

DE LA RECHERCHE À L'INDUSTRIE



DÉPARTEMENT
Sciences et Technologies
de l'Information
et de la Communication

Co-design de noyaux irréguliers sur machine manycore : cas de l'adaptation de maillages.

encadré par :

- Pr. Franck Pommereau (IBISC)
- Dr. Franck Ledoux (CEA)

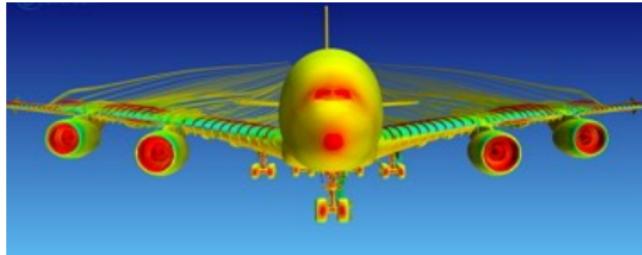
Hoby Rakotoarivelo

Soutenance de thèse, 6 Juillet 2018

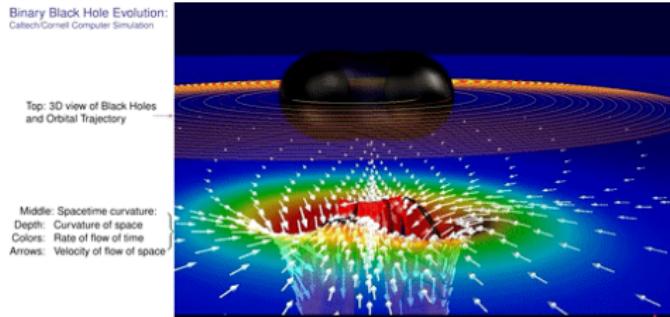
Prédire ou reproduire numériquement des phénomènes physiques ;

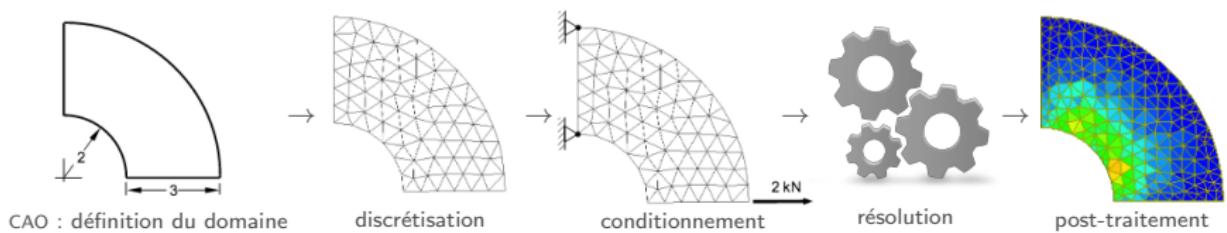
- 1 industriel : accélérer la conception,
- 2 scientifique : étudier des phénomènes inobservables ou hors de portée.

écoulements d'air sur un avion [Ansys]



collision de trous noirs [Caltech]



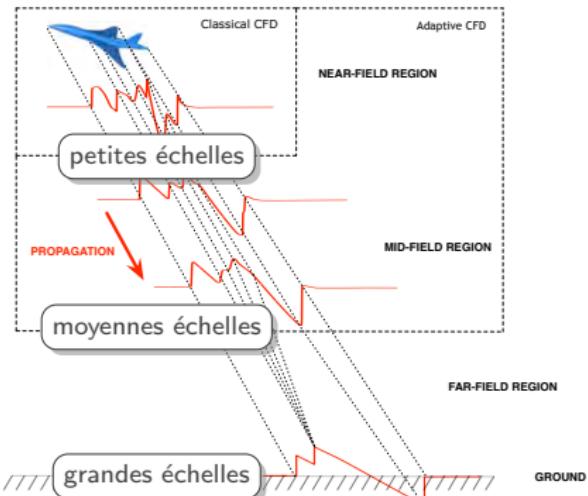


Exemple de workflow en simulation numérique.

le temps de calcul dépend du nombre de points.

Exemple : prédition du bang sonique [Los08].

- taille avion 40m, altitude 10km,
- but : minimiser intensité du choc au sol.



[Los08] Adrien Loseille. Anisotropic 3D hessian-based multi-scale and adjoint-based mesh adaptation for CFD. Application to high fidelity sonic boom prediction. Ph.D thesis, UPMC, 2008 .



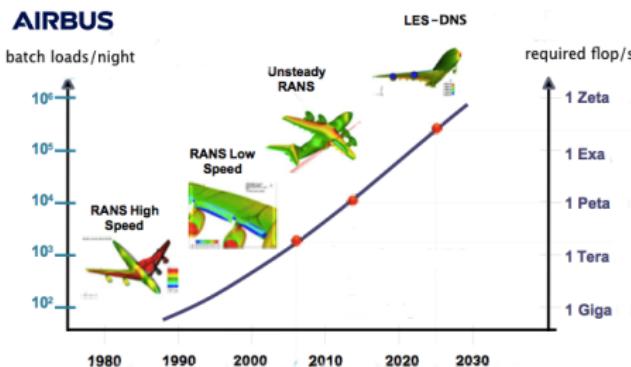
statistiques sur une machine 8-core classique,

it	points	tétrahédres	durée
5	432 454	2 254 826	1h10'
10	608 369	3 294 197	2h54'
15	1 104 910	6 243 462	6h09'
20	1 757 865	10 125 724	11h15'
25	2 572 814	14 967 820	18h47'
30	3 299 367	19 264 402	28h35'

Défis :

- 1 avoir des simulations plus réalistes :
 - ✓ focus sur les méthodes numériques.
 - ✗ temps de calcul conséquent.
- 2 réduire le temps de calcul.

Évolution des simulations en aéronautique.



RANS : Reynolds-Average Navier-Stokes,

LES : Large Eddy Simulation,

DNS : Direct Numerical Simulation.

[SDT06,Cha15].

Pistes possibles :

- 1 calcul parallèle,
- 2 adaptation de maillages.

