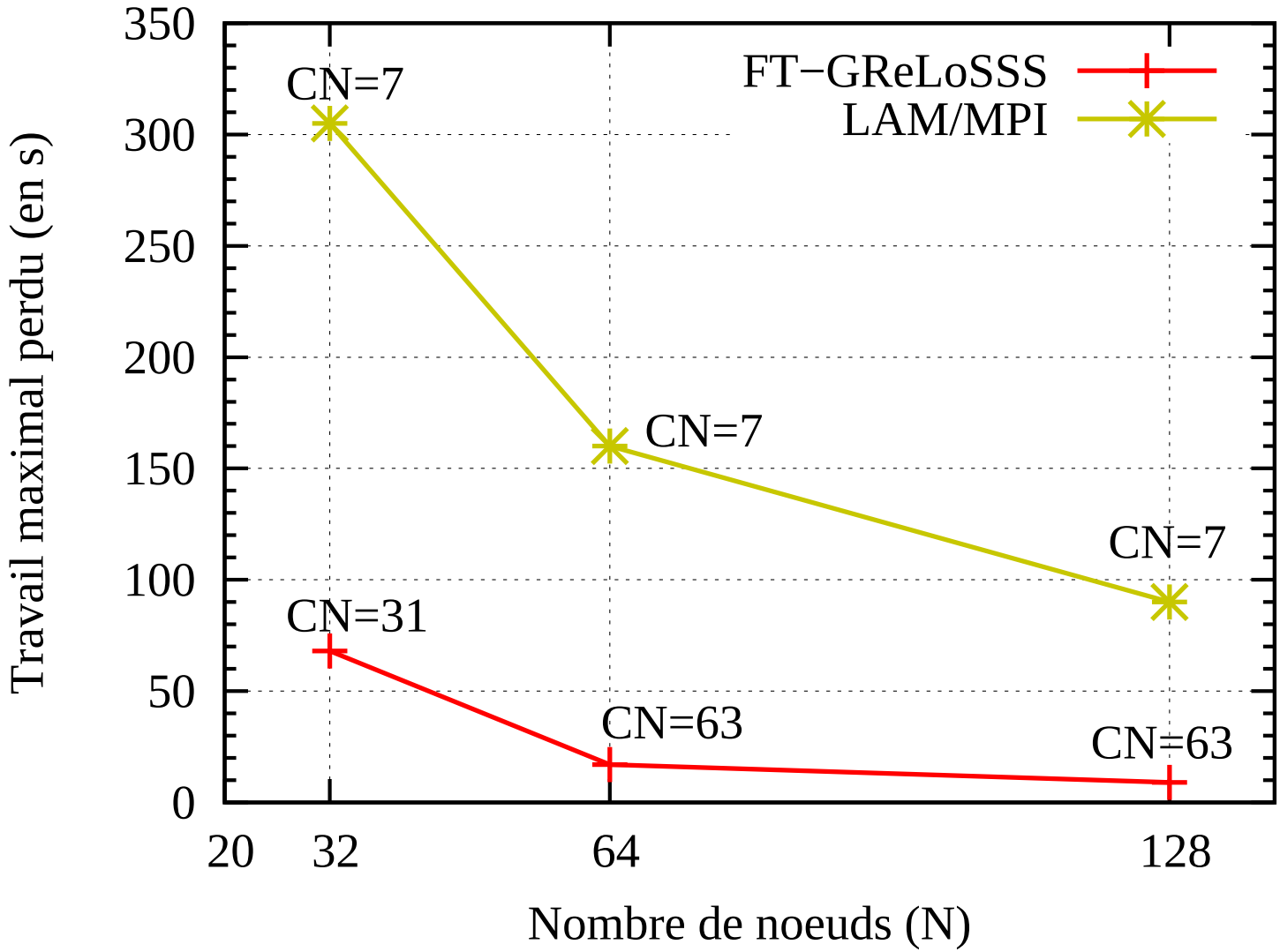
- Évaluation de $T_{reprise}$ après une seule panne (absence de détection et relance auto)
- Mesures de $T_{reprise}$ plus difficiles que prévu
  - Cadre distribué
  - Hétérogénéité des niveaux (système vs. applicatif)

Résultats

- Surcoûts négligeables pour LAM/MPI et FT-GReLoSSS
- Reprise impossible de DMTCP sur *Intercell*

