Ordonnancement hybride statique-dynamique en algèbre linéaire creuse pour de grands clusters de machines NUMA et multi-cœurs

Mathieu Faverge

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

LaBRI

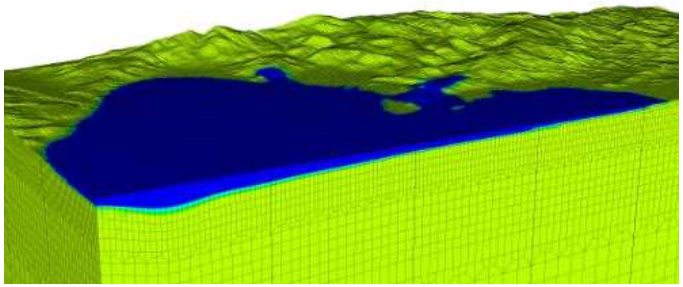
NUMASIS

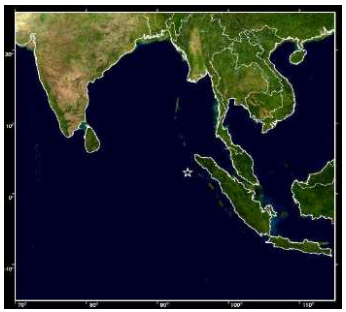
7 décembre 2009

Contexte : ANR NUMASIS

Objectif : Simuler des tremblements de terre sur des clusters de machines NUMA.
Partenaires : BRGM, Bull, CEA, Total et les laboratoires ID-IMAG, IRISA et LaBRI

- Modélisation de la croûte et du manteau terrestre
- Étude de la propagation des ondes dans les différentes couches
- Nécessite une quantité importante de calcul et de mémoire

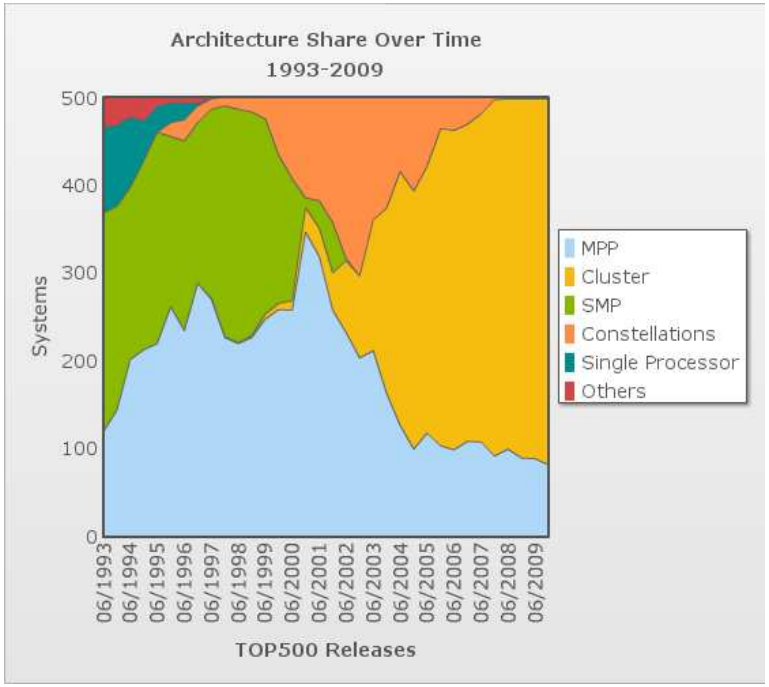




Plan

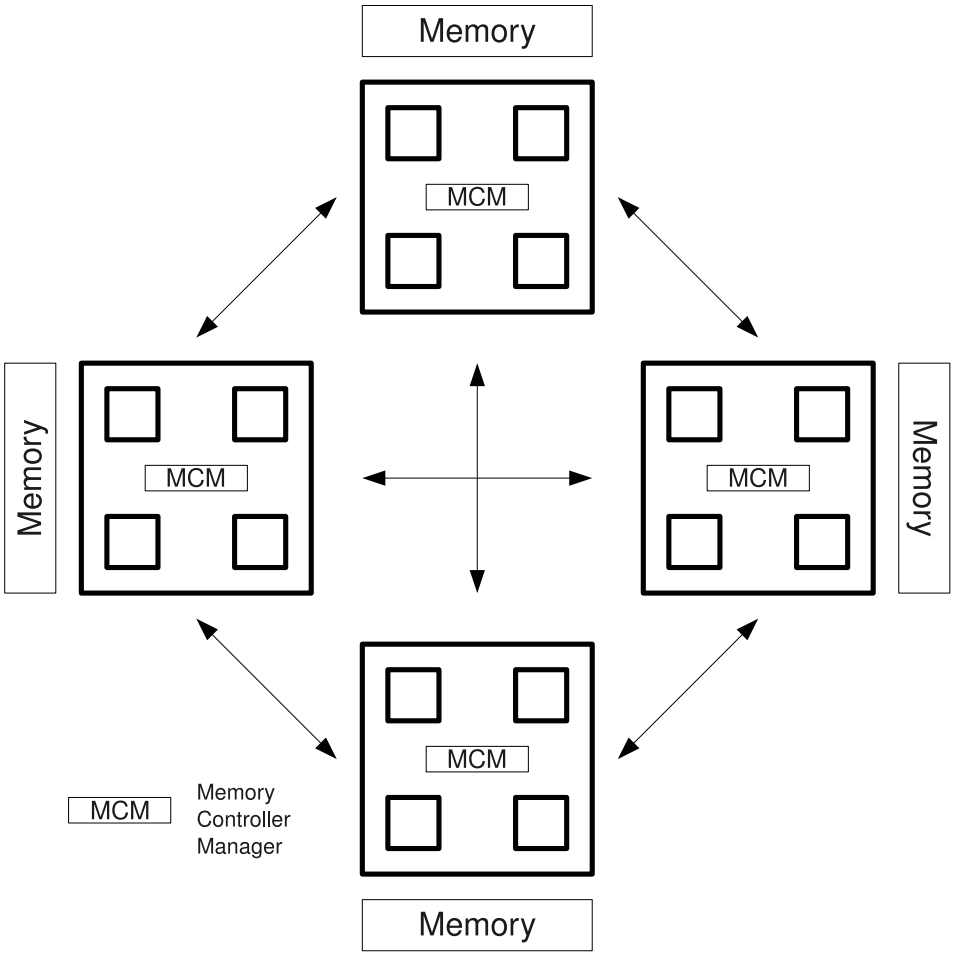
Plan

TOP500 (Novembre 2009)



- Augmentation du nombre de cœurs par nœud de calcul
- Augmentation de l'hétérogénéité des architectures (CPU, GPU, SPU, ...)
- Annonce par Intel d'un nouveau processeur à 48 cœurs

Exemple d'architecture SMP



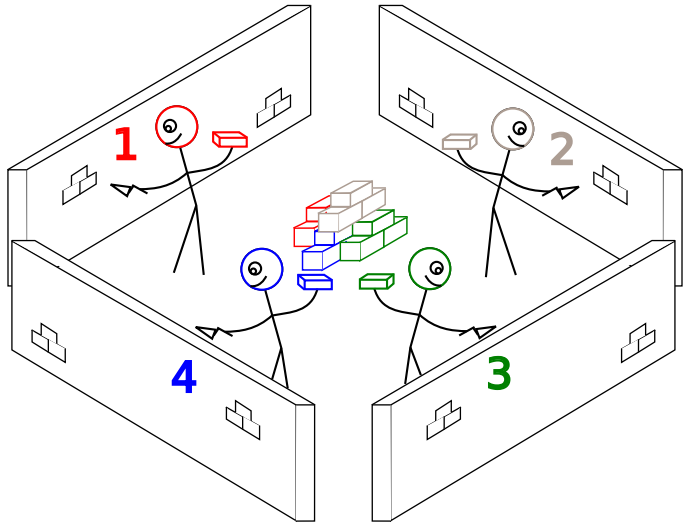
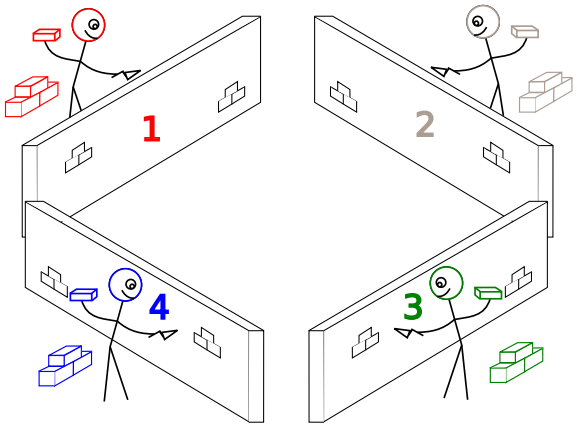


FIG.: Decrypthon, DRIMM Bordeaux 1

Exemples d'architectures NUMA



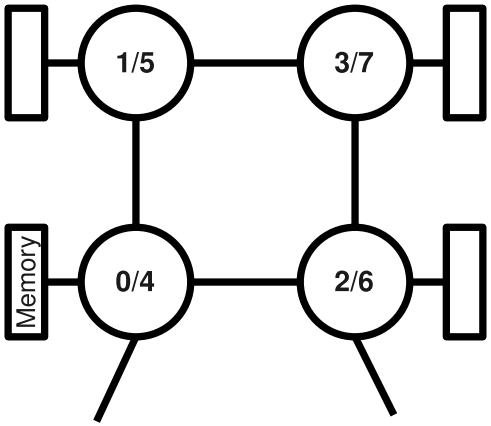


FIG.: Borderline, Grid'5000

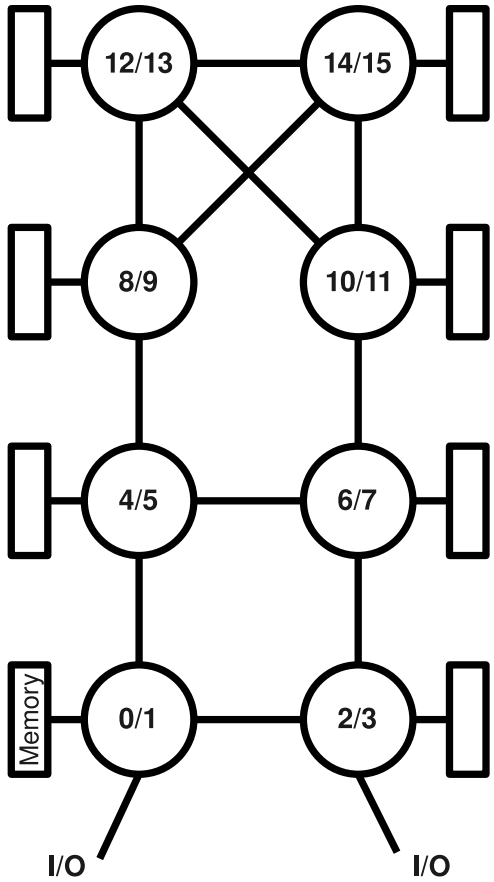


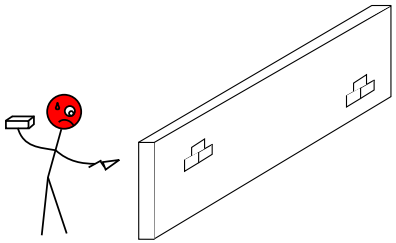
FIG.: Hagrid, CREMI Bordeaux 1

Architectures utilisées

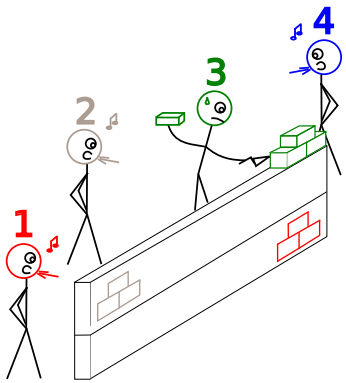
TAB.: Caractéristiques des clusters utilisés pour l'ensemble des évaluations.

	NUMA			SMP	
	Hagrid	Borderline	Titane	Decrypthon	Vargas
Type	AMD Opteron	AMD Opteron	Intel Nehalem	IBM Power5	IBM Power6
Nb. processeurs	8	4	2	8	16
Nb. coeurs par processeur	2	2	4	2	2
Nb. coeurs par noeud	16	8	8	16	32
Memoire par coeur en Go	4	4	3	1,75	3,2
Memoire par noeud	64	32	24	28	102,4
Nb. noeuds	1	10	1068	12	112
Réseau	-	Infiniband / Myrinet	Infiniband	IBM Federation	Infiniband
MPI	-	Mvapich2	Intel MPI	IBM MPI	IBM MPI

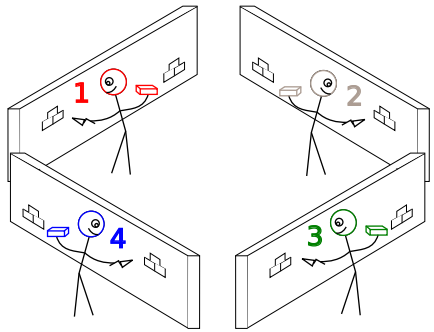
Comment exploiter ces architectures ?



Séquentiel



Parallélisme inefficace



Parallélisme efficace

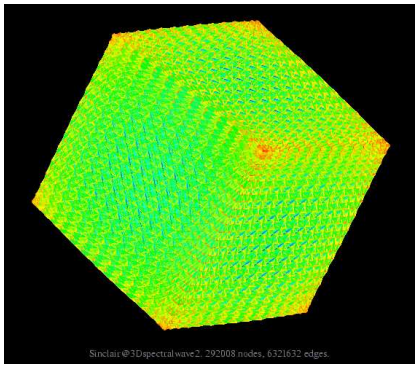
1. Bibliothèques parallèles :
   - Facile à mettre en place
   - Faible partie du code parallélisée
2. Directives de compilation : (HPF, OpenMP, ...)
   - Parallélisation facile des boucles de calcul
   - Ne prend pas en compte les architectures NUMA
3. Parallélisation intrusive : (MPI, Cilk, Thread, ...)
   - Permet une parallélisation du code au *tournevis*
   - Programmation très intrusive et difficile à mettre en œuvre

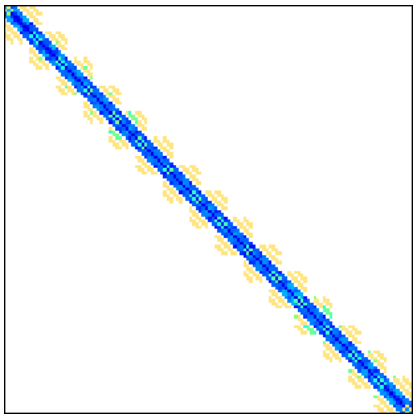
Plan

Le calcul HPC

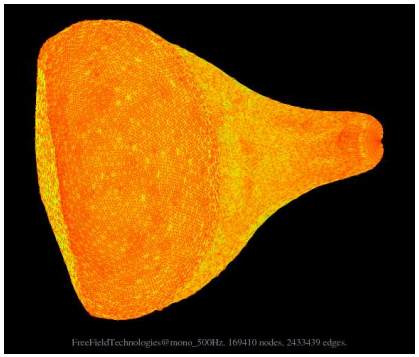
- Modélisation des phénomènes physiques pour prédire ou comprendre leurs comportements : électronique, météorologie, géophysique, ...
  $\Longrightarrow$ Résolution de systèmes linéaires $Ax = b$ avec plusieurs (dizaines de) millions d'inconnues
- La plupart de ces systèmes sont dits *creux*
- Très coûteux en temps de calcul et en espace mémoire
  $\Longrightarrow$ Nécessité de paralléliser les bibliothèques de résolution et de les adapter aux architectures modernes

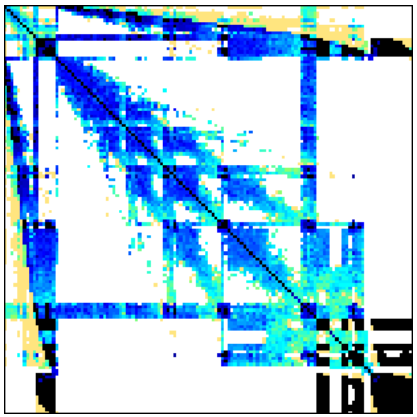
Exemples de problèmes (1/2)





(a) 3DSpectralWave2





(b) Mono_500Hz

Exemples de problèmes (2/2)

TAB.: Matrices utilisées pour l'ensemble des évaluations.