Idée : combiner les deux.

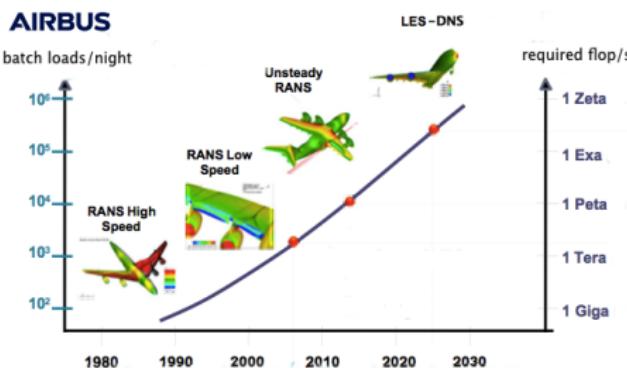
[SDT06] Sagaut et al. Multiscale and multiresolution approaches in turbulence. Imperial College Press, UK, 2006.

[Cha15] Éric Chaput. Exascale needs and challenges for aeronautics industry. PRACEdays15, 2015

Défis :

- 1 avoir des simulations plus réalistes :
 - ✓ focus sur les méthodes numériques.
 - ✗ temps de calcul conséquent.
- 2 réduire le temps de calcul.

Évolution des simulations en aéronautique.



RANS : Reynolds-Average Navier-Stokes,

LES : Large Eddy Simulation,

DNS : Direct Numerical Simulation.

[SDT06,Cha15].

[SDT06] Sagaut et al. Multiscale and multiresolution approaches in turbulence. Imperial College Press, UK, 2006.

[Cha15] Éric Chaput. Exascale needs and challenges for aeronautics industry. PRACEdays15, 2015

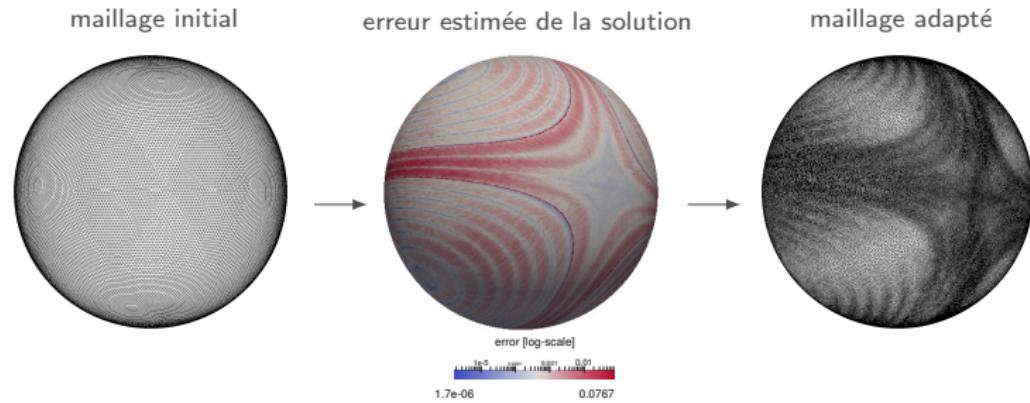
Pistes possibles :

- 1 calcul parallèle,
- 2 adaptation de maillages.

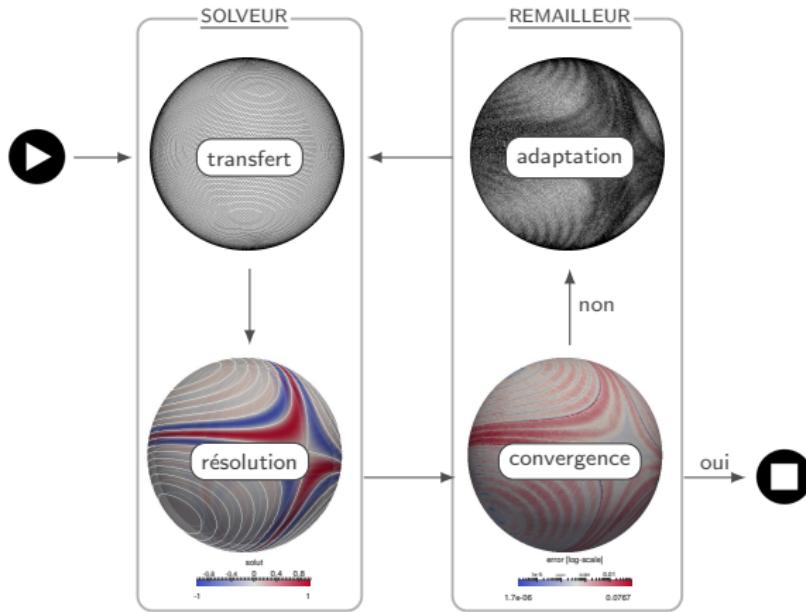
Idée : combiner les deux.

Principe :

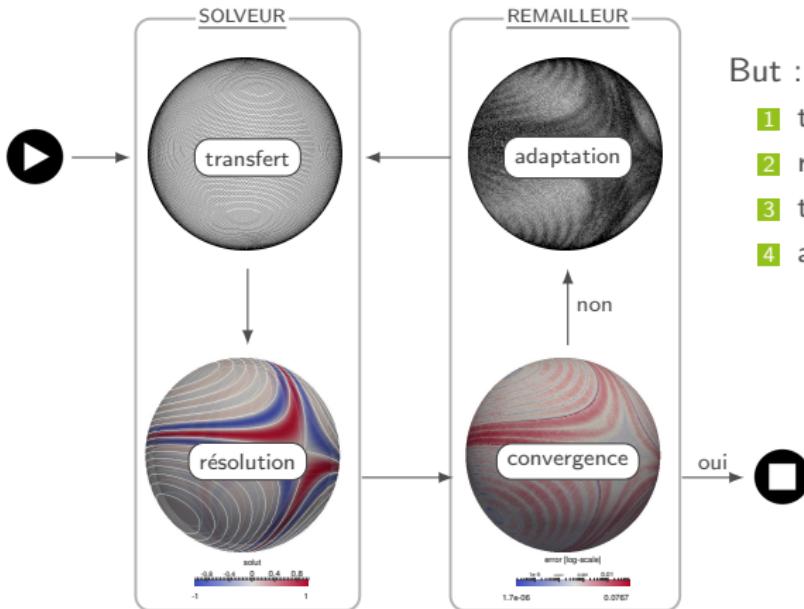
- 1 estimer l'erreur de la solution physique calculée sur le domaine,
- 2 en déduire une distribution de points sur le domaine,
- 3 modifier itérativement le maillage par des opérations locales (noyaux).