Évaluation à surcoût constant de 10%

- 32768×32768 (24 Go)



Premiers résultats pour ToMaWork

- Application synthétique : variation du nombre et de la taille des tâches
- Paramètres : 10000 tâches de 1000 `double` d'une durée de 30s

  $\longrightarrow$ comparables à application réelle
- Tolérance aux pannes totale, sauf sur le maître :
  - Son approche de sauvegarde actuelle est inefficace car non incrémentale
  - Revient au cas où le maître s'exécute sur un nœud de fiabilité élevée

Nombres de travailleurs	**Temps d'exécution (s)**		
	Théorique	**Expérimental sans FT**	**Expérimental avec FT**
32	9390	9392	9411
64	4710	4712	4725
128	2370	2372	2389

Premiers résultats pour ToMaWork

- Application synthétique : variation du nombre et de la taille des tâches
- Paramètres : 10000 tâches de 1000 `double` d'une durée de 30s

  $\longrightarrow$ comparables à application réelle
- Tolérance aux pannes totale, sauf sur le maître :
  - Son approche de sauvegarde actuelle est inefficace car non incrémentale
  - Revient au cas où le maître s'exécute sur un nœud de fiabilité élevée

Nombres de travailleurs	Temps d'exécution (s)		
	Théorique	Expérimental sans FT	Expérimental avec FT
32	9390	9392	9411
64	4710	4712	4725
128	2370	2372	2389

Surcoûts négligeables $<0.8\%$

Plan

Synthèse

Tolérance aux pannes dans les grappes de PCs

MoLOToF

Squelettes tolérants aux pannes
Deux modes d'exécution
Collaborations

Familles d'algorithmes

GReLoSS-SPMD

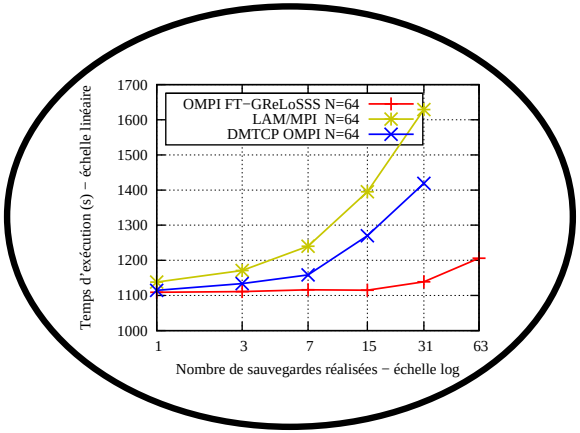
Maître-Travailleurs

FT-GReLoSSS
MPI

ToMaWork
Javaspaces

Frameworks

Surcoût de développement limité
Efficacité, Portabilité



Perspectives I

Facilité de développement

- Imposition de structure supplémentaire à certains niveaux pour
  - Faciliter la spécification des fonctions de partitionnement (FT-GReLoSSS)
  - Automatiser l'écriture de la méthode `compute` de reprise (ToMaWork)
- Évaluation alternative avec un public de programmeurs non spécialistes

Collaborations

- Enrichir les collaborations « Programmeur–Framework » existantes
  - *e.g.* : manière de sauver les données plus futée
- Rendre effectives les collaborations « Environnement–Framework »

Perspectives II

Applications réelles

- Pour FT-GReLoSSS,
  - Application de *Swing Gazier* développée par EDF R&D et Supélec
  - Communications irrégulières
- Pour ToMaWork,
  - Applications de *machine learning* développées à Supélec (algorithme *AdaBoost*)

MoLOToF

- Extension du modèle pour le support de contextes imbriqués

Publications

V. Galtier, C. Makassikis et S. Vialle
A Javaspace-based Framework for Efficient Fault-tolerant Master-Worker Distributed Applications, PDP 2011, (to appear)

C. Makassikis, V. Galtier et S. Vialle
A Skeletal-Based Approach for the Development of Fault-Tolerant SPMD Applications, PDCAT 2010

C. Makassikis
Distribution Large Échelle d'un Algorithme Financier de Contrôle Stochastique, RenPar'18

S. Vialle, X. Warin et C. Makassikis
Large Scale Distribution of Stochastic Control Algorithms for Financial Applications, PDCoF-IPDPS 2008

C. Makassikis, X. Warin et S. Vialle
Distribution of a Stochastic Control Algorithm Applied to Gas Storage Valuation, ISSPIT 2007