Nom	N	NNZ_A	NNZ_L	OPC	T	S	Source
Matr5	485 597	12 359 369	680 915 459	9.84e+12	D	U	UMN
Matr6	470 596	12 127 402	637 756 616	9.10e+12	D	U	UMN
Audi	943 695	39 297 771	1 141 513 029	5.21e+12	D	S	PARASOL
Nice20	715 923	28 066 527	1 050 576 453	5.19e+12	D	S	BRGM
Inline	503 712	18 660 027	158 830 261	1.41e+11	D	S	PARASOL
Nice25	140 662	2 914 634	51 133 109	5.26e+10	D	S	BRGM
MchInf	49 800	4 136 484	22 878 995	4.79e+10	D	U	UMN
Thread	29 736	2 249 892	25 370 568	4.45e+10	D	S	PARASOL
3DSpectralWave	680 943	17 165 766	1 340 207 093	1.00e+13	Z	H	UFL
3DSpectralWave2	292 008	7 307 376	394 131 174	1.67e+12	Z	H	UFL
Haltere	1 288 825	10 476 775	404 977 313	7.55e+11	Z	S	CEA-Cesta
FemHifreqCircuit	491 100	10 365 178	178 227 119	4.75e+11	Z	U	UFL
Mono_500hz	169 410	2 602 849	77 043 060	2.61e+11	Z	U	UFL

Les méthodes de résolution

- Méthodes directes
  - Décomposition de la matrice en $A = LU$ ou $A = L(D)L^t$
  - Coûteuses en mémoire et en temps de calcul mais très précises
- Méthodes itératives
  - Gauss-Seidel, Jacobi, Gradient Conjugué, GMRES, ...
  - Moins coûteuses en mémoire et en calculs mais peuvent ne pas converger dans certains cas
  - Possibilité d'utiliser un préconditionneur pour améliorer la convergence
- Méthodes hybrides (HIPS, MaPHyS)
  - Décomposition de domaine
  - Méthode directe à l'intérieur des sous-domaines et itérative sur les interfaces
- Méthodes multi-grilles
  - Numériquement scalables
  - Restriction et prolongation souvent dépendantes du problème modélisé

Les différents solveurs directs (1/2)

- MUMPS, `http://graal.ens-lyon.fr/MUMPS/`
  - Code en Fortran / MPI
  - Out-of-core, renumérotation non-symétrique
  - Développement d'une version OpenMP en projet
- PARDISO, `http://www.pardiso-project.org/`
  - Fortran / OpenMP, pas de version MPI (Fortran)
  - Renumérotation non-symétrique
- SUPERLU, SUPERLU_DIST et SUPERLU_MT, `http://crd.lbl.gov/~xiaoye/SuperLU/`
  - 3 versions distinctes en C
  - Séquentiel, MPI et threads ou OpenMP
  - Pas de couplage entre SUPERLU_DIST et SUPERLU_MT

Les différents solveurs (2/2)

- WSMP, `http://www.alphaworks.ibm.com/tech/wsmp/`
  - Une version mémoire partagée (C / Thread)
  - Une version mémoire distribuée (C / MPI)
  - Possibilité de couplage
- PaStiX, `http://pastix.gforge.inria.fr/`
  - Code hybride en C MPI / Thread
  - Choix de la version threads et/ou MPI à la compilation

PaStiX

- EPI Bacchus (INRIA Bordeaux - Sud-Ouest)
- `http://pastix.gforge.inria.fr/`
- Méthode supernodale
- Implémentation hybride :
  - MPI en mémoire distribuée
  - Posix thread en mémoire partagée
- Distribution et ordonnancement statique (pré-calculé à l'avance)

Les méthodes directes (PaStiX)

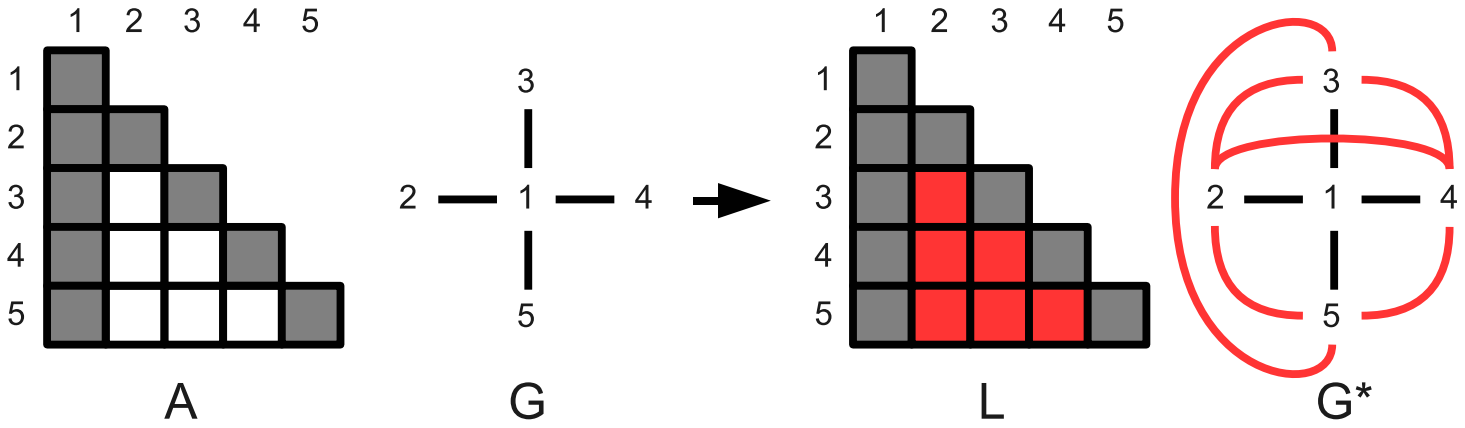
Les étapes

1. Analyse :
   - Renumérotation
   - Factorisation symbolique
   - Distribution des données
2. Factorisation
3. Résolution

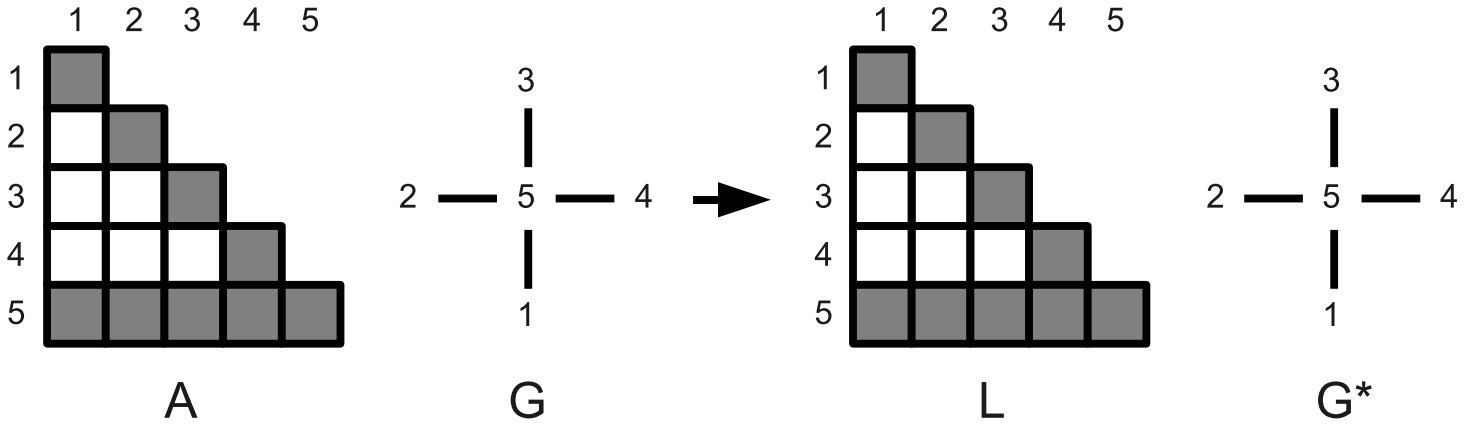
Les objectifs de l'analyse

- Minimiser le remplissage de la matrice
- Maximiser le parallélisme

La renumérotation

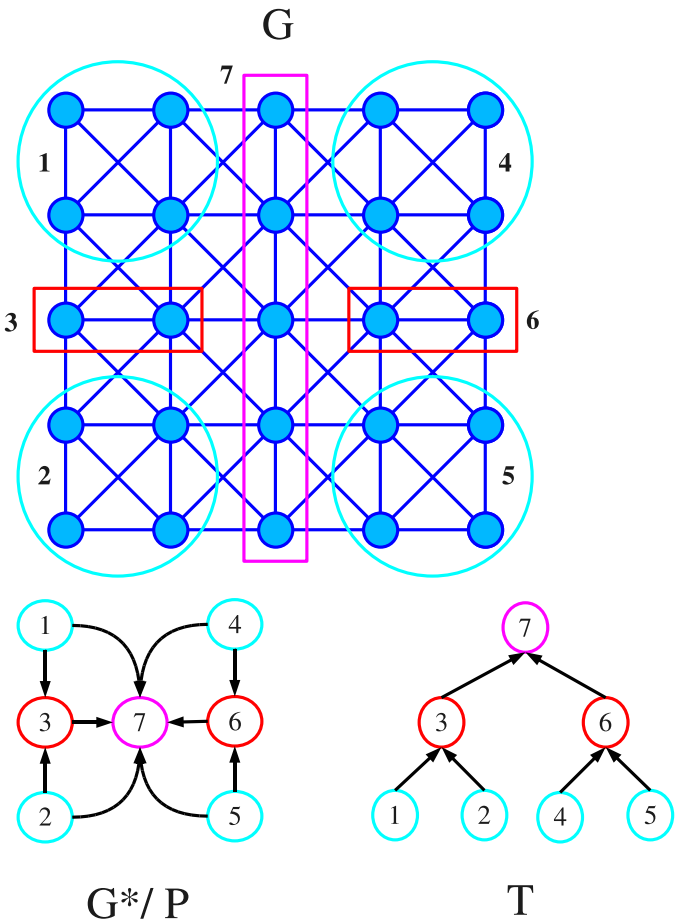


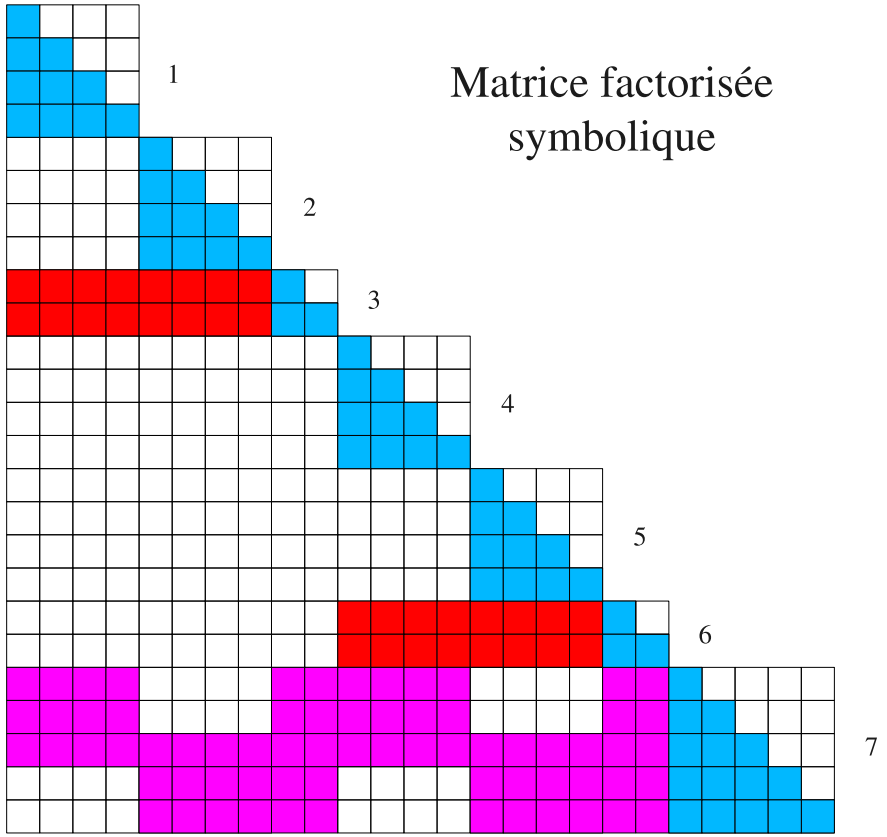
(a) Sans renumérotation



(b) Avec renumérotation

Factorisation symbolique





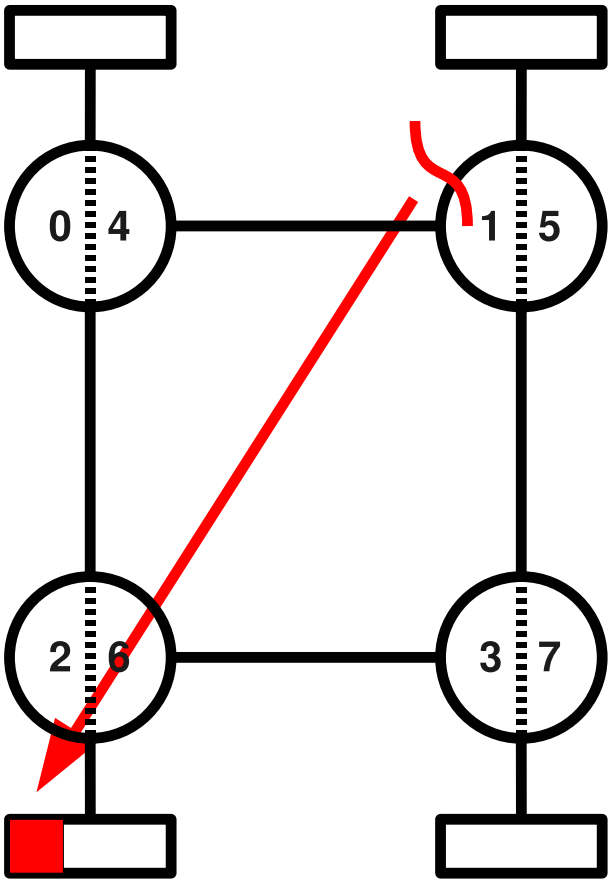
Matrice factorisée symbolique

Objectifs

- Les architectures modernes ont une topologie hiérarchique impliquant des accès mémoire non uniformes (NUMA)
  $\Longrightarrow$ Structures de données adaptées
- Les accès mémoire hétérogènes sont difficiles à intégrer dans les modèles de coût de l'ordonnancement statique
  $\Longrightarrow$ Ordonnancement dynamique pour corriger les défauts
- La distribution par défaut n'est plus efficace sur les grands problèmes
  $\Longrightarrow$ Calculs basés sur un grain plus fin

Plan

Placement des données



- Données allouées proche du cœur i
- Calculs effectués sur le cœur j
- 1000 appels aux routines BLAS (dAXPY, dGEMV, dGEMM)
- 1000 ensembles de 3 matrices 128×128

Résultats sur Borderline avec la routine dAXPY

Placement des données	Placement du thread de calcul 0	1	2	3	4	5	6	7
0	**1.00**	1.34	1.31	*1.57*	**1.00**	1.34	1.31	*1.57*
1	1.33	**1.00**	*1.57*	1.31	1.33	**1.00**	*1.57*	1.31
2	1.28	*1.57*	**1.00**	1.33	1.29	*1.57*	**1.00**	1.32
3	*1.56*	1.32	1.34	**1.00**	*1.56*	1.31	1.33	**0.99**
4	**1.00**	1.34	1.31	*1.57*	**1.00**	1.34	1.30	*1.58*
5	1.33	**1.00**	*1.58*	1.30	1.33	**1.00**	*1.57*	1.30
6	1.29	*1.57*	**1.00**	1.33	1.29	*1.57*	**1.00**	1.32
7	*1.57*	1.32	1.33	**1.00**	*1.57*	1.32	1.35	**1.00**

Les résultats sont normalisés par rapport au temps de calcul du meilleur cas ($i = j$)

Maximum : $1,58$

Résultats sur Borderline avec la routine dGEMM

Placement des données	Placement du thread de calcul 0	1	2	3	4	5	6	7
0	**1.00**	1.04	1.04	*1.07*	**1.00**	1.04	1.04	*1.07*
1	1.04	**1.00**	*1.07*	1.04	1.04	**1.00**	*1.08*	1.04
2	1.04	*1.07*	**1.00**	1.04	1.04	*1.07*	**1.00**	1.04
3	*1.08*	1.04	1.04	**1.00**	*1.07*	1.04	1.05	**1.00**
4	**1.00**	1.04	1.04	*1.07*	**1.00**	1.04	1.04	*1.07*
5	1.05	**1.00**	*1.08*	1.04	1.05	**1.00**	*1.08*	1.04
6	1.04	*1.07*	**1.00**	1.05	1.04	*1.07*	**1.00**	1.04
7	*1.08*	1.04	1.05	**1.00**	*1.08*	1.04	1.05	**1.00**

Les résultats sont normalisés par rapport au temps de calcul du meilleur cas ($i = j$)

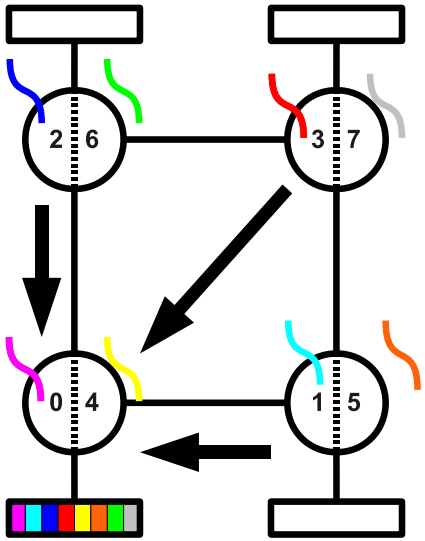
Maximum : $1,08$

Résumé des expériences de placement NUMA

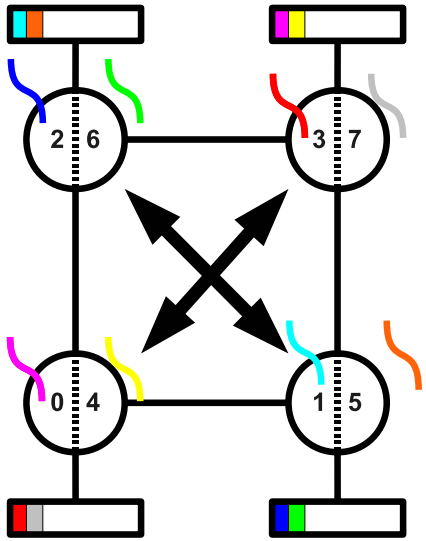
Plate-forme	dAXPY	dGEMV	dGEMM
Borderline	1,58	1,50	1,08
Hagrid	2,08	1,78	1,26
Decrypthon	1,07	1,02	1,01

TAB.: Facteurs NUMA maximaux obtenus

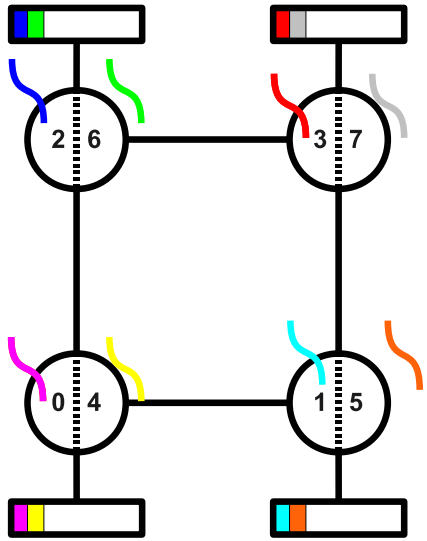
Problèmes de contention



(a) Sur un cœur



(b) Placements entrelacés



(c) Placements locaux

- Nécessité de prendre en compte les problèmes de contention
- Même expérience que précédemment avec autant de threads que de cœurs sur la plate-forme