à chaque pas de temps :



à chaque pas de temps :



But : paralléliser chaque phase.

- 1 transfert : bien étudié.
- 2 résolution : bien étudié.
- 3 test de convergence : pas trop dur.
- 4 adaptation : non triviale,
 - tâches non prédictibles [PKN+05],
 - faible réutilisation de données [KRS11].
 - ✗ peut **ralentir** tous les autres.

[PKN+05] Pingali et al. Amorphous data-parallelism in irregular algorithms. University of Texas, 2005
 [KRS11] Korch et al. Memory-intensive applications on a manycore processor. HPCC. 2011

Simulations de fluides à grande échelle \Rightarrow machines exaflopiques.

machine exaflorique du CEA [phase 1].



1 exaflop : 10^{18} opérations flottantes par sec.

- investissements en millions d'euros : rendement optimal des calculateurs,
- consommation électrique de l'ordre du mégawatt : changements majeurs.

Réduire la consommation d'électricité :

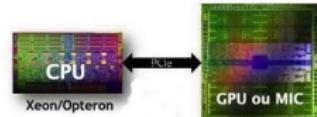
- maintenir puissance consommée (watt),
- augmenter capacité de calcul (flops).
- ✓ multiplication de cores peu cadencés.

Simulations de fluides à grande échelle ⇒ machines exaflopiques.

machine exaflopique du CEA [phase 1].



nœud de calcul



évolution des accélérateurs

sortie	architecture	cores	GHz	instr.
2011	Tilera Tile-Gx	100	1.2	RISC
2012	Intel Knights Corner	60	1.2	RISC
2016	Nvidia Tesla Kepler	2 280	1	RISC
2014	Kalray MPPA Bostan	256	0.4	RISC
2015	Kalray MPPA Coolidge	1 024	1	RISC
2015	Nvidia Tesla Maxwell	3 272	1.2	RISC
2015	Adapteva Epiphany-V	1 024	1	RISC
2016	Intel Knights Landing	72	1.5	CISC
2016	Nvidia Tesla Pascal	3 840	1.4	RISC
2017	Intel Knights Mill	72	1.5	CISC
2018	Nvidia Tesla Volta	6 048	1.2	RISC

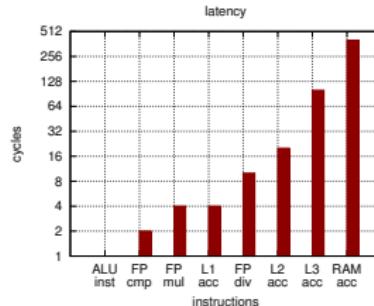
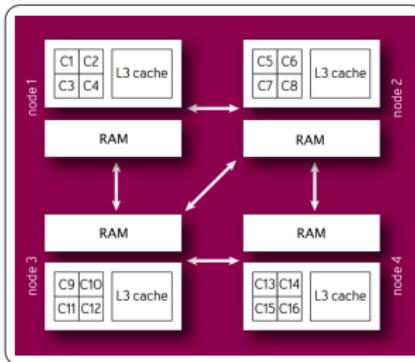
Réduire la consommation d'électricité :

- 1 maintenir puissance consommée (watt),
- 2 augmenter capacité de calcul (flops).
- ✓ multiplication de cores peu cadencés.

Contraintes matérielles :

- 1 rendement [GFlops-GHz-GB] par core faible,
- 2 cout élevé et asymétrique des accès-mémoire.

exemple : machine NUMA



Contraintes applicatives :

- 1 exposer un parallélisme très fin et très élevé,
- 2 pas d'indirections, réutilisation max. de données déjà accédées.

Deux axes :

- 1 Étude de noyaux surfaciques locality-aware [RL18, LR18],
- 2 Portage de noyaux sur accélérateurs manycore [RLP16, RLP+17],

Deux axes :

1 Étude de noyaux surfaciques locality-aware [RL18, LR18],

- reconstruction locale et projection de points,
- noyau de lissage de points mixte diffusion-optimisation,
- preuve d'équivalence des mesures usuelles de gradation,
- transport optimal de tenseurs métriques (preuve et algorithme).

2 Portage de noyaux sur accélérateurs manycore [RLP16, RLP+17],

- extraction du parallélisme basée sur les graphes,
- approche de structuration de noyaux irréguliers,
- schéma de réduction asynchrone ou NUMA-aware de données,
- synchronisation grain fin pour les primitives topologiques.

Deux axes :

- 1 Étude de noyaux surfaciques locality-aware [RL18, LR18],**
 - reconstruction locale et projection de points,
 - noyau de lissage de points mixte diffusion-optimisation,
 - preuve d'équivalence des mesures usuelles de gradation,
 - transport optimal de tenseurs métriques (preuve et algorithme).
- 2 Portage de noyaux sur accélérateurs manycore [RLP16, RLP+17],**
 - extraction du parallélisme basée sur les graphes,
 - approche de structuration de noyaux irréguliers,
 - schéma de réduction asynchrone ou NUMA-aware de données,
 - synchronisation grain fin pour les primitives topologiques.

1 Étude de noyaux surfaciques locality-aware

- Rappels de nos objectifs
- Problématique de localité et choix de noyaux
- Primitive de projection de points
- Noyau de lissage mixte diffusion-descente
- Résultats numériques

2 Portage des noyaux sur accélérateurs manycore

- Rappels de nos objectifs
- Problématique d'irrégularité et approche proposée
- Extraction du parallélisme à grain fin
- Synchronisation pour les primitives topologiques
- Résultats numériques

3 Conclusion

1 Étude de noyaux surfaciques locality-aware

- Rappels de nos objectifs
- Problématique de localité et choix de noyaux
- Primitive de projection de points
- Noyau de lissage mixte diffusion-descente
- Résultats numériques

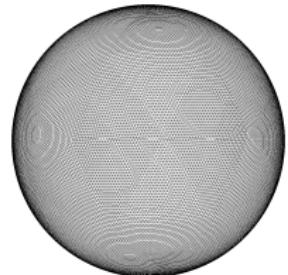
2 Portage des noyaux sur accélérateurs manycore

- Rappels de nos objectifs
- Problématique d'irrégularité et approche proposée
- Extraction du parallélisme à grain fin
- Synchronisation pour les primitives topologiques
- Résultats numériques

3 Conclusion

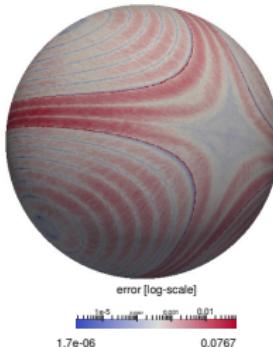
Ce qu'on veut faire

maillage initial

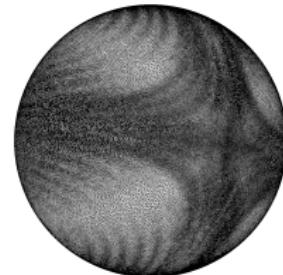


1

erreur estimé de la solution



maillage adapté

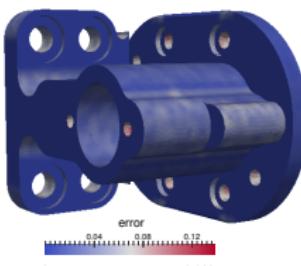


maillage initial

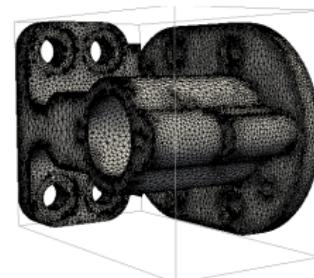


2

erreur estimé de la surface



maillage adapté

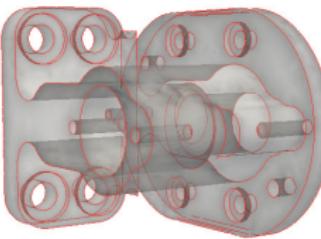


Vue synthétique des étapes de remaillage :

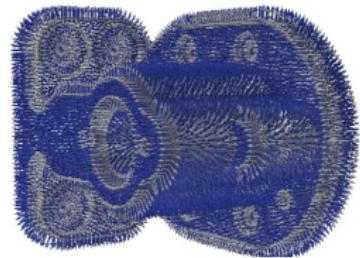
1. extraire topologie



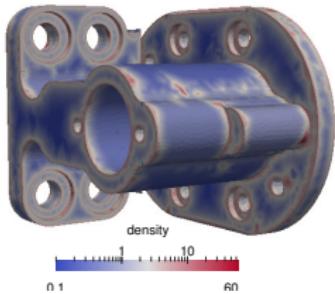
2. extraire arêtes vives



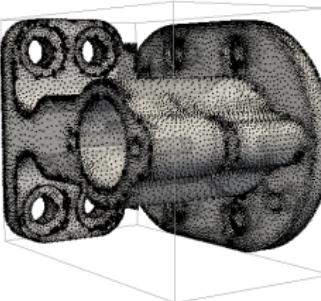
3. extraire normales



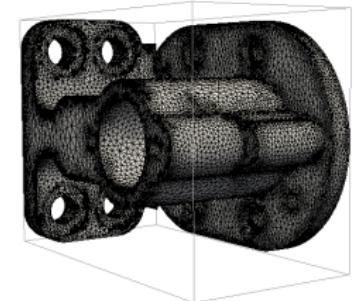
4. extraire distribution des points



5. rééchantillonage



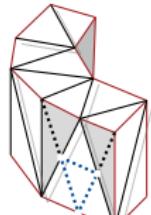
6. régularisation



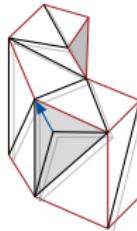
Noyau : algorithme dédié à une opération spécifique sur le maillage.

⚠ parallélisation des noyaux \Rightarrow localité maximale ⚠

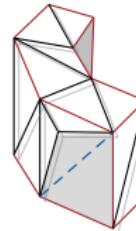
- 1 nous n'utilisons que des noyaux impliquant un voisinage restreint et borné.
- 2 nous concevons des **primitives géométriques précis** pour compenser.



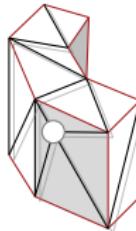
raffinement



simplification



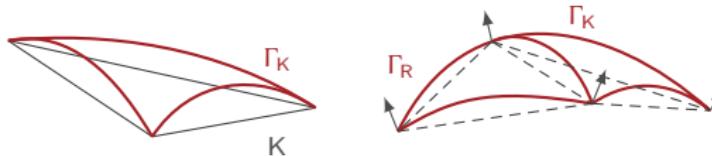
relaxation



lissage

⚠ On ne connaît pas la surface idéale ⚠

Idée : reconstruire la surface localement à l'aide de "patchs".



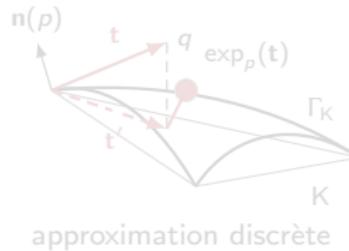
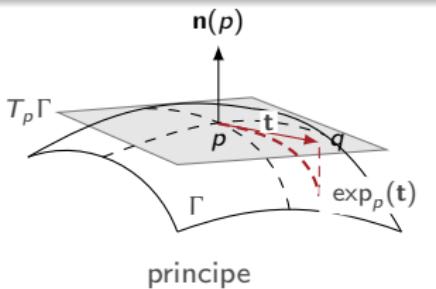
Patch : triangle courbe défini par des points de contrôle.