Résultats sur Hagrid

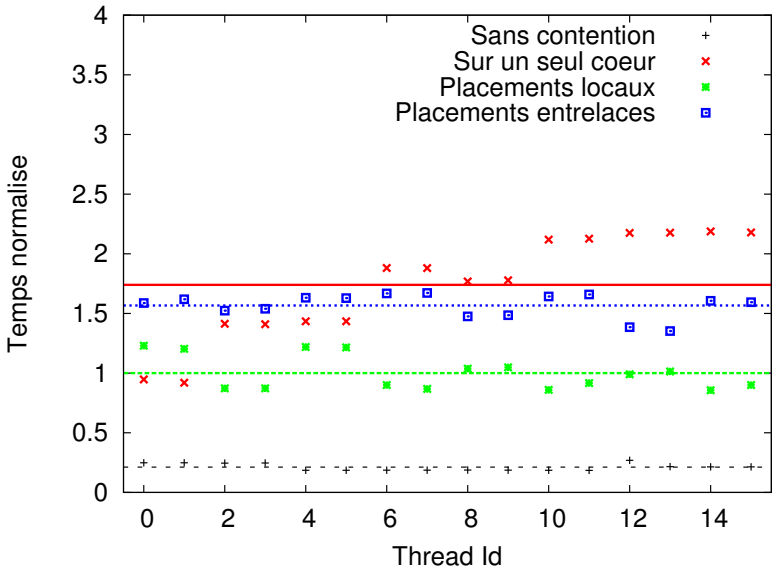


FIG.: Routine dAXPY

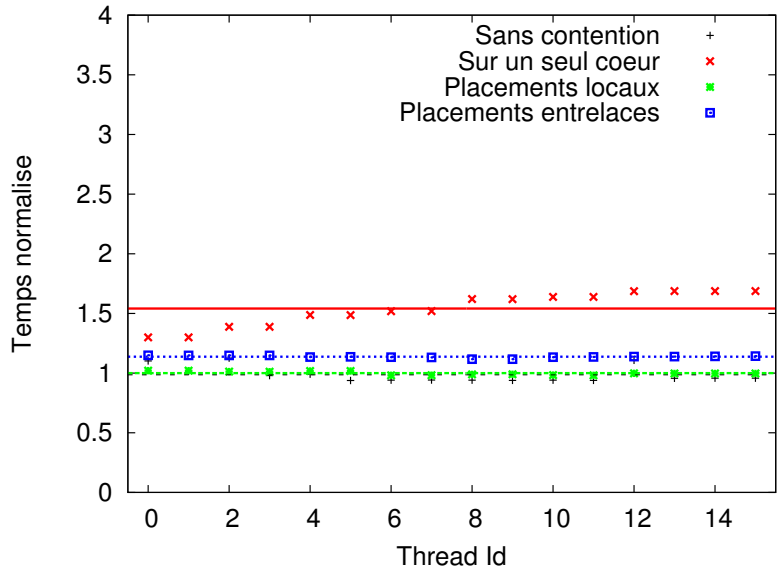


FIG.: Routine dGEMM

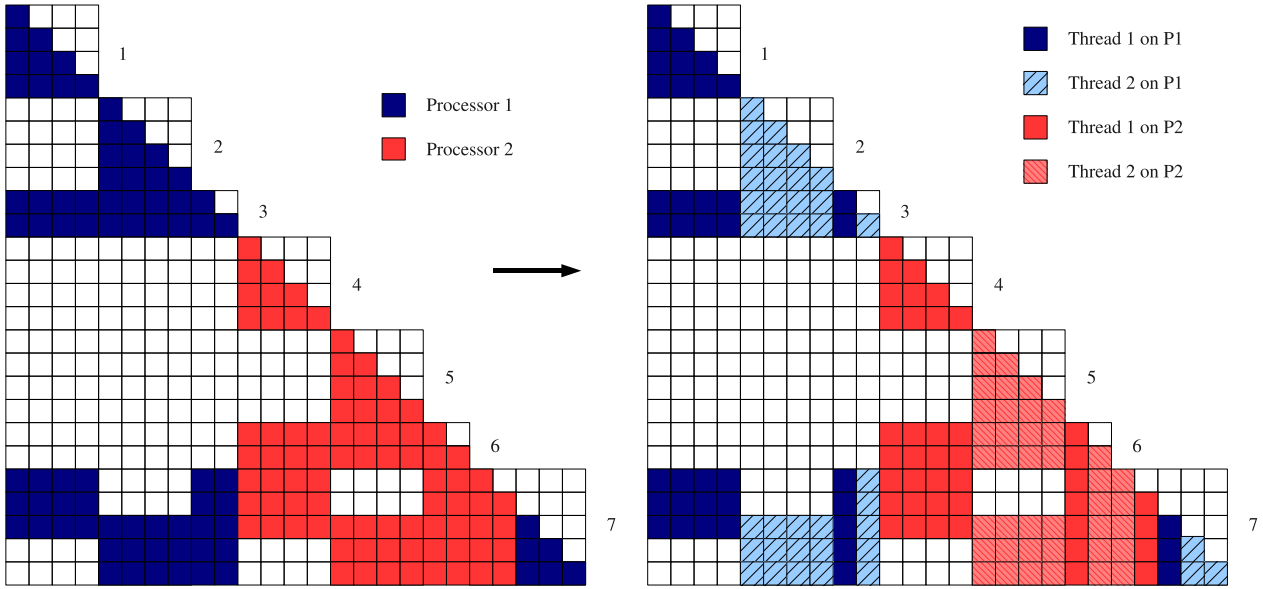
Maxima : $2,2$ et $1,7$

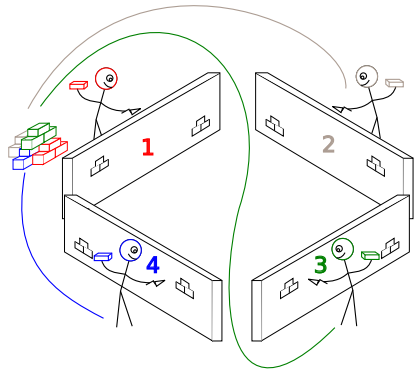
Résumé des expériences de contention

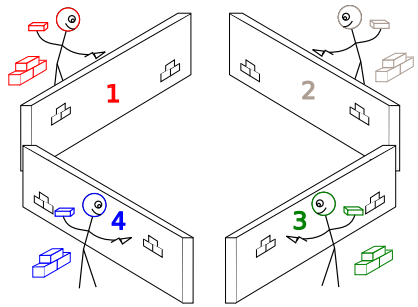
Plate-forme	dAXPY	dGEMM
Borderline	3,1	1,09
Hagrid	2,18	1,62
Decrypthon	1,05	1,02

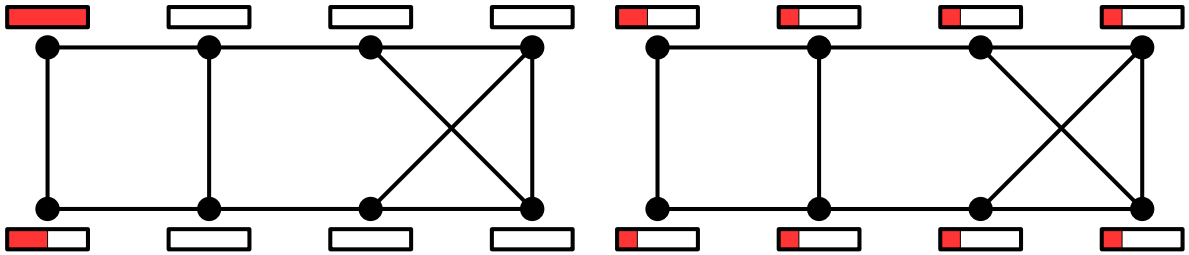
TAB.: Facteurs NUMA maximaux obtenus

Allocation pour architecture NUMA









(a) Allocation initiale (b) Allocation “NUMA”

Évaluation

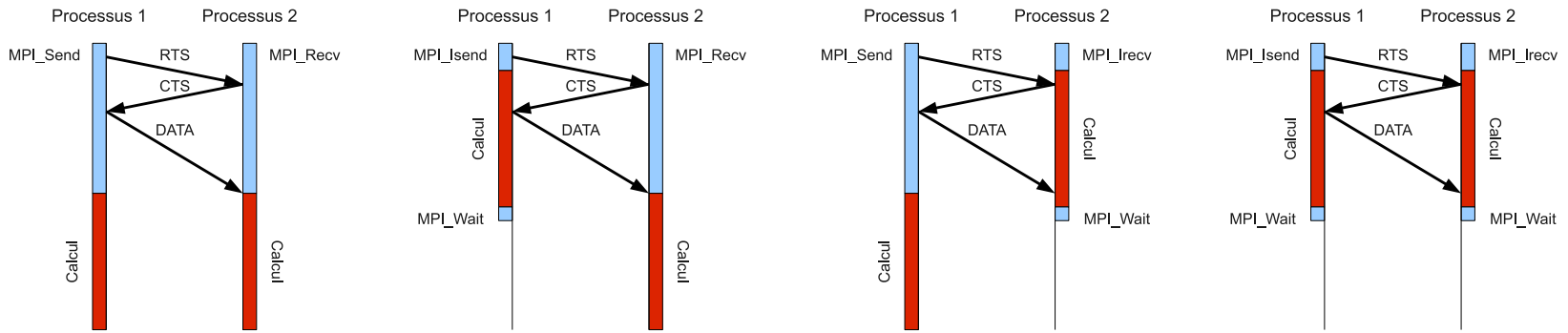
Matrice	Borderline		Hagrid		Decrypthon	
	V0	N	V0	N	V0	N
Matr5	469	**422**	481	**386**	162	**161**
Matr6	426	**405**	450	**361**	**177**	178
Audi	274	**244**	255	**230**	101	**99,9**
Nice20	248	**230**	248	**188**	91,8	**90,7**
Inline	9,07	**7,95**	18,4	**17,3**	5,88	**5,62**
Nice25	3,28	**2,62**	7,23	**5,08**	2,07	**1,9**
MchInf	3,13	**2,43**	5,86	**3,24**	1,92	**1,89**
Thread	2,49	**2,17**	3,84	**2,26**	1,15	**1,12**
3DSpectralWave	2060	**1650**	1290	**1040**	**575**	603
3DSpectralWave2	311	**287**	267	**174**	**107**	**107**
Haltere	140	**140**	124	**92.1**	48.2	**47,7**
Fem_Hifreq_Circuit	**101**	106	82	**68**	32,9	**32,3**
Mono_500Hz	51,8	**50,1**	41,6	**37,1**	18,8	**18,7**

TAB.: Influence du placement des données sur le temps de factorisation du solveur PASTIX.

Gains de 20 à 40% sur NUMA

Plan

Recouvrement calculs / communications (1/2)

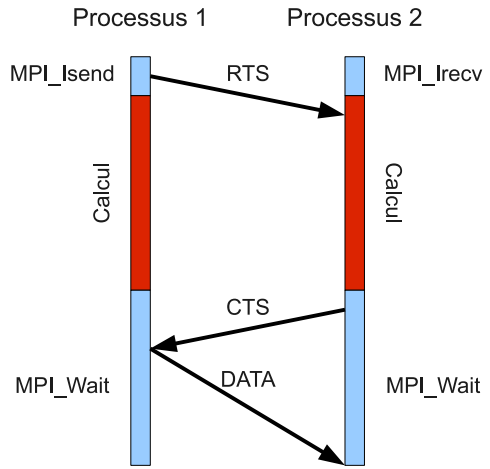


`MPI_Send/Recv` `MPI_Isend/Recv` `MPI_Send/Irecv` `MPI_Isend/Irecv`

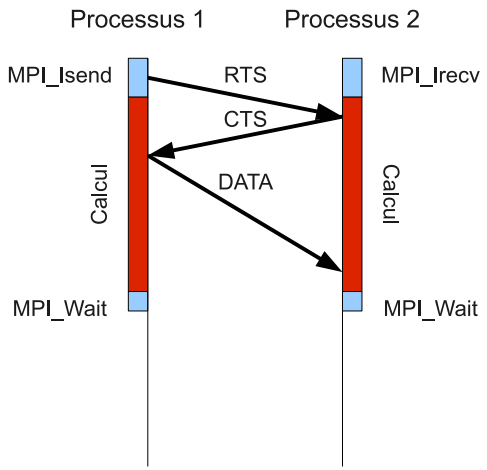
FIG.: Efficacité théorique des différentes combinaisons de communications bloquantes et non-bloquantes avec le protocole de *rendez-vous*.

- Différentes combinaisons possibles suivant les besoins
- Communications non bloquantes plus efficaces

Recouvrement calculs / communications (2/2)



(a) Sans progression des communications



(b) Avec progression des communications

FIG.: Comparaison du recouvrement calculs/communications possible avec ou sans une progression des communications efficace.

- Essayer de recouvrir les communications par des calculs
- Imposer par l'implémentation MPI utilisée

Les communications dans PaStiX

Déroulement de la boucle principale sur chaque thread :

1. Attendre toutes les contributions nécessaires au prochain calcul
2. Factoriser le bloc-colonne
3. Envoyer les contributions calculées

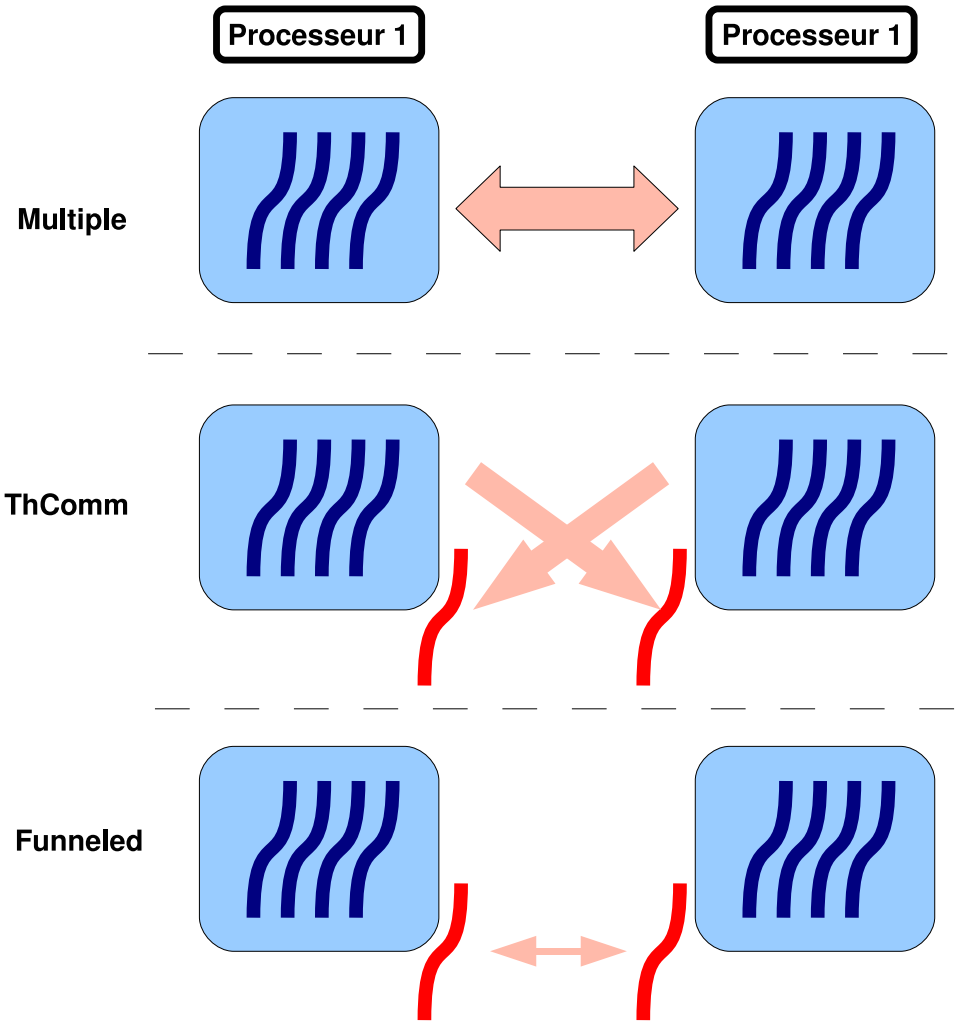
Problèmes

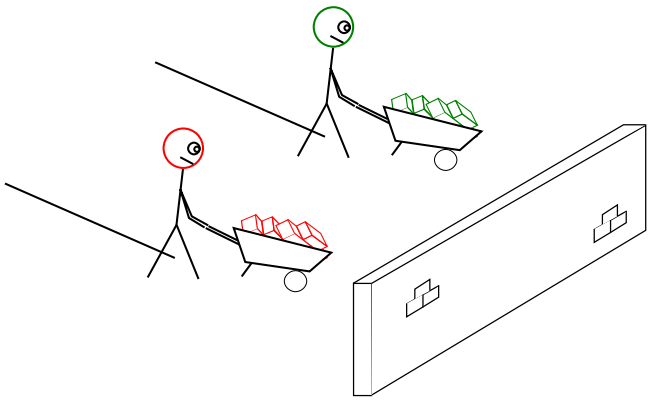
- Chaque thread attend les contributions qui sont destinées au bloc-colonne qu'il doit factoriser
- Tous les threads communiquent vers tous les autres threads

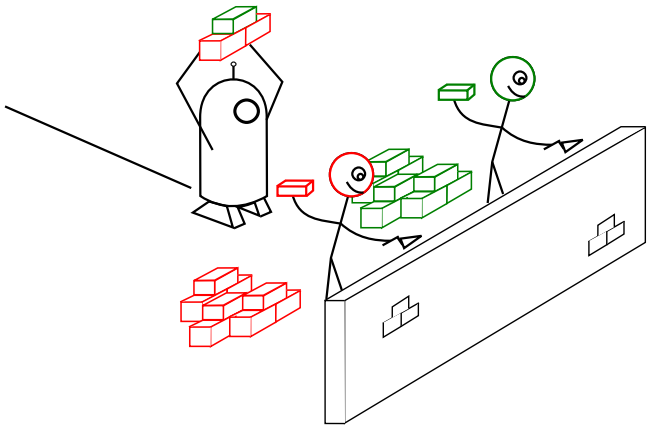
Niveaux de support des threads des bibliothèques MPI

- `MPI_THREAD_SINGLE`
  - L'application ne doit pas être multi-threadée
- `MPI_THREAD_FUNNELED`
  - Seul le thread principal peut effectuer des communications
- `MPI_THREAD_SERIALIZED`
  - Tous les threads peuvent communiquer vers tout le monde
  - À la charge du programmeur d'empêcher les communications simultanées et d'ordonnancer les communications pour éviter les blocages
- `MPI_THREAD_MULTIPLE`
  - Tous les threads peuvent communiquer vers tout le monde
  - La bibliothèque se charge de la protection des zones critiques

Les différents schémas de communication dans PaStiX







Évaluation : Decryphton (1/2)

- Réseau IBM Fédération

Nb. Threads			Matr6					3DSpectralWave				
			1	2	4	8	16	1	2	4	8	16
Nb. Processus	1	Séquentiel	2110	1100	553	292	178	8260	4260	2104	1150	573
	2	Multiple	1140	601	307	191	102	4400	2320	1290	724	367
		ThComm	**1090**	**552**	**284**	**161**	**101**	**4130**	2110	**1090**	**569**	**315**
		Funneled	1100	583	330	234	179	4140	**2090**	1130	621	446
	4	Multiple	672	368	195	108	61,4	2560	1300	654	352	190
		ThComm	**557**	**292**	**152**	**87,4**	**59,8**	2220	**1110**	**575**	**300**	**189**
		Funneled	593	346	279	236	226	**2090**	1180	684	576	481
	8	Multiple	483	213	130	68,9	56	1510	715	**359**	**195**	**131**
		ThComm	**353**	**179**	**101**	**64,4**	**44,5**	1270	**650**	474	197	144
		Funneled	389	313	291	288	272	**1160**	752	632	487	445

- Thread de progression permet des gains importants
- Version dégradée ne passe pas bien à l'échelle

Évaluation : Platine (2/2)

- Réseau Infiniband (IntelMPI)

Nb. Threads			Matr6				3DSpectralWave			
			1	2	4	8	1	2	4	8
Nb. Processus	1	Séquentiel	1040	532	295	144	4190	2130	1110	563
	2	Multiple	547	276	144	91,3	3290	1570	630	304
		ThComm	**518**	**275**	**143**	**75,3**	**2090**	**1060**	**560**	**285**
		Funneled	520	276	145	77,2	**2090**	**1060**	563	286
	4	Multiple	351	146	82,1	51,9	1510	667	385	179
		ThComm	**271**	**135**	**76**	**43**	**1030**	**539**	**284**	**151**
		Funneled	273	137	82,2	50,9	1040	540	288	157
	8	Multiple	217	97,8	49,5	29,4	935	371	200	101
		ThComm	**134**	**71**	**51,9**	**25,3**	**538**	**275**	**151**	**84,3**
		Funneled	137	80,8	54,5	35,1	540	282	159	94,8

- Thread de progression permet également des gains importants
- Version dégradée passe mieux à l'échelle