- 1 on veut que chaque courbe soit une géodésique,
- 2 on veut garantir l'unicité du plan tangent en tout point ;
 - cela fait normalement intervenir les mailles voisines.
- ✓ ici, les points de contrôle sont calculés individuellement sur une maille.

Idée : utiliser un **opérateur continu** pour faire de la **projection de points**.

Application exponentielle

Soit p un point de la variété Γ et R une région de son plan tangent $T_p\Gamma$. L'application exponentielle $\exp_p : R \rightarrow \Gamma$ fait correspondre à chaque vecteur tangent t de R le segment courbe géodésique γ d'origine p , de vitesse initiale $\frac{d\gamma}{ds}(0) = t$ et de longueur $\|t\|$.



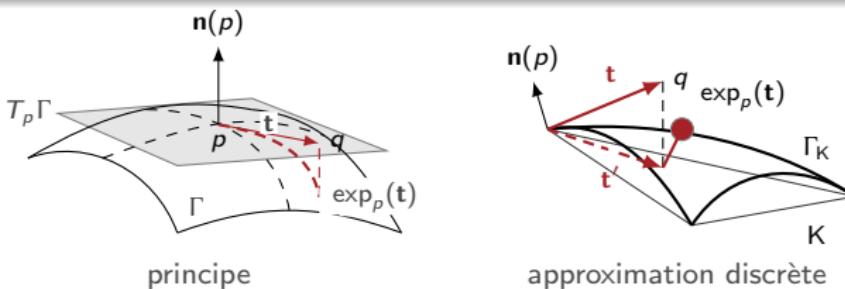
Principe :

- 1 retrouver la maille pointée par t , puis le projeté $t' = \overrightarrow{pq}$ de t sur cette maille,
- 2 ajuster t' par une recherche linéaire puis projeter \tilde{q} sur le patch associé à la maille.

Idée : utiliser un **opérateur continu** pour faire de la **projection de points**.

Application exponentielle

Soit p un point de la variété Γ et R une région de son plan tangent $T_p\Gamma$. L'application exponentielle $\exp_p : R \rightarrow \Gamma$ fait correspondre à chaque vecteur tangent t de R le segment courbe géodésique γ d'origine p , de vitesse initiale $\frac{d\gamma}{ds}(0) = t$ et de longueur $\|t\|$.



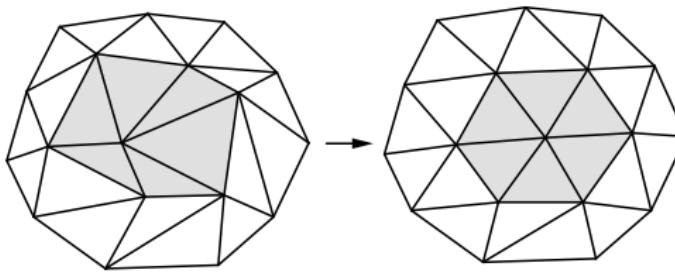
Principe :

- 1 retrouver la maille pointée par t , puis le projeté $t' = \overrightarrow{pq}$ de t sur cette maille,
- 2 ajuster t' par une recherche linéaire puis projeter \tilde{q} sur le patch associé à la maille.

En résumé, nous avons proposé une primitive de projection de points qui :

- étant donné un vecteur déplacement $t = \vec{pq}$, donne le projeté de q sur la surface ;
- ✓ est précise : interpolation quartique de la surface et respectant la continuité G^1 ;
- ✓ est locale : tous les calculs se font localement à une maille ;

On va maintenant l'illustrer sur un noyau en particulier : le lissage.



But : améliorer la qualité des mailles par déplacement de points.

Important :

- 1** l'erreur estimée d'une solution physique est liée à la qualité des mailles,
- 2** on ne s'occupe pas de la qualité des mailles dans les autres noyaux.

- 1 diffusion : surface vue comme un signal à débruiter par un filtre laplacien.
 - point repositionné au barycentre pondéré de son voisinage ;
 - ✓ simple, permet de débruiter la géométrie et de régulariser les mailles [OBB00],
 - ✗ tend à réduire la surface et n'améliore pas strictement la qualité des mailles [Tau95].
- 2 optimisation : minimisation explicite d'une fonction coût.
 - déplacement pas à pas du point jusqu'à atteindre un minimum local ;
 - ✓ features avancées : débruitage volume-preserving [KKG+01,Jia06], amélioration de la pire maille dans des cas particuliers [Aub14] ;
 - ✗ très coûteux, potentiellement lent à converger.

[Tau95] Gabriel Taubin. A signal processing approach to fair surface design. SIGGRAPH'95, 1995

[OBB00] Othake, Belyaev and Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization, GMP, 2000

[KKG+01] Kuprat and al. Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines. JCP, 2001

[Jia06] Xiangmin Jiao. Volume and feature preservation in surface mesh optimization. IMR, 2006

[Aub14] Daniel Aubram. Optimization-based smoothing algo. for triangle meshes over arbitrarily shaped domains. Tech. rep., 2014.

1 diffusion : surface vue comme un signal à débruiter par un filtre laplacien.

- point repositionné au barycentre pondéré de son voisinage ;
- simple, permet de débruiter la géométrie et de régulariser les mailles [OBB00],
- tend à réduire la surface et n'améliore pas strictement la qualité des mailles [Tau95].

2 optimisation : minimisation explicite d'une fonction coût.

- déplacement pas à pas du point jusqu'à atteindre un minimum local ;
- features avancées : débruitage volume-preserving [KKG+01,Jia06], amélioration de la pire maille dans des cas particuliers [Aub14] ;
- très coûteux, potentiellement lent à converger.

Idée : combiner les deux comme dans [CTS98, Fre99]

[Tau95] Gabriel Taubin. A signal processing approach to fair surface design. SIGGRAPH'95, 1995

[OBB00] Othake, Belyaev and Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization, GMP, 2000

[KKG+01] Kuprat and al. Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines. JCP, 2001

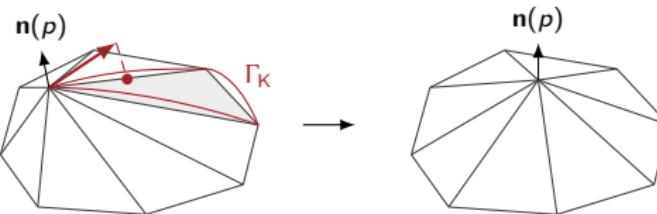
[Jia06] Xiangmin Jiao. Volume and feature preservation in surface mesh optimization. IMR, 2006

[Aub14] Daniel Aubram. Optimization-based smoothing algo. for triangle meshes over arbitrarily shaped domains. Tech. rep., 2014.

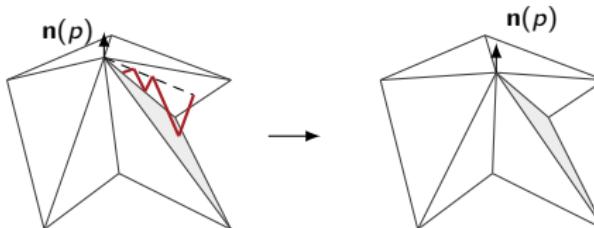
[CTS98] Canann, Tristano and Staten. An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. IMR, 1998

[Fre99] Lori Freitag. On combining laplacian and optimization-based mesh smoothing techniques. MeshTrends, 1999

lissage par diffusion



⚠ la déformation de la surface doit être minimale ⚠



lissage par optimisation

Diffusion : égaliser l'aire des triangles courbes incidentes au point p .

- 1 trouver la direction de déplacement vers le centre de masse du voisinage de p .

$$p \leftarrow \exp_p \left[\alpha \frac{\int_C \rho[x] \log_p[x] dx}{\int_C \rho[x] dx} \right], \text{ avec} \begin{cases} C : \text{voisinage continu de } p, \\ \alpha : \text{facteur d'échelle,} \\ \rho : \text{fonction de densité.} \end{cases}$$

- 2 déplacement accepté si pire maille améliorée entre autres,
sinon on réduit le facteur d'échelle de moitié.
✓ rapide et efficace dans 90% des cas.

Optimisation : minimiser la distorsion de la pire maille incidente à p .

- 1 se déplacer graduellement vers la direction opposée à son gradient,

$$\text{à l'itéré } t, \quad p^{[t+1]} \leftarrow \exp_p[-\alpha \nabla f_k(p^{[t]})], \text{ avec} \begin{cases} f_j : \text{distorsion de la } j^{\text{eme}} \text{ maille} \\ f_k = \max_j f_j[p^{[t]}] \\ \alpha : \text{pas de déplacement optimal} \end{cases}$$

- 2 le calcul du pas α tient compte des autres mailles incidentes à p ;
il est déterminé par une recherche linéaire (selon les critères de Wolfe).
✓ peu d'itérations nécessaires.

Diffusion : égaliser l'aire des triangles courbes incidentes au point p .

- 1 trouver la direction de déplacement vers le centre de masse du voisinage de p .

$$p \leftarrow \exp_p \left[\alpha \frac{\int_C \rho[x] \log_p[x] dx}{\int_C \rho[x] dx} \right], \text{ avec} \begin{cases} C : \text{voisinage continu de } p, \\ \alpha : \text{facteur d'échelle,} \\ \rho : \text{fonction de densité.} \end{cases}$$

- 2 déplacement accepté si pire maille améliorée entre autres,
sinon on réduit le facteur d'échelle de moitié.
- ✓ rapide et **efficace** dans 90% des cas.

Optimisation : minimiser la distorsion de la pire maille incidente à p .

- 1 se déplacer graduellement vers la direction opposée à son gradient,

$$\text{à l'itéré } t, p^{[t+1]} \leftarrow \exp_p[-\alpha \nabla f_k(p^{[t]})], \text{ avec} \begin{cases} f_j : \text{distorsion de la } j^{\text{eme}} \text{ maille} \\ f_k = \max_j f_j[p^{[t]}] \\ \alpha : \text{pas de déplacement optimal} \end{cases}$$

- 2 le calcul du pas α tient compte des autres mailles incidentes à p ;
il est déterminé par une recherche linéaire (selon les critères de Wolfe).
- ✓ peu d'itérations nécessaires.

En résumé, nous avons proposé des noyaux pour :

- l'adaptation de surfaces triangulées,
 - 1 ré-échantillonnage : raffinement et simplification,
 - 2 régularisation : relaxation et lissage.

- avantages :
 - 1 local : n'implique qu'un voisinage restreint et borné,
 - 2 précis : primitives géométriques avancées pour compenser.
(application exponentielle, transport parallèle de tenseurs métriques)

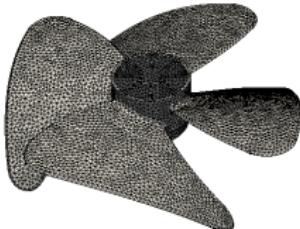
Nous allons maintenant évaluer leur efficacité une fois combinés.