Plan

Distribution et ordonnancement

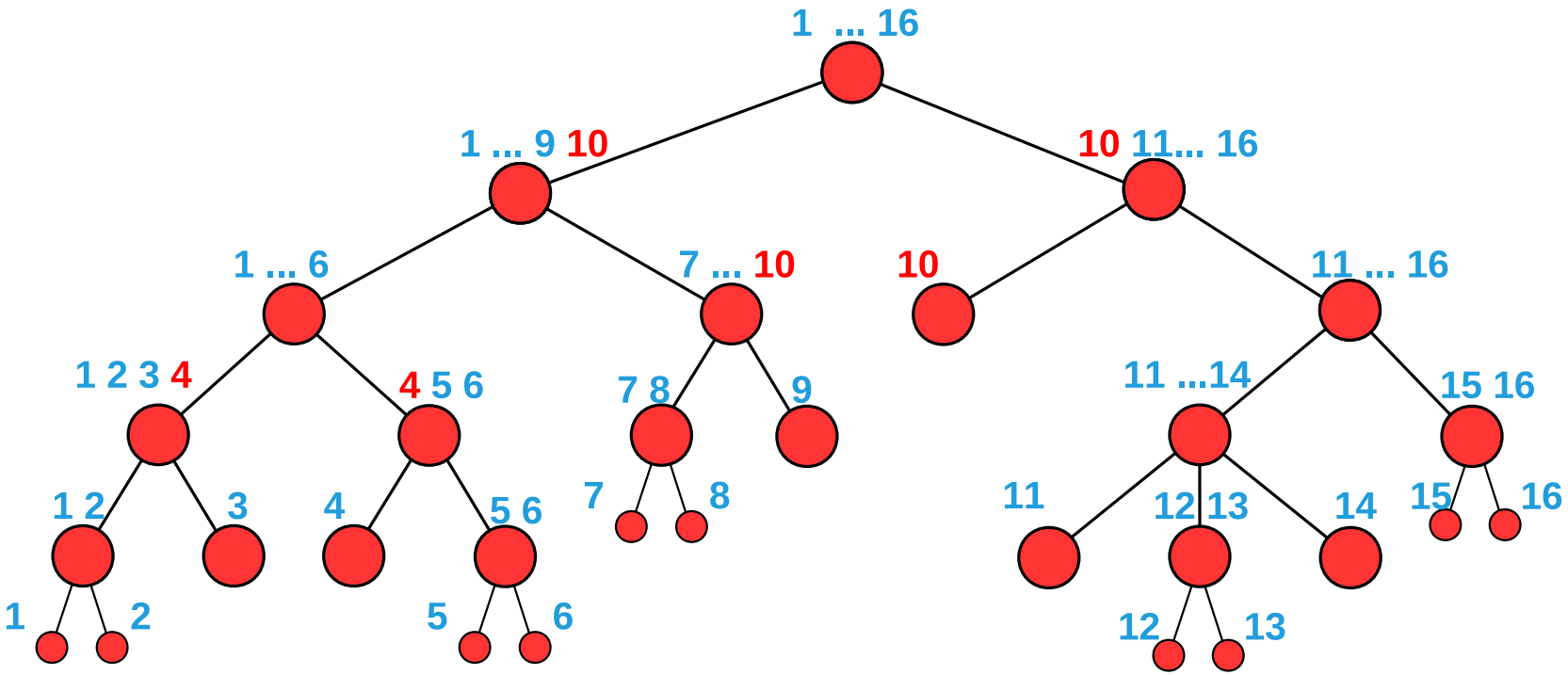
Existant

- Distribue statiquement les données
- Ne prend pas en compte la localité des données
- N'intègre pas les coûts d'accès hiérarchiques

Solution proposée

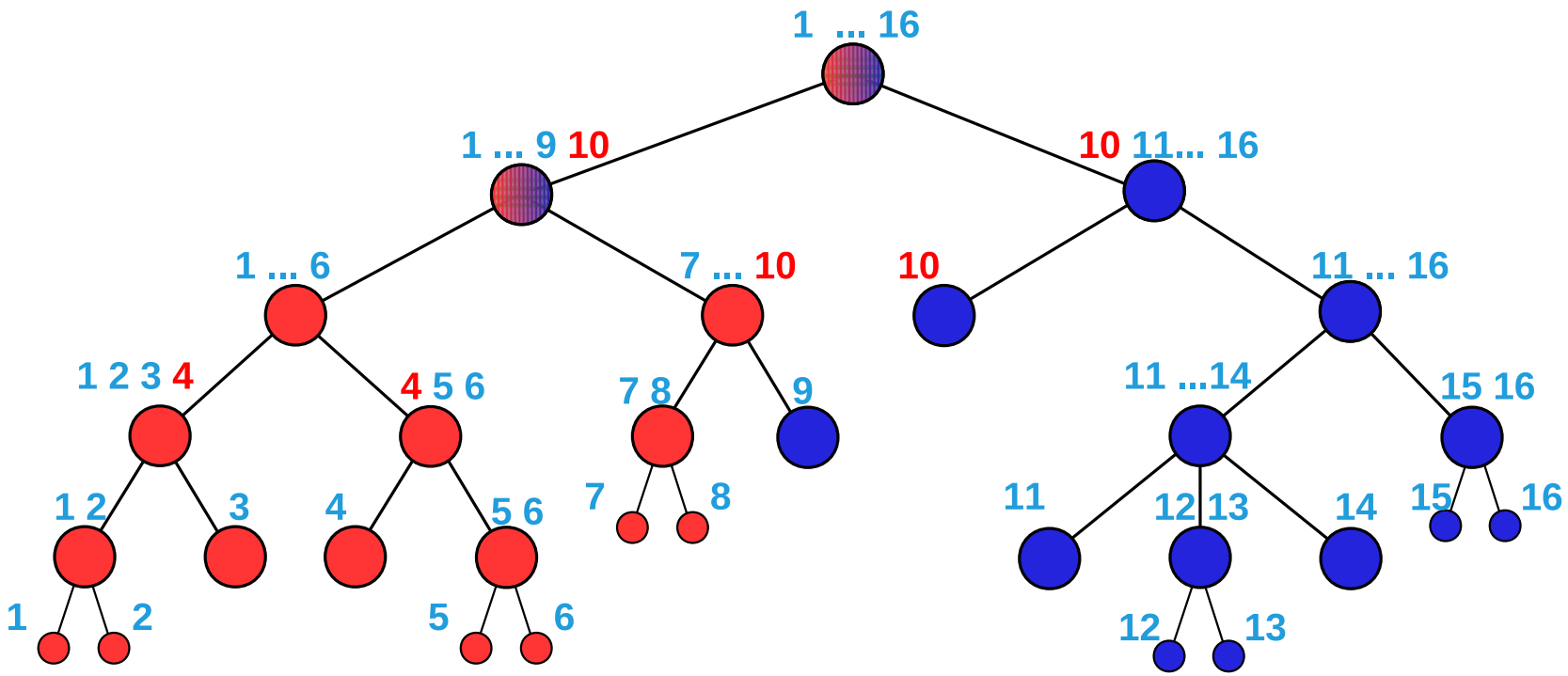
- Exploiter la distribution statique existante
- Re-distribuer les données au sein de chaque nœud
- Utiliser un thread dédié aux communications pour avoir une meilleure réactivité
- Faire du vol de travail "classique" mais limité pour conserver l'affinité mémoire

Répartition proportionnelle



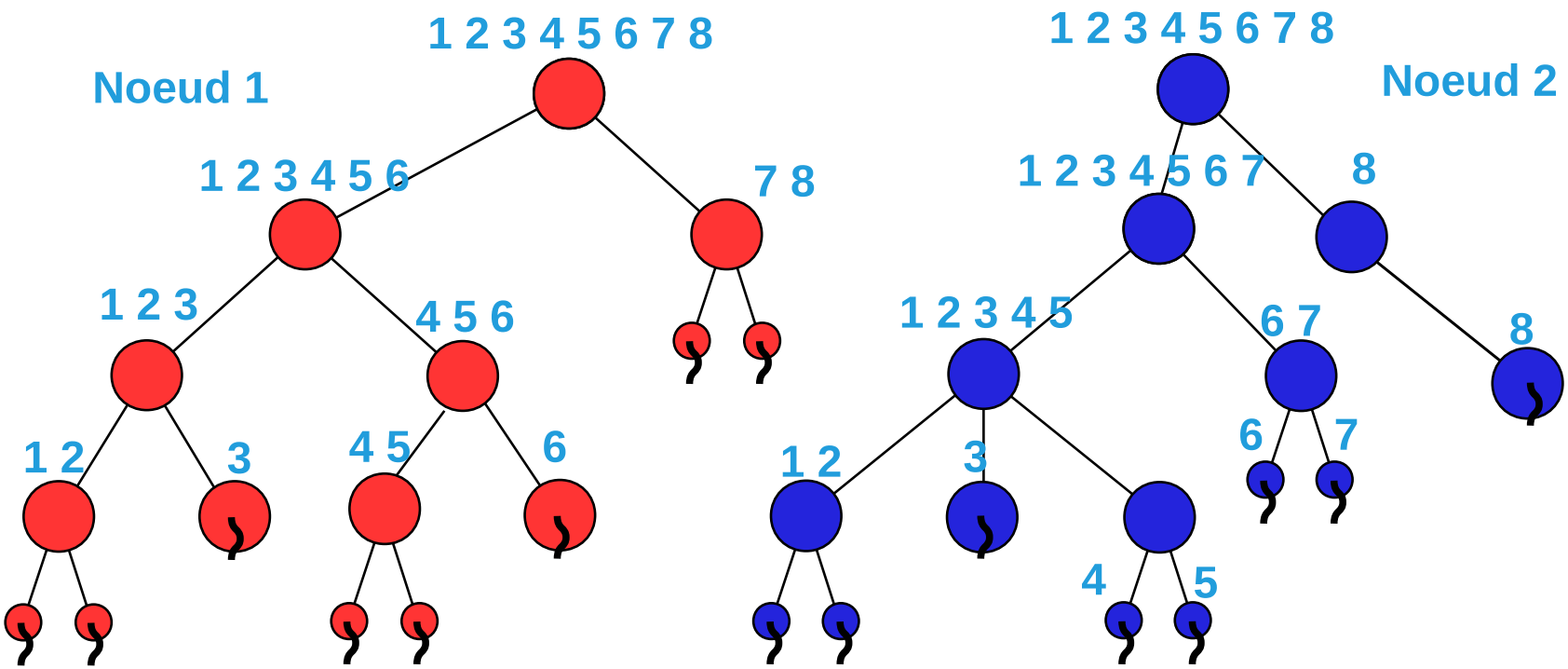
- Distribution des super-nœuds sur les processeurs
- Estimation du coût des sous-arbres à partir d'un modèle de coût des routines BLAS

Nouvelle distribution



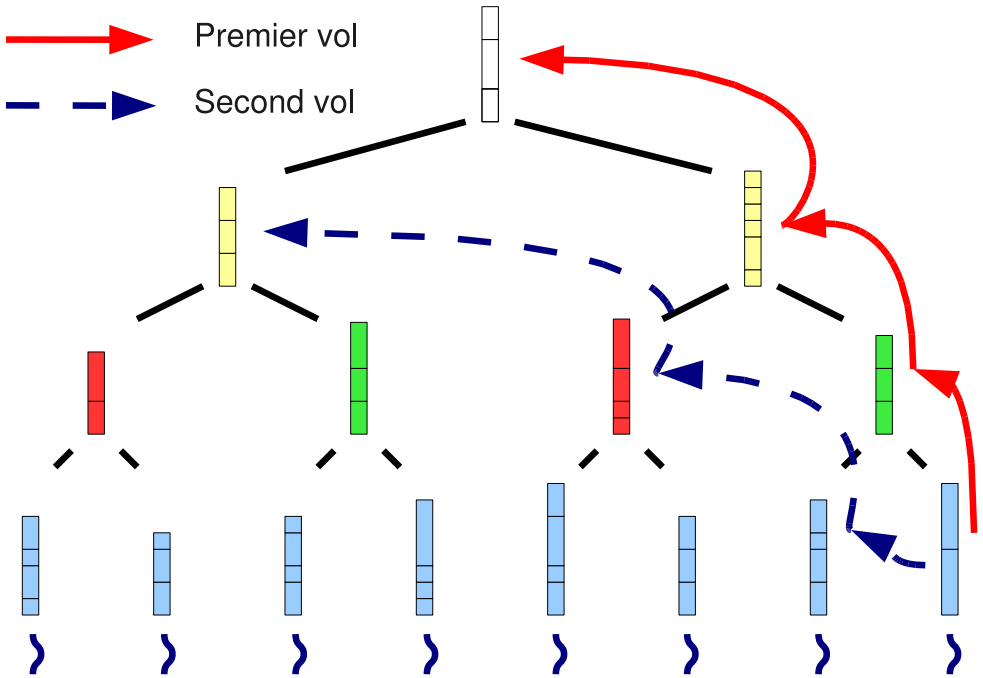
- Distribution des données sur les processus MPI
- Deux passes de l'algorithme :
  - **Répartition proportionnelle pour distribuer les données**
  - Construction d'une structure arborescente de files pour le vol de travail

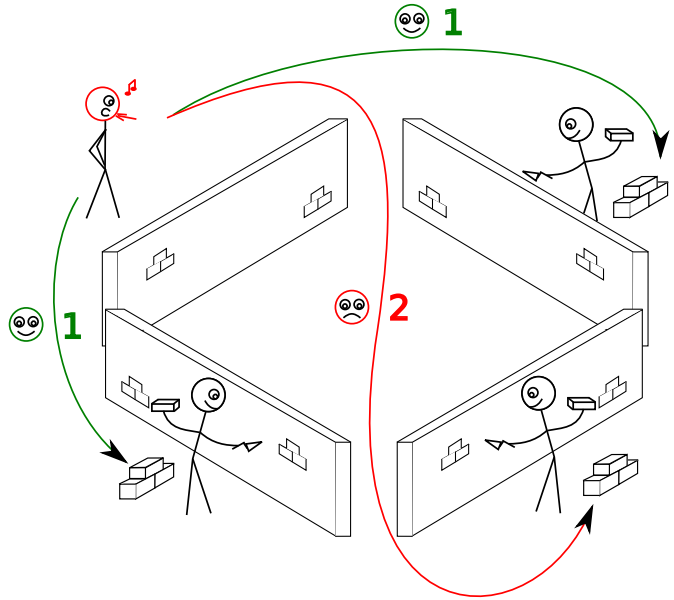
Nouvelle distribution



- Distribution des données sur les processus MPI
- Deux passes de l'algorithme :
  - Répartition proportionnelle pour distribuer les données
  - **Construction d'une structure arborescente de files pour le vol de travail**

Arbre de vol de travail





- Construction d'un arbre de files contenant les tâches
- Premier vol : le long du chemin vers la racine
- Second vol : à distance 1 du chemin vers la racine

Évaluation en mémoire partagée

Matrice	Borderline		Hagrid		Decrypthon	
	Stat.	Dyn.	Stat.	Dyn.	Stat.	Dyn.
Matr5	422	**390**	386	**352**	161	**151**
Matr6	405	**363**	361	**315**	178	**150**
Audi	244	**208**	230	**196**	99,9	**99,9**
Nice20	230	**211**	188	**173**	90,7	**89,3**
Inline	7,95	**7,41**	17,3	**14,7**	**5,62**	6,12
Nice25	**2,62**	2,81	**5,08**	5,22	1,9	**1,9**
MchInf	2,43	**2,42**	**3,24**	3,47	1,89	**1,72**
Thread	2,17	**2,04**	2,26	**2,18**	1,12	**1,02**
3DSpectralWave	1650	**1590**	1040	**960**	603	**544**
3DSpectralWave2	287	**279**	174	**168**	107	**100**
Haltere	140	**131**	**92.1**	94	47,7	**47,3**
Fem_Hifreq_Circuit	106	**96,2**	**68**	73,1	32,3	**31,6**
Mono_500Hz	50,1	**49,1**	37,1	**33,4**	18,7	**17,5**

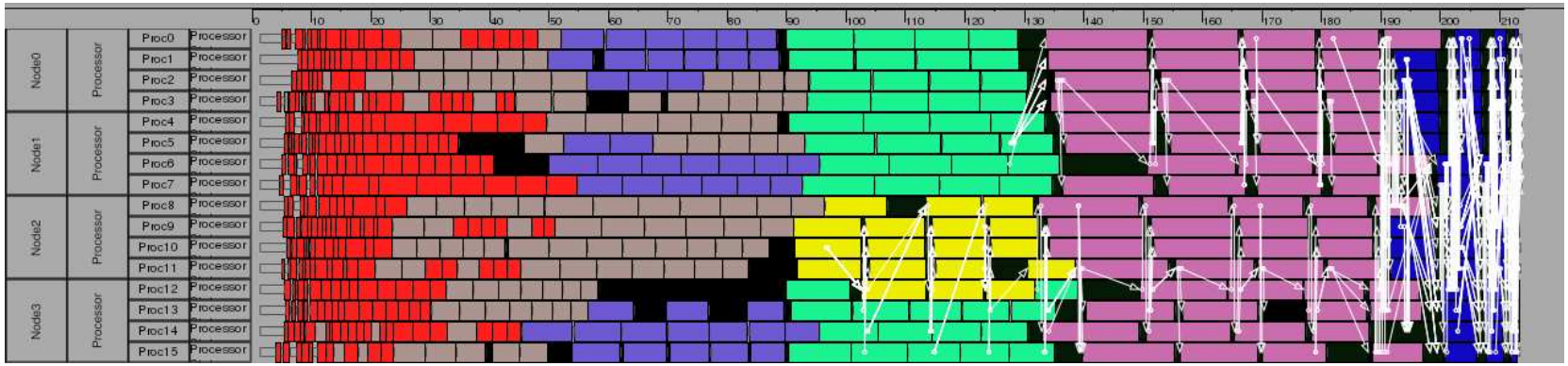
- Allocation pour architectures NUMA
- 1 thread de calcul par cœur disponible
- Gains de 15%

Évaluation en mémoire distribuée

Matrices	Nb. Nœuds	Titane Stat.	Titane Dyn.	Decrypthon Stat.	Decrypthon Dyn.
Matr5	1	159	**151**	161	**151**
	2	80,1	**78,7**	99,6	**90,9**
	4	56,7	**45,3**	65	**55,7**
	8	**27,8**	**27,8**	45,9	**40,9**
Matr6	1	144	**141**	178	**150**
	2	**75,3**	77,3	101	**92,3**
	4	43	**42,1**	59,8	**58,2**
	8	**25,3**	26,1	44,5	**43**
Audi	1	86,9	**82,6**	**100**	**100**
	2	**43,8**	45,9	67,3	**66,8**
	4	**24,3**	24,4	33,6	**32,9**
	8	**14,6**	15,9	23,2	**22,1**
3DSpectralWave	1	563	**545**	603	**544**
	2	285	**279**	315	**315**
	4	**151**	154	189	**173**
	8	**84,3**	85,7	144	**110**

- Allocation pour architectures NUMA
- 1 thread de progression par processus MPI / nœud
- 1 thread de calcul par cœur disponible
- Gains de 20%

Diagramme de Gantt : Matr5 (4x4)