Résultats numériques

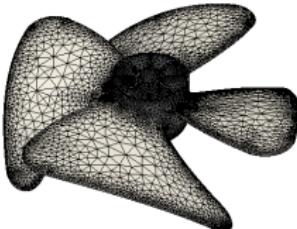
Convergence en erreur et en qualité de mailles

Adaptation à l'erreur estimée de la surface.

propeller_init: 14500 points



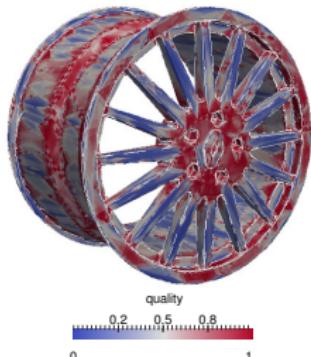
propeller_adapted: 30000 points



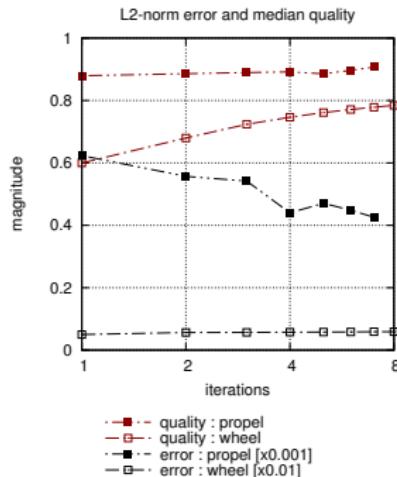
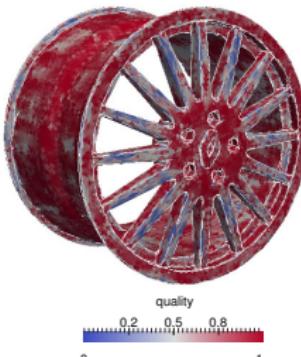
- qualité d'une maille : $\frac{R}{2r}$,
- erreur : distance de Hausdorff local,
- courbures calculées selon [MDS+03].

Phase de régularisation.

wheel_orig: 71830 points



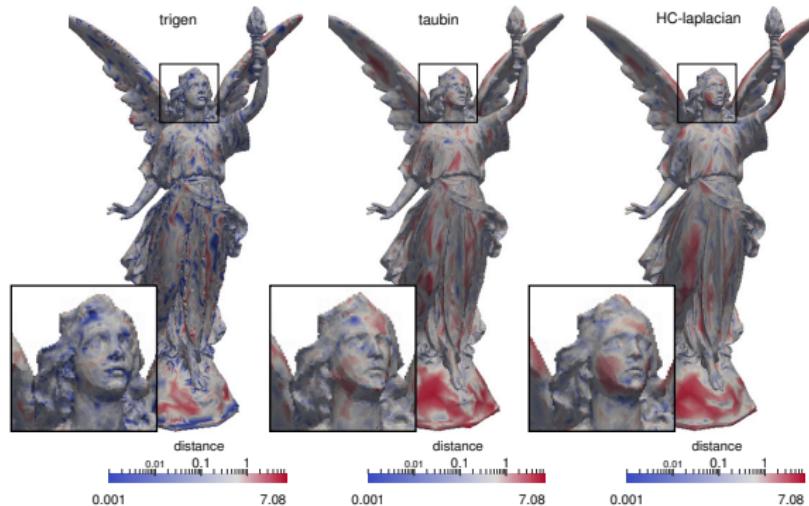
wheel_regul: 71830 points



[MDS+03] Meyer, Desbrun, Schröder and Barr. Discrete differential-geometry operators for triangulated 2-manifolds. Visualization and Maths III, Springer, 2003.

Résultats numériques

Lissage : évaluation de la déformation de la surface.



Paramètres

- $\theta_{\max} = 7^\circ$, ridges : 35° ,
- [Tau95] : $(\lambda, \mu) = (0.5, -0.53)$,
- samples : 115 470, 20 itérés.

Erreur d'approximation

	trigen	[Tau95]	[VMM99]
hausd.	6.960	8.896	6.940
mean	0.427	0.609	0.656
RMS	0.652	0.906	0.958

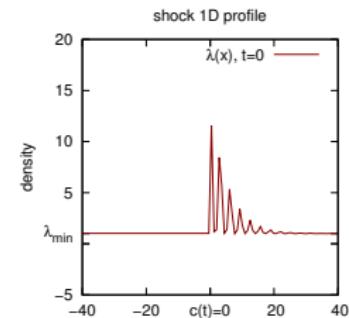
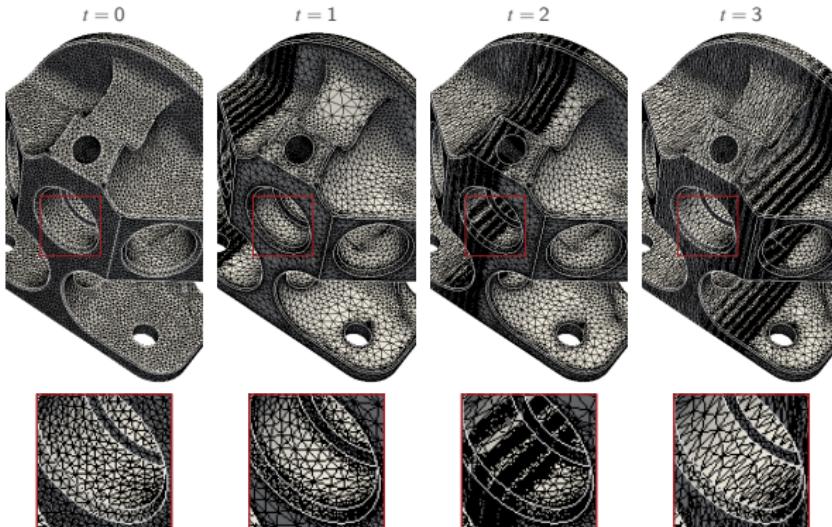
calculé avec Metro [CRS96]

[Tau95] Gabriel Taubin. Curve and surface smoothing without shrinkage. ICCV, 1995.

[VMM99] Vollmer, Mencl and Miller. Improved laplacian smoothing of noisy surface meshes. Eurographics, 1999.

[CRS96] Cignoni, Rocchini and Scopigno. Metro : Measuring error on simplified surfaces. Tech. rep, CNRS, 1996.

Adaptation anisotrope à une onde de choc instationnaire.



1 Étude de noyaux surfaciques locality-aware

- Rappels de nos objectifs
- Problématique de localité et choix de noyaux
- Primitive de projection de points
- Noyau de lissage mixte diffusion-descente
- Résultats numériques

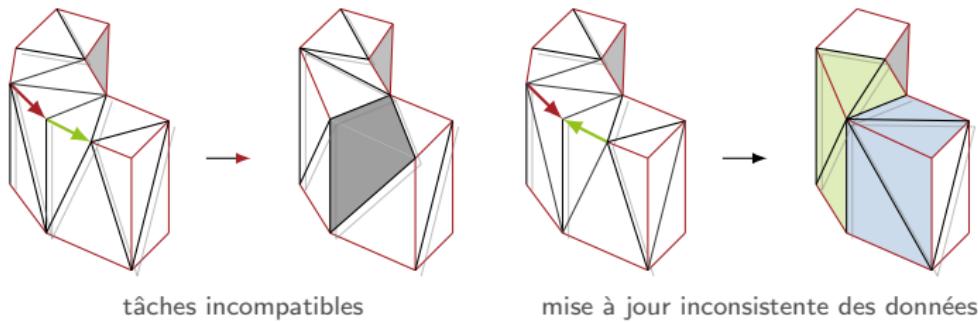
2 Portage des noyaux sur accélérateurs manycore

- Rappels de nos objectifs
- Problématique d'irrégularité et approche proposée
- Extraction du parallélisme à grain fin
- Synchronisation pour les primitives topologiques
- Résultats numériques

3 Conclusion

But : paralléliser chaque noyau.

- ✓ mémoire partagée : on va utiliser des threads,
- ✗ problème : compétition des threads.



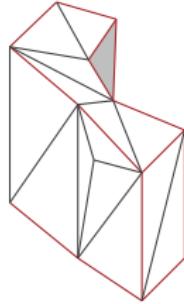
1 data-driven : dépendances de tâches non prédictibles,

- ✗ nombre de tâches et charge par core dynamiques,
- ✗ les tâches générées peuvent invalider les tâches déjà prévues.

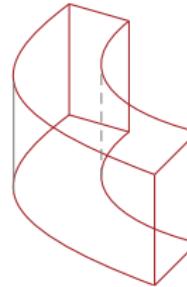
2 data-intensive : faible réutilisation de données,

- ✗ pénalités élevées dues aux indirections mémoire.

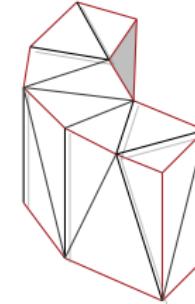
maillage initial



surface idéale

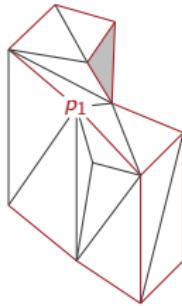


maillage final



- data-driven : dépendances de tâches non prédictibles,
 - ✗ nombre de tâches et charge par core dynamiques,
 - ✗ les tâches générées peuvent invalider les tâches déjà prévues.
- data-intensive : faible réutilisation de données,
 - ✗ pénalités élevées dues aux indirections mémoire.

maillage



graphe de calcul

$$G = (V, E, \omega)$$

move[p_1]
 $\omega = 4$

V : tâches à traiter,
 E : dépendances de tâches,
 ω : durées des tâches.



les tâches et leurs durées varient dynamiquement

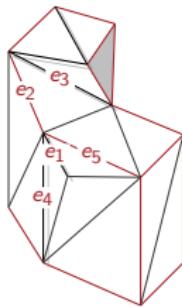
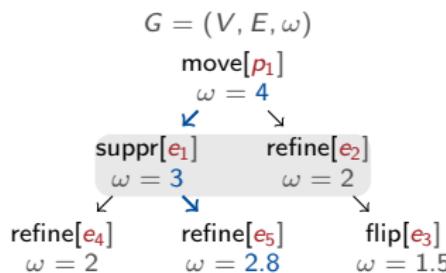


1 data-driven : dépendances de tâches non prédictibles,

- ✗ nombre de tâches et charge par core dynamiques,
- ✗ les tâches générées peuvent invalider les tâches déjà prévues.

2 data-intensive : faible réutilisation de données,

- ✗ pénalités élevées dues aux indirections mémoire.

maillagegraphe de calcul V : tâches à traiter, E : dépendances de tâches, ω : durées des tâches.

les tâches et leurs durées varient dynamiquement

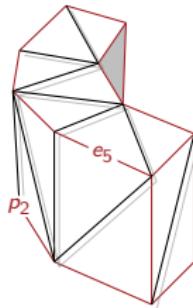
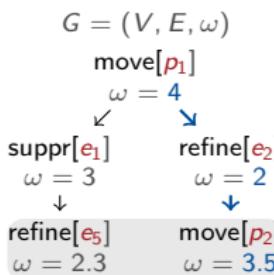


1 data-driven : dépendances de tâches non prédictibles,

- ✗ nombre de tâches et charge par core dynamiques,
- ✗ les tâches générées peuvent invalider les tâches déjà prévues.

2 data-intensive : faible réutilisation de données,

- ✗ pénalités élevées dues aux indirections mémoire.

maillagegraphe de calcul

V : tâches à traiter,

E : dépendances de tâches,

ω : durées des tâches.



les tâches et leurs durées varient dynamiquement

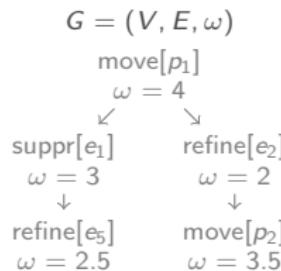


- 1 data-driven : dépendances de tâches non prédictibles,
 - ✗ nombre de tâches et charge par core dynamiques,
 - ✗ les tâches générées peuvent invalider les tâches déjà prévues.
- 2 data-intensive : faible réutilisation de données,
 - ✗ pénalités élevées dues aux indirections mémoire.

maillage



graphe de calcul



V : tâches à traiter,

E : dépendances de tâches,

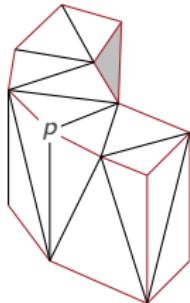
ω : durées des tâches.



les tâches et leurs durées varient dynamiquement