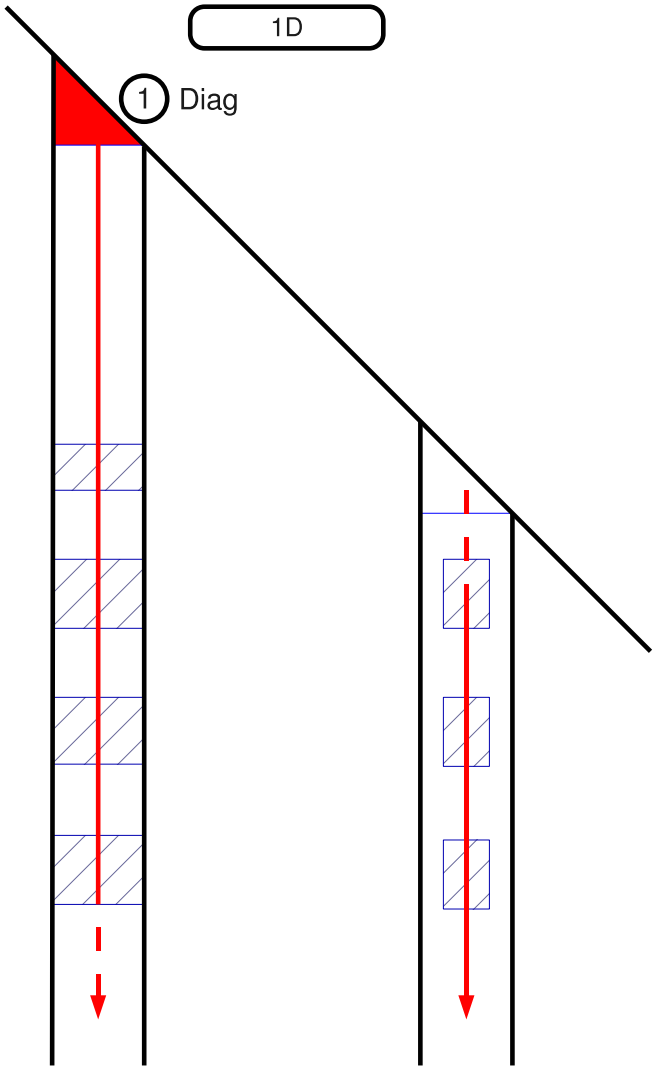


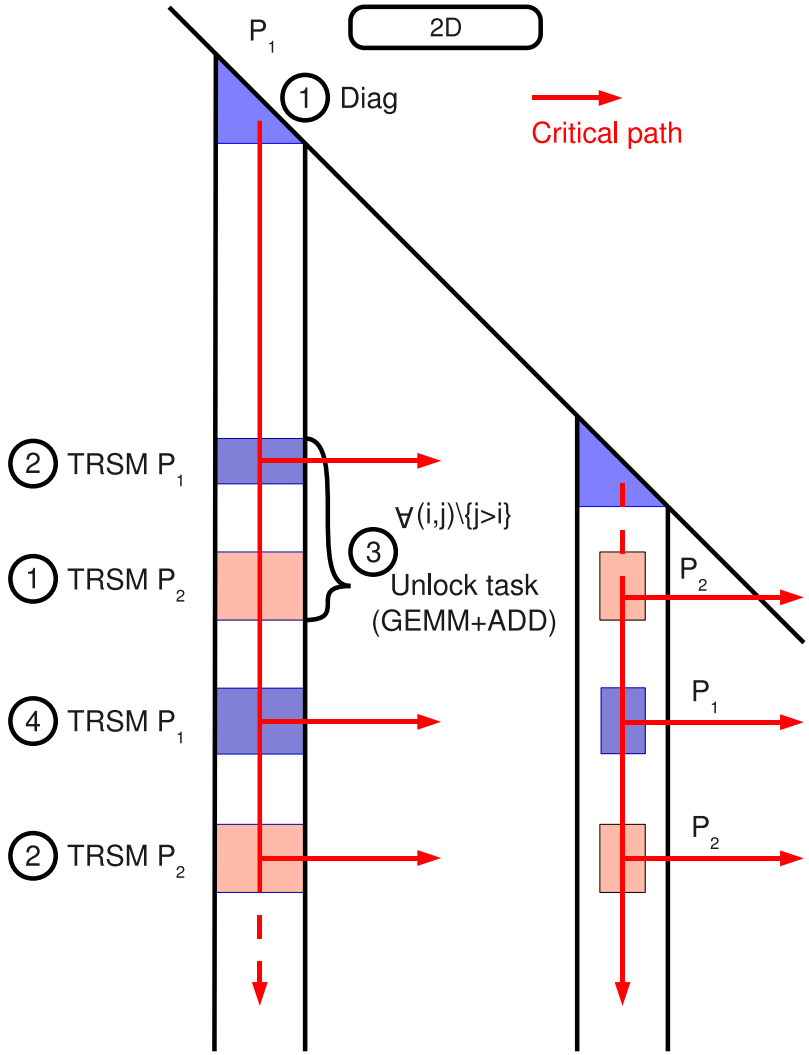
(a) Ordonnancement statique

(b) Ordonnancement dynamique

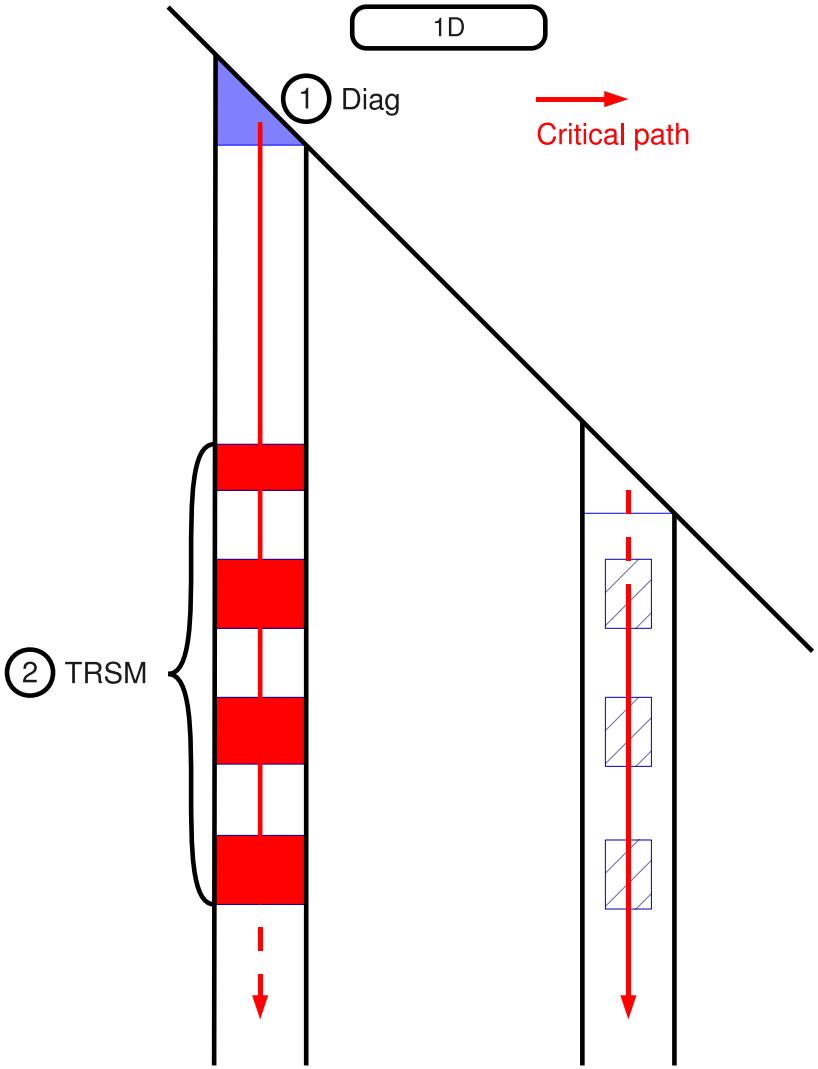
Plan

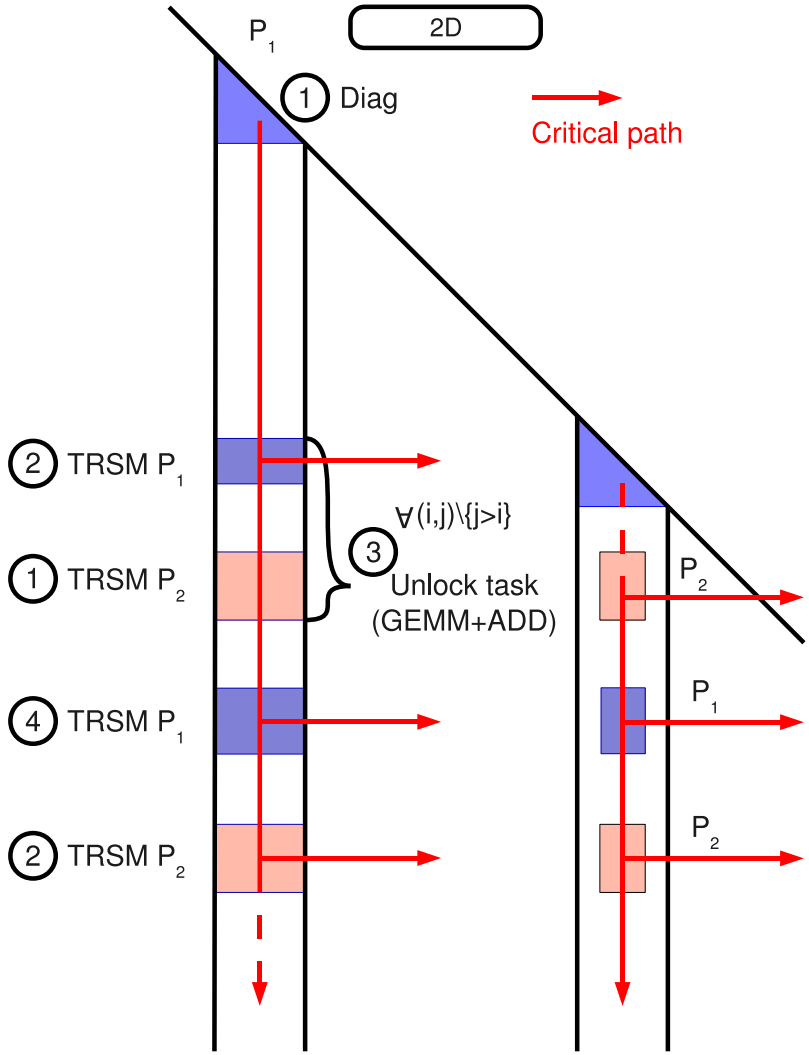
Distribution 1D et 2D



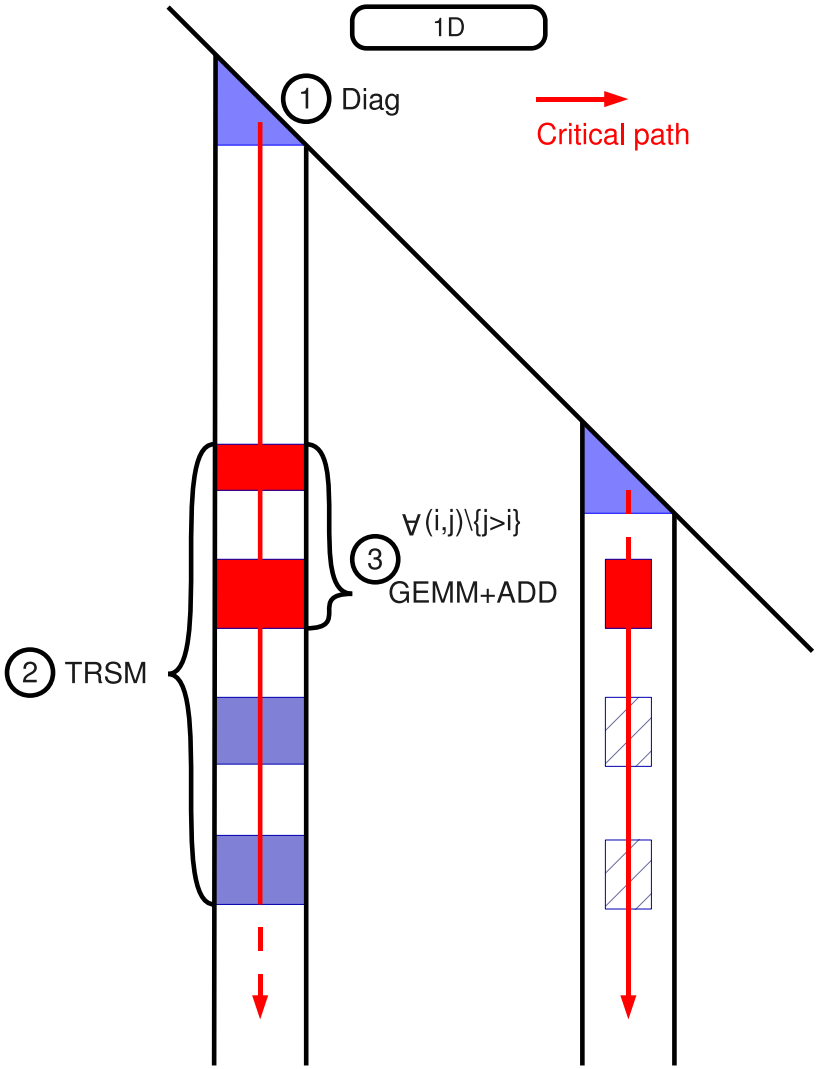


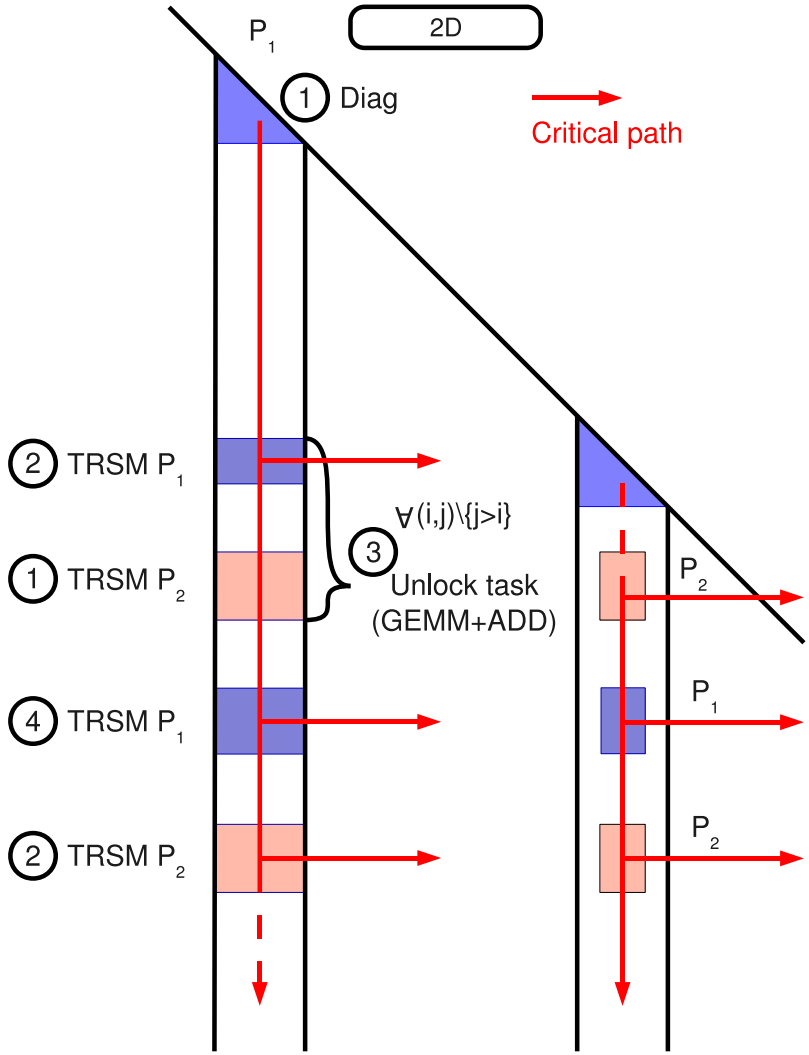
Distribution 1D et 2D



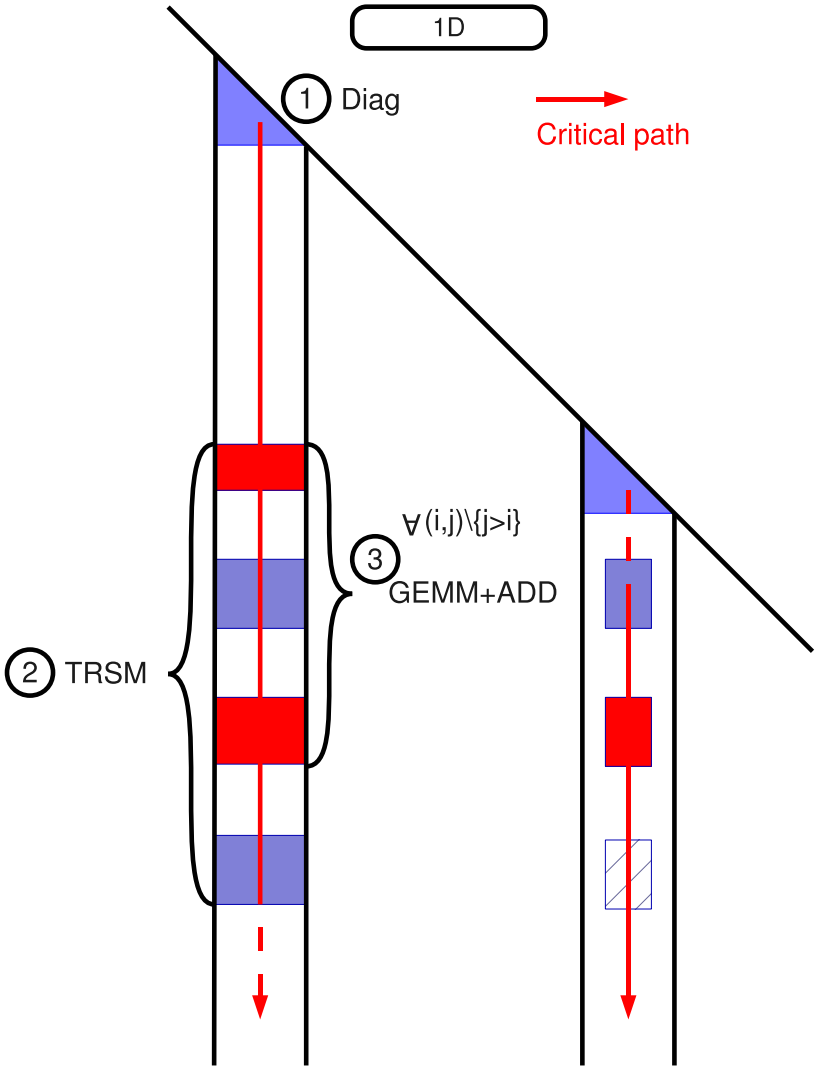


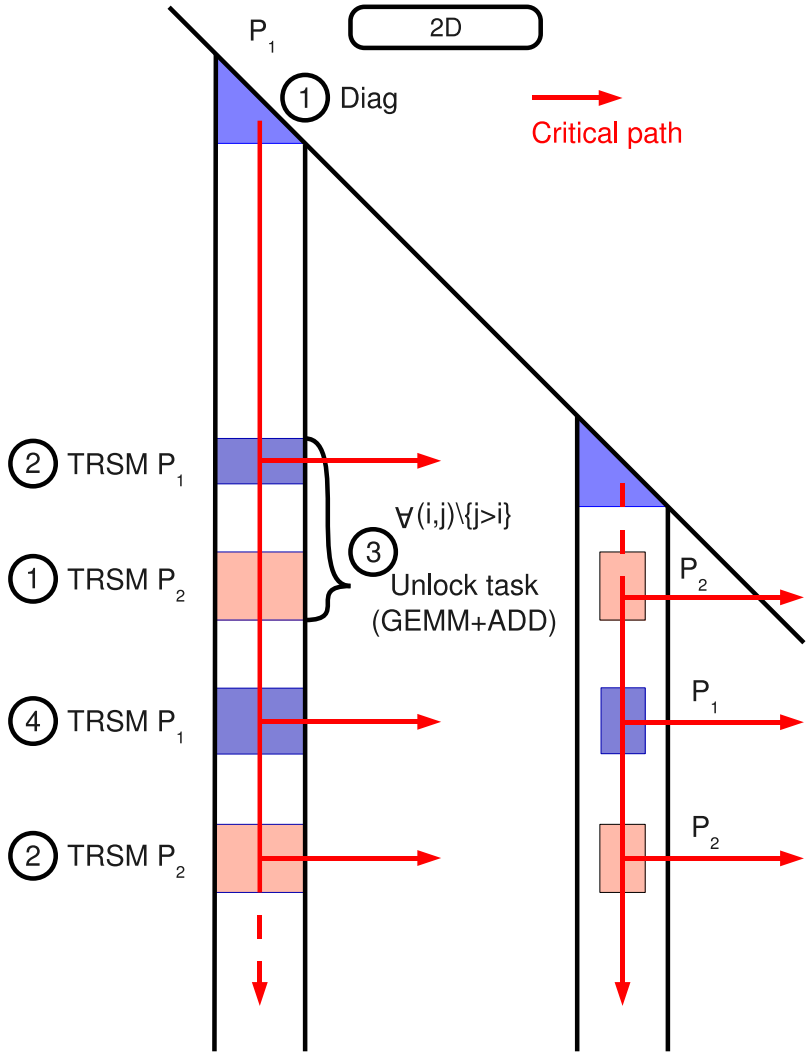
Distribution 1D et 2D



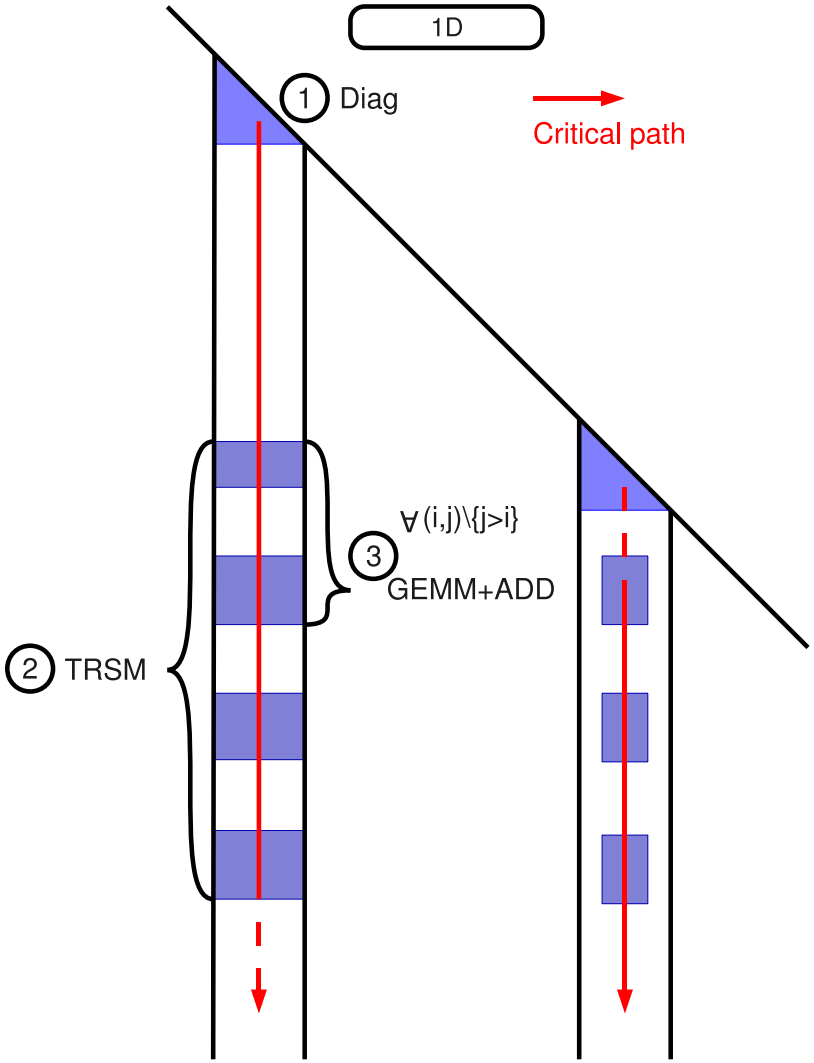


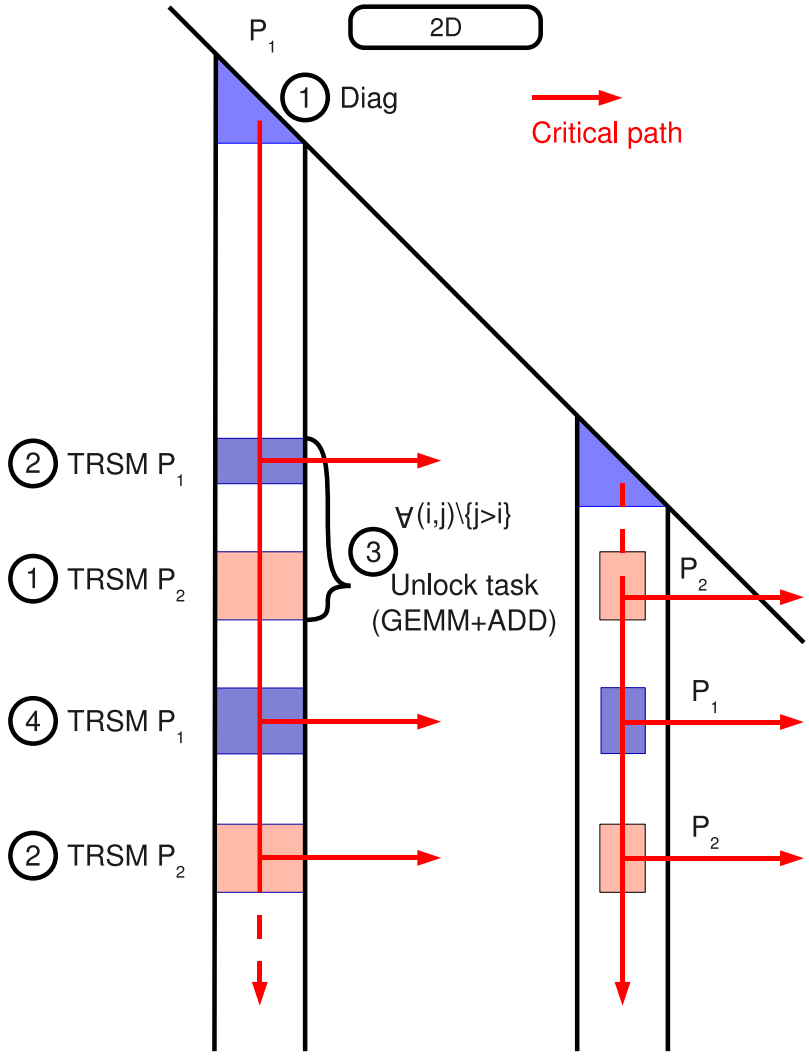
Distribution 1D et 2D





Distribution 1D et 2D



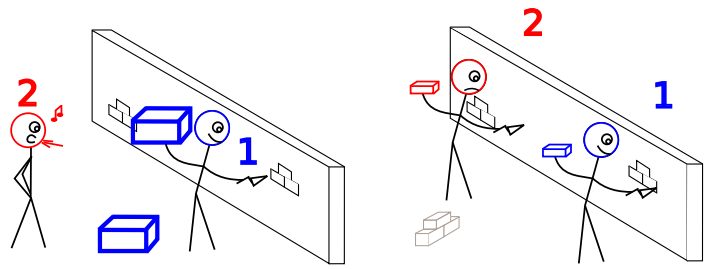


Méthode ESP : Enhanced Sparse Parallelism

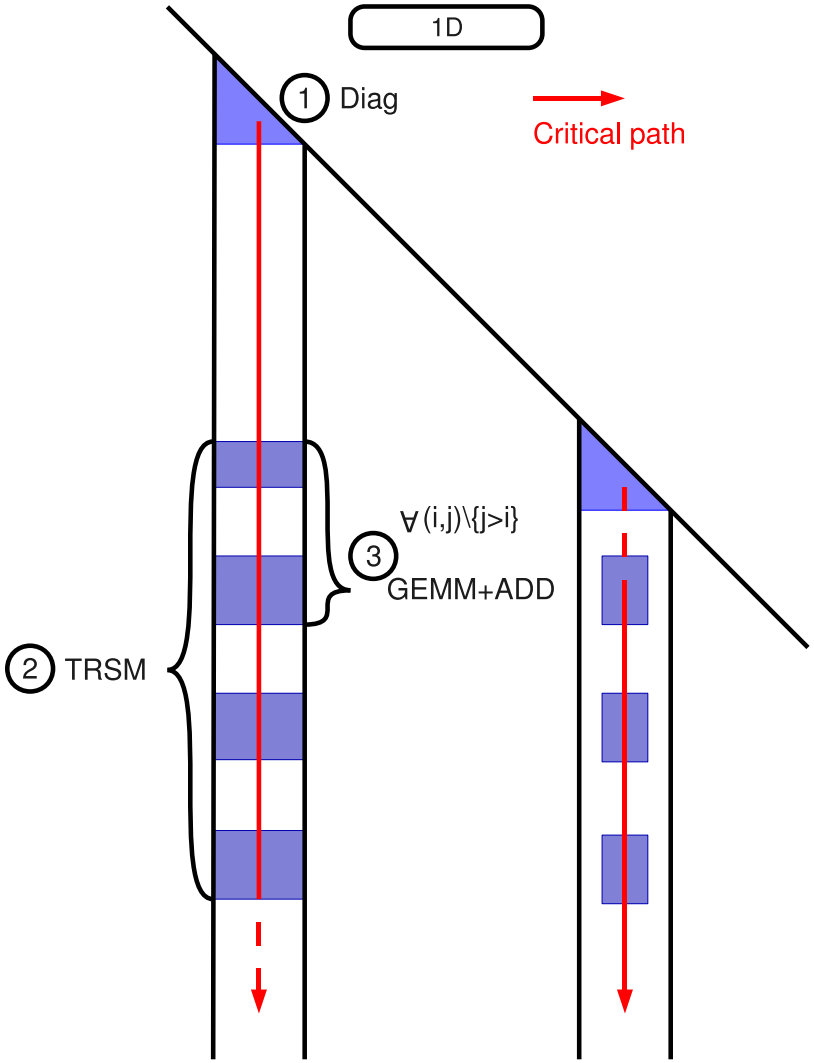
- Multiplication du nombre de cœurs en mémoire partagée
  $\Longrightarrow$ grain de calcul plus fin
- Distribution 2D par blocs plus coûteuse :
  le nombre de tâches est quadratique, contre linéaire en 1D

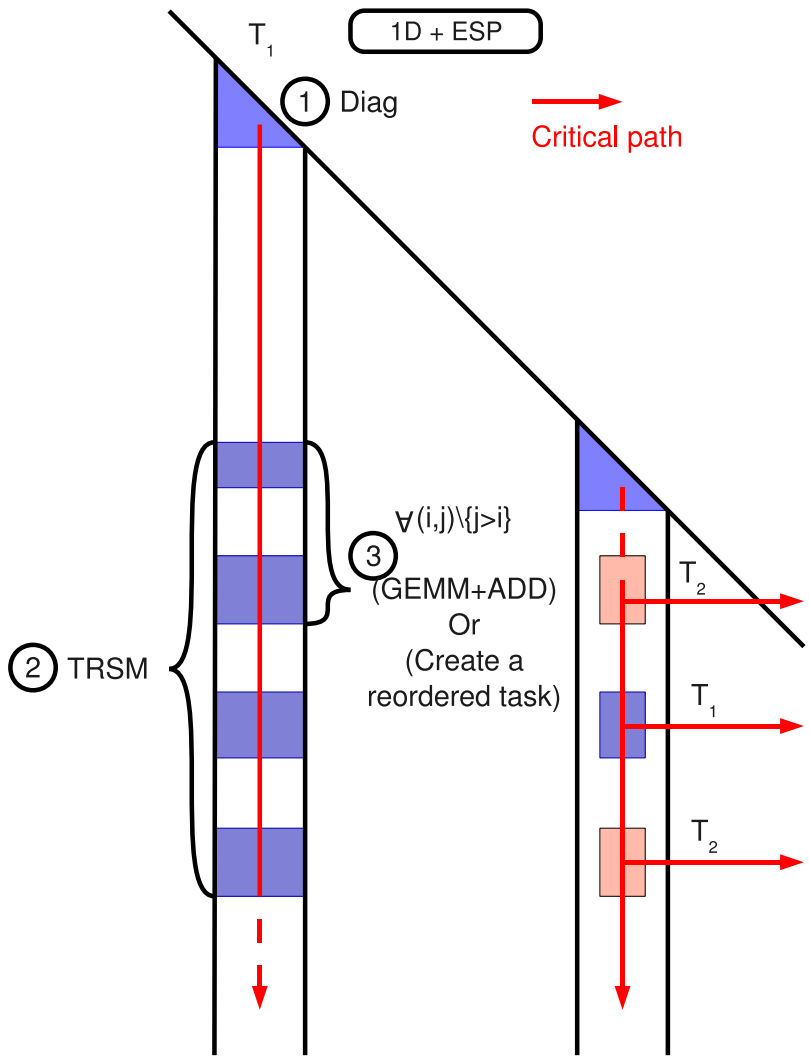
Exploiter le nouvel ordonnancement pour créer dynamiquement des tâches 2D au sein des nœuds :

- ne remet pas en cause la distribution initiale
- fournit un grain de calcul plus fin sur chaque nœud

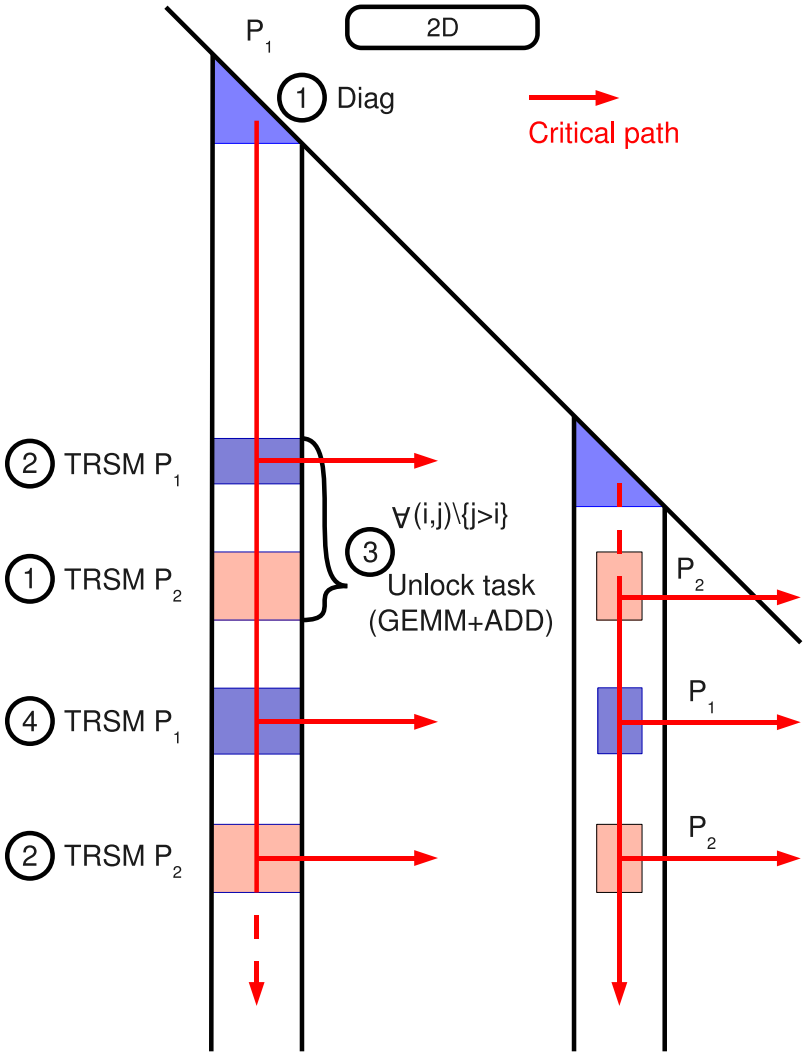


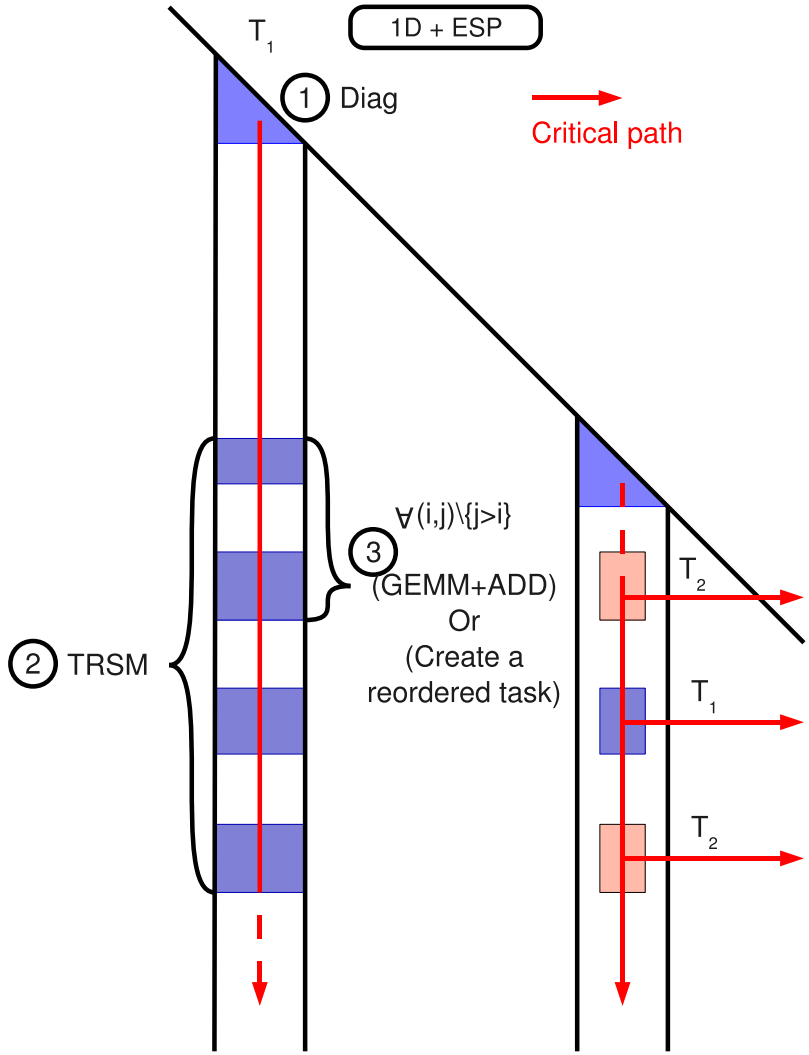
Distribution 1D VS ESP





Distribution 2D VS ESP





Évaluation en mémoire partagée

Matrice	Borderline			Hagrid			Decrypthon		
	Dyn.	ESP	N_t	Dyn.	ESP	N_t	Dyn.	ESP	N_t
Matr5	390	**384**	8091	352	**328**	11755	151	**149**	10305
Matr6	363	**357**	7623	315	**309**	10771	150	**149**	11012
Audi	208	**205**	1547	196	**174**	5579	99,9	**97,5**	6454
Nice20	**211**	**211**	4201	173	**165**	5414	89,3	**88,4**	6981
Inline	7,41	**7,09**	6	14,7	**14**	33	6,12	**5,57**	161
Nice25	2,81	**2,74**	136	5,22	**5,15**	331	1,9	**1,89**	251
MchInf	**2,42**	**2,42**	57	3,47	**3,36**	173	**1,72**	1,81	154
Thread	2,04	**1,92**	147	2,18	**2,09**	201	**1,02**	1,04	201
3DSpectralWave	**1590**	**1590**	8385	960	**946**	15450	554	**538**	16371
3DSpectralWave2	279	**268**	2415	**168**	**168**	191	100	**99.4**	121
Haltere	**131**	132	947	94	**91,6**	290	47,3	**47**	136
Fem_Hifreq_Circuit	**96,2**	96,8	0	73,1	**70,3**	21	**31,6**	**31,6**	126
Mono_500Hz	49,1	**49**	14	33,4	**32**	233	**17,5**	17,7	151

- Allocation pour architectures NUMA
- 1 thread de calcul par cœur disponible
- Ordonnancement dynamique
- Gains de 5%

Évaluation en mémoire distribuée

Matrices	Nb. Nœuds	Titane			Decrypthon		
		Stat.	Dyn.	ESP	Stat.	Dyn.	ESP
Matr5	1	159	151	**150**	161	151	**149**
	2	80,1	78,7	**78,4**	99,6	90,9	**90,1**
	4	56,7	45,3	**45,2**	65	**55,7**	**55,7**
	8	**27,8**	**27,8**	29,6	45,9	**40,9**	43,4
Matr6	1	144	141	**140**	178	150	**149**
	2	75,3	77,3	**74,7**	101	92,3	**92,1**
	4	43	42,1	**41,6**	59,8	58,2	**55,8**
	8	**25,3**	26,1	27,7	44,5	**43**	44
Audi	1	86,9	82,6	**81,7**	100	100	**97,5**
	2	**43,8**	45,9	45,8	67,3	66,8	**62,4**
	4	**24,3**	24,4	**24,3**	33,6	**32,9**	33,2
	8	**14,6**	15,9	14,8	23,2	**22,1**	23,5
3DSpectralWave	1	563	545	**543**	603	544	**538**
	2	285	**279**	286	315	315	**306**
	4	**151**	154	152	189	173	**171**
	8	**84,3**	85,7	85	144	**110**	112

- Allocation pour architectures NUMA
- 1 thread de progression par processus MPI / nœud
- 1 thread de calcul par cœur disponible
- Ordonnancement dynamique
- Gains de 5%

Plan

Présentation du cas

Caractéristiques

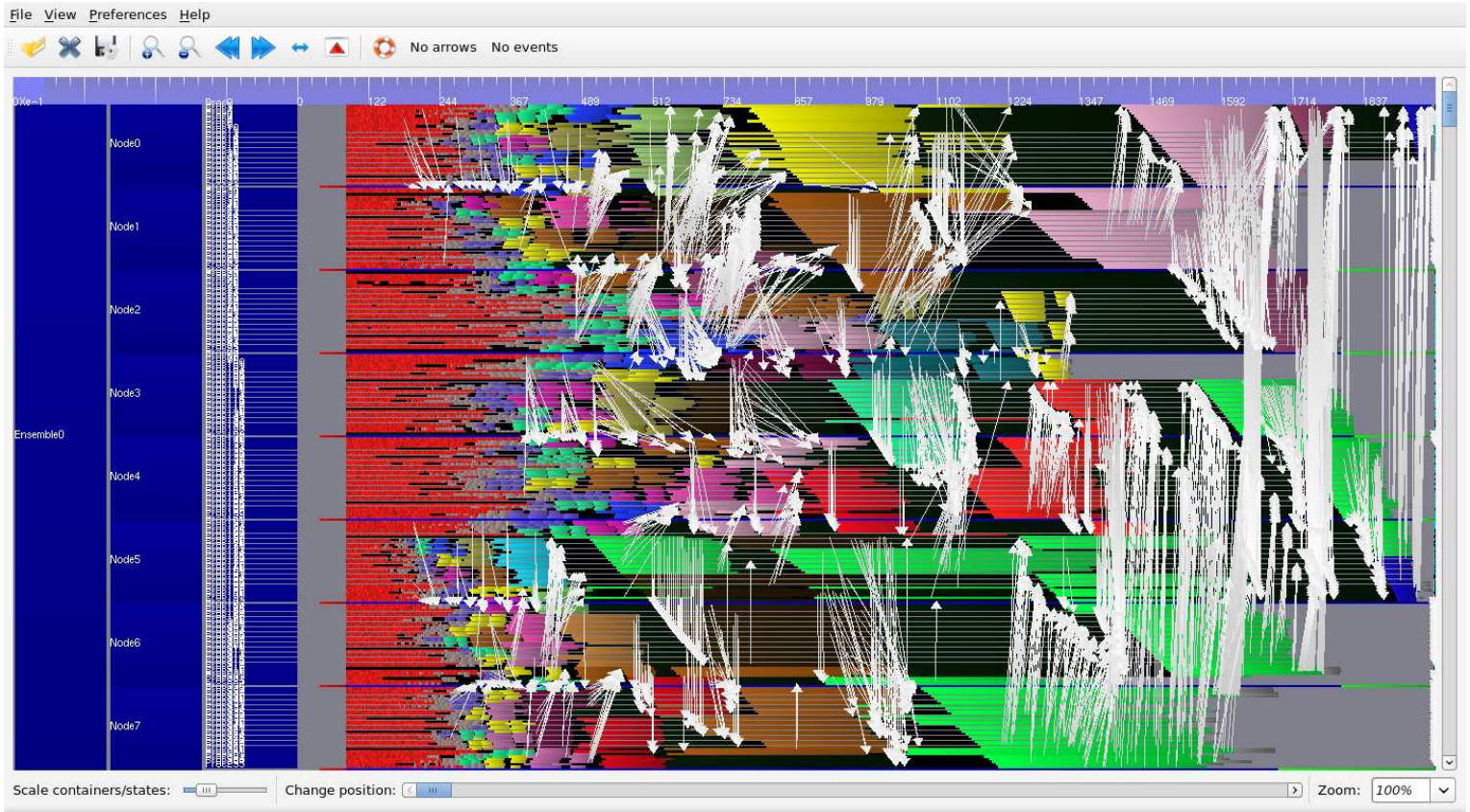
Nombre d'inconnues	10 423 737
Nombre de termes non nuls de A	89 072 871
Nombre de termes non nuls de L	6 724 303 039
Nombre d'OPC pour la factorisation de Cholesky	4,42e+13
Espace mémoire pour le stockage des coefficients	104 Go

Complexes double précision

Problème symétrique

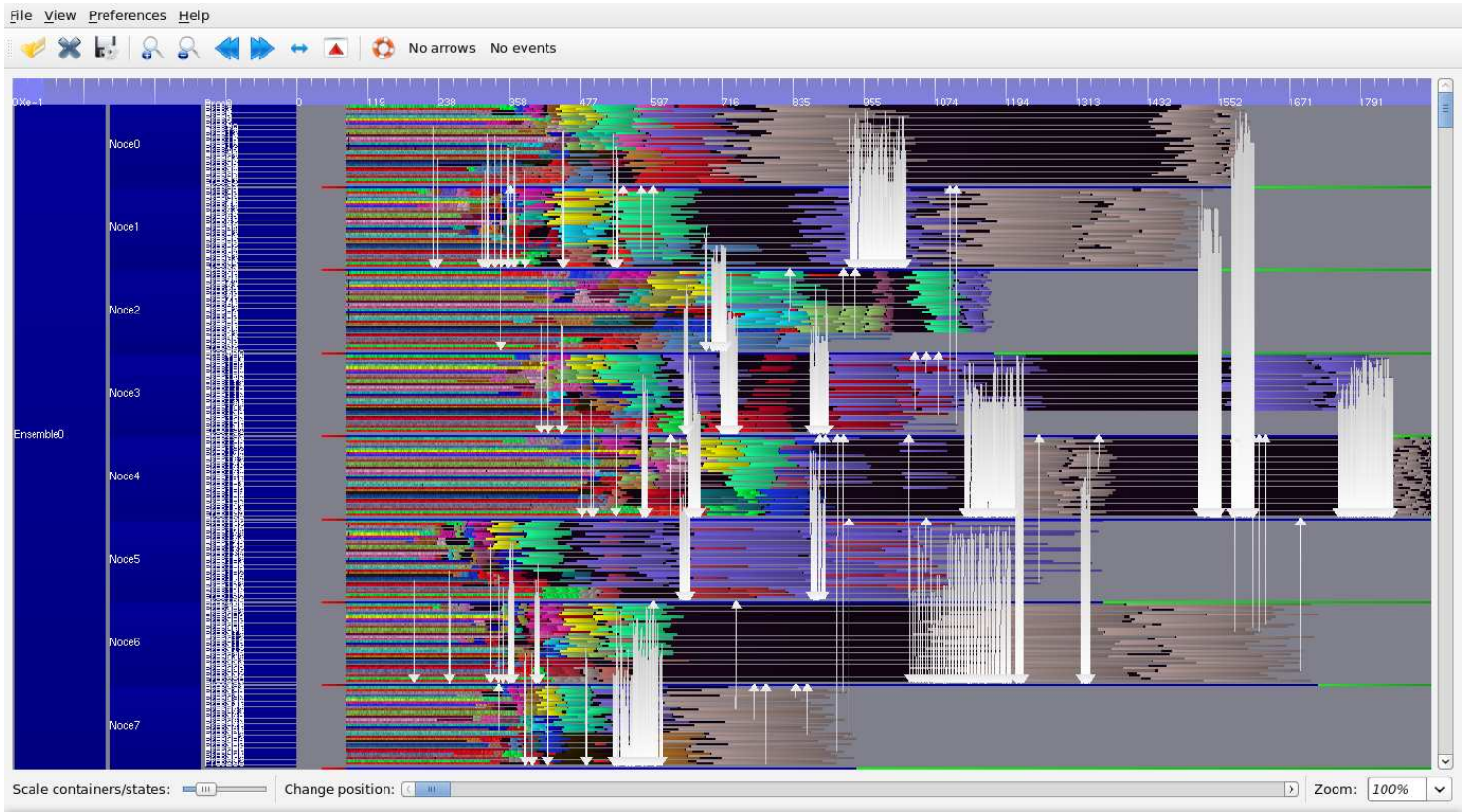
- Problème d'électromagnétisme du CEA Cesta
- Disponible sur la plate-forme GridTLSE
- Machine Vargas de l'IDRIS à base de nœuds comportant 16 Power6 dual-core et 128Go

Ordonnancement statique (8 nœuds × 32 threads)



- Chaque couleur correspond à un niveau de l'arbre d'élimination
- Les blocs noirs sont les temps d'inactivité

Ordonnancement dynamique (8 nœuds × 32 threads)



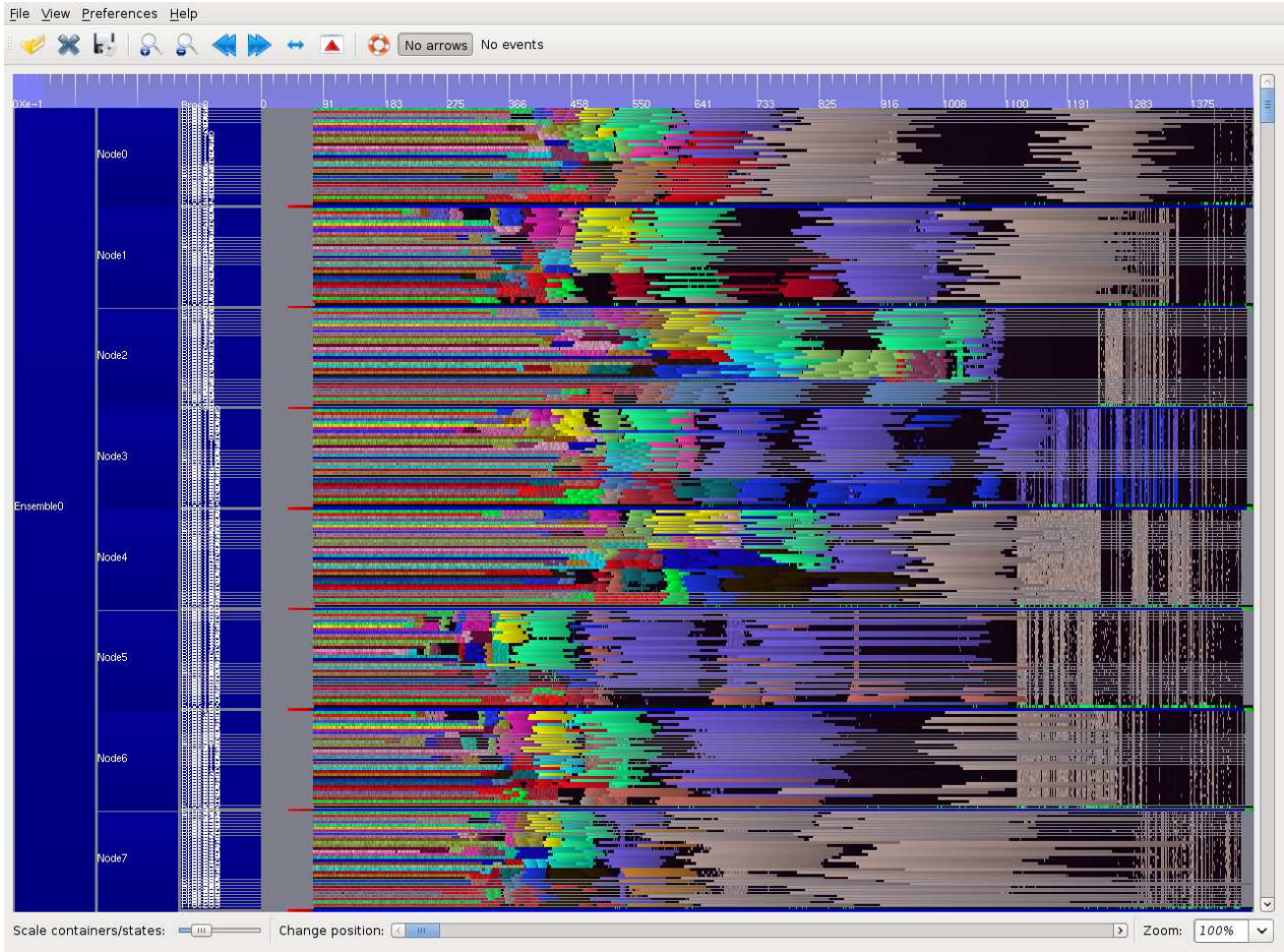
- Distribution statique 1D par blocs-colonnes
- Amélioration de la réactivité des communications
- Limites de la distribution 1D

Utilisation de la distribution statique 2D

TAB.: Temps de la phase de distribution des données en secondes

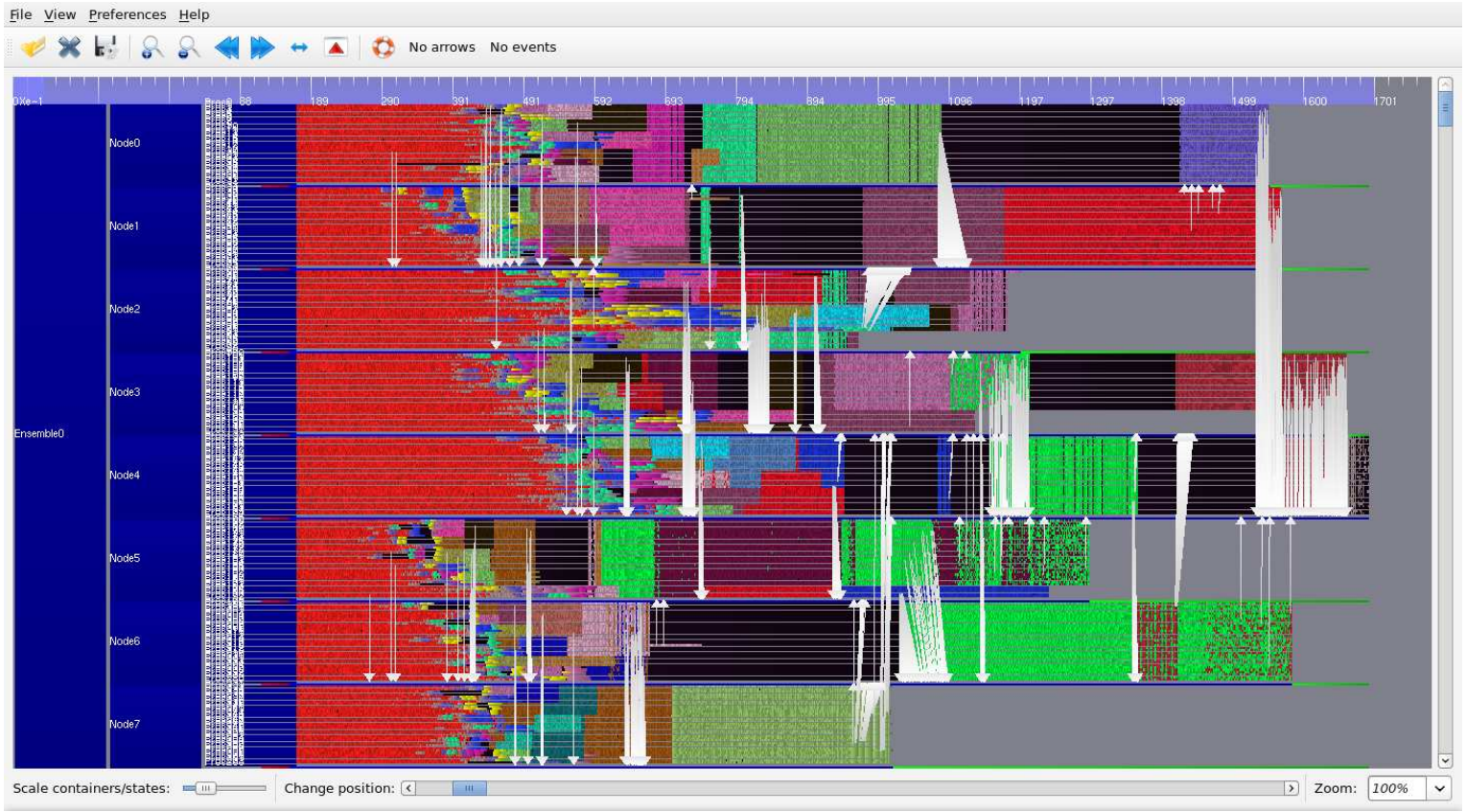
Nb. Niveaux	8 Processus				16 Processus			
	Analyse	Nb. Tâches	Prédiction Min	Prédiction Max	Analyse	Nb. Tâches	Prédiction Min	Prédiction Max
0	41	388941	72,3	90,3	58	389168	24,7	81,8
1	42,6	459957	82,7	88,6	64,3	459961	60,2	80,8
2	66,4	1428395	79,2	82,7	101	1429258	70,6	73,1
3	159	4131521	70	70,5	238	4132793	40,9	45,7
4	516	14801342	67,1	67,6	905	14802801	35,2	35,7
5	791	22557280	67,9	69,1	1480	22562495	33,8	34,4

Ordonnancement dynamique + 2D (8×32)



- Distribution statique 2D sur 2 niveaux de l'arbre d'élimination
- Distribution statique 1D pour les autres niveaux

Ordonnancement dynamique + ESP (8×32)



- Distribution statique 1D par bloc-colonnes

Temps de factorisation

TAB.: Temps de factorisation en secondes.

	8×32
Version Statique	154
Version Dynamique	138
Version Dynamique + Distribution 2D	132
Version Dynamique + Méthode ESP	130

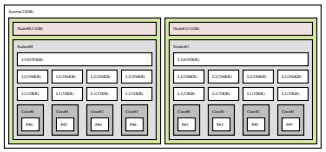
- 8 processus MPI de 32 threads
- 2D : Distribution de 3 niveaux de l'arbre d'élimination
- ESP : Distribution statique 1D

Plan

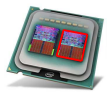
Conclusion et perspectives

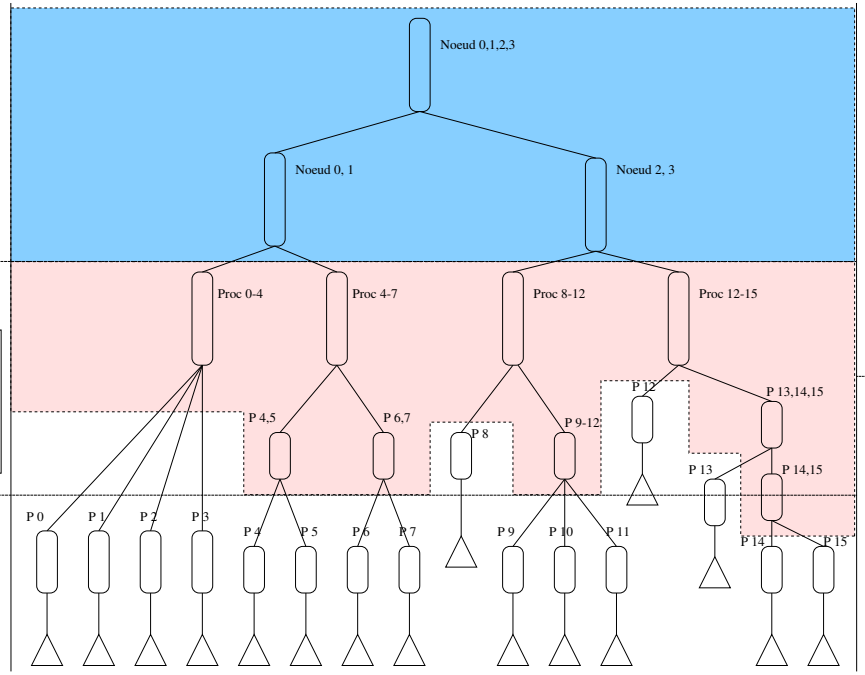
Clusters / MPI

Noeud / Threads



Coeur / BLAS





Distrib.

1D ou 2D statique

1D Statique

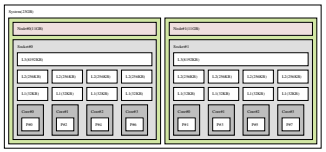
Ordo.

Stat.

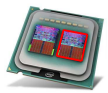
Conclusion et perspectives

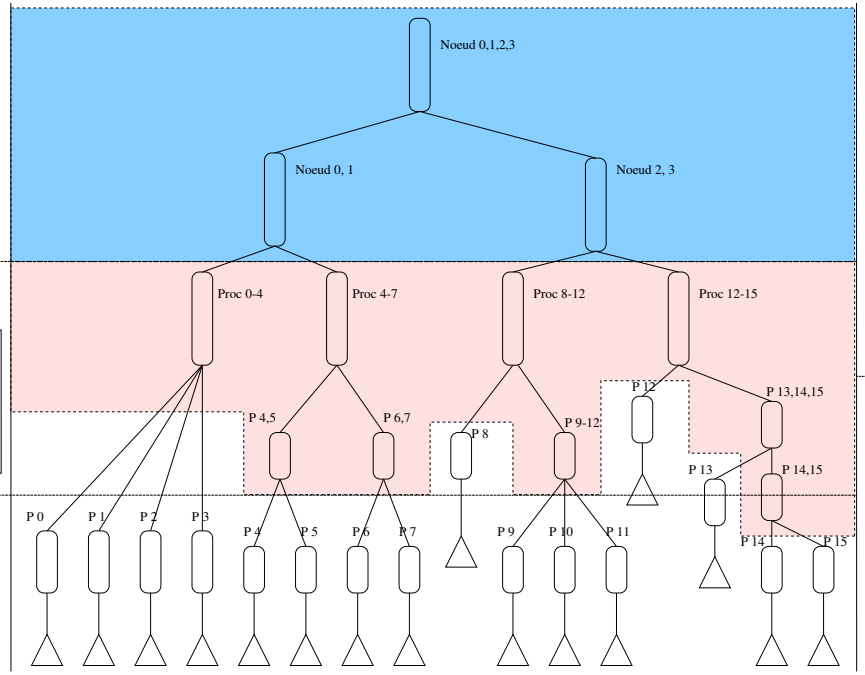
Clusters / MPI

Noeud / Threads



Coeur / BLAS





Distrib.

1D ou 2D statique

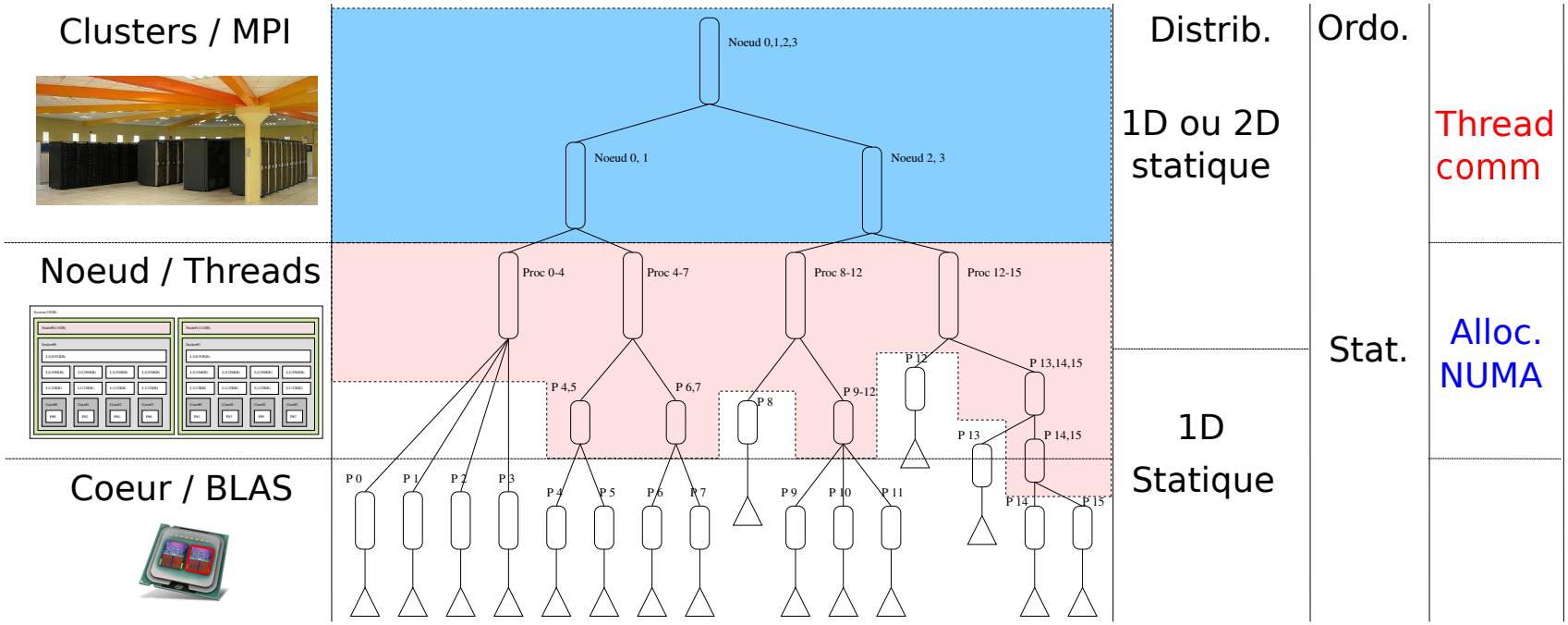
1D Statique

Ordo.

Stat.

Alloc. NUMA

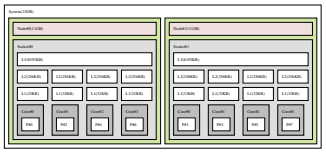
Conclusion et perspectives



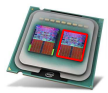
Conclusion et perspectives

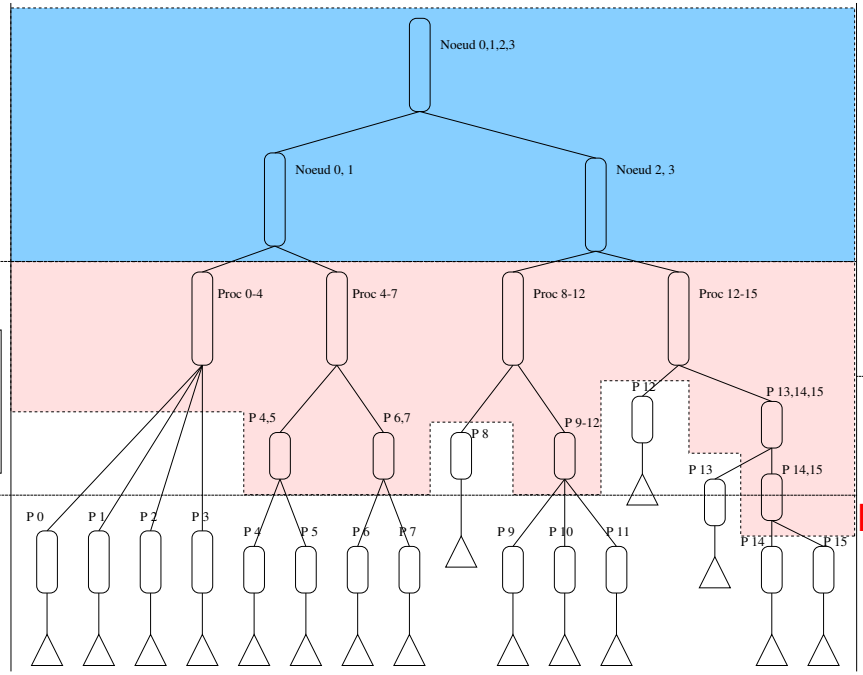
Clusters / MPI

Noeud / Threads



Coeur / BLAS





Distrib.

1D ou 2D statique

1D Dynamique

Ordo.

Dyn.

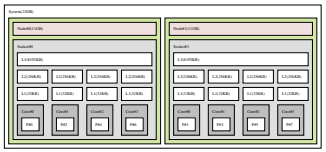
Thread comm

Alloc. NUMA

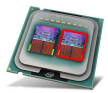
Conclusion et perspectives

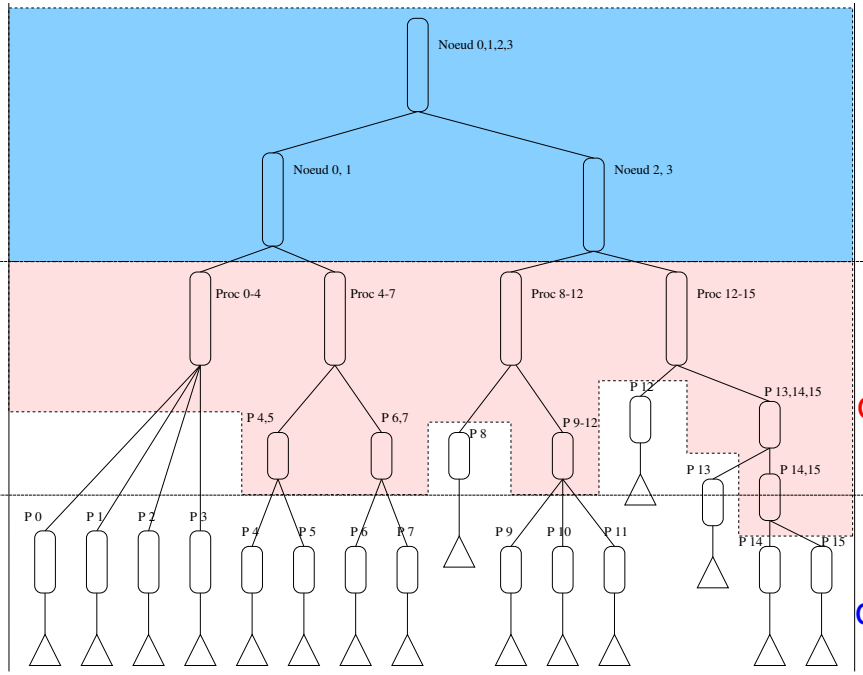
Clusters / MPI

Noeud / Threads



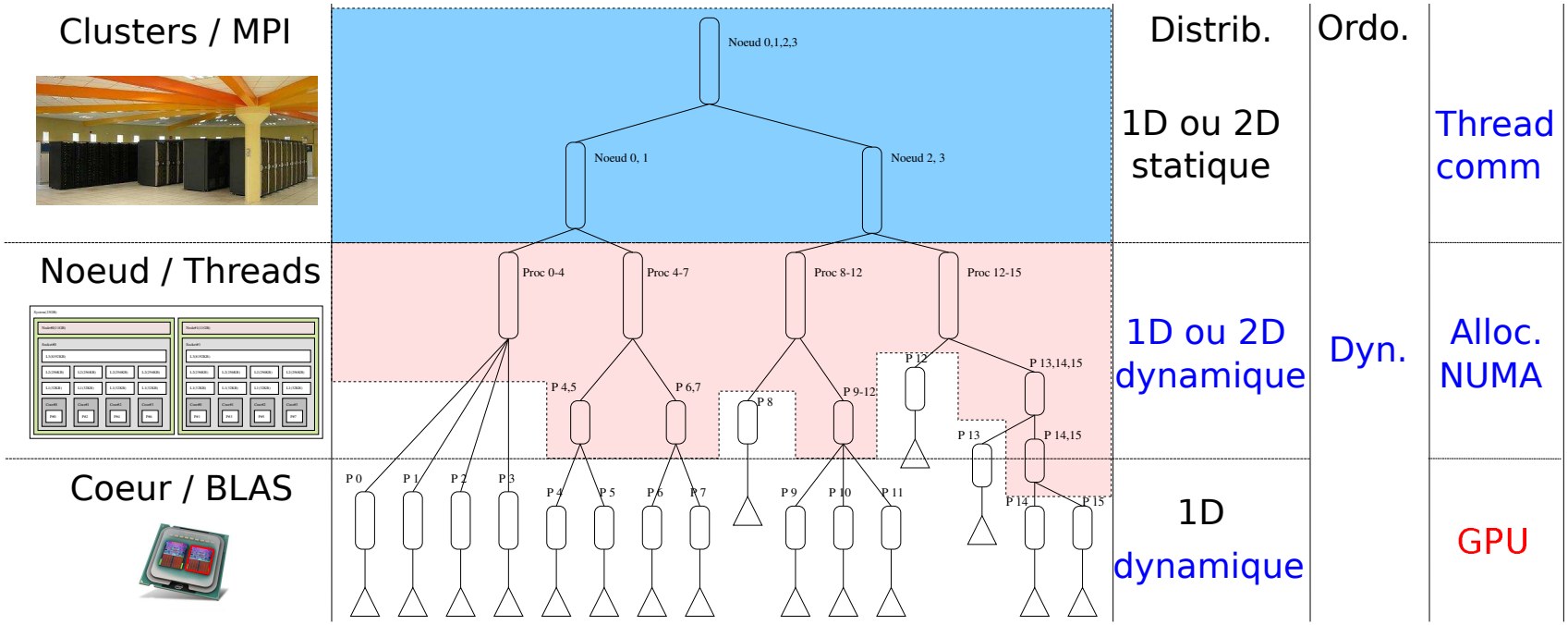
Coeur / BLAS





	Distrib.	Ordo.	
	1D ou 2D statique		Thread comm
	1D ou 2D dynamique	Dyn.	Alloc. NUMA
	1D dynamique		

Conclusion et perspectives



Perspectives

Validation sur de plus grandes architectures

- TERA 10 puis TERA100, CEA
- Bertha : 96 cœurs en mémoire partagée

Perspectives

Validation sur de plus grandes architectures

- TERA 10 puis TERA100, CEA
- Bertha : 96 cœurs en mémoire partagée

Architectures Hétérogènes

- CPU, GPU, SPU, ...
- StarPU

Perspectives

Validation sur de plus grandes architectures

- TERA 10 puis TERA100, CEA
- Bertha : 96 cœurs en mémoire partagée

Architectures Hétérogènes

- CPU, GPU, SPU, ...
- StarPU

Intégration des *Runtimes* existants

- Travaux de l'équipe RunTime
- Bibliothèque de thread : Marcel
- Gestion mémoire : MaMI
- BubbleSched
- Communications : NewMadeleine

What else ?

- L'ensemble des travaux est disponible ou sera disponible dans la prochaine version du solveur PaStiX.
- Réalisation d'un logiciel de visualisation de trace performant : ViTE

Jean-François Méhaut

Comparaison de deux types de parcours de l'arbre

Matrice	Parcours 1	Parcours 2
Matr5	**390**	414
Matr6	**363**	400
Audi	**208**	284
Nice20	**211**	251
Inline	**7,41**	8,6
Nice25	**2,81**	3,19
MchInf	**2,42**	2,88
Thread	**2,04**	3,74
3DSpectralWave	**1590**	2090
3DSpectralWave2	**279**	320
Haltere	**284**	316
Fem_Hifreq_Circuit	**96,2**	104
Mono_500Hz	**49,1**	53,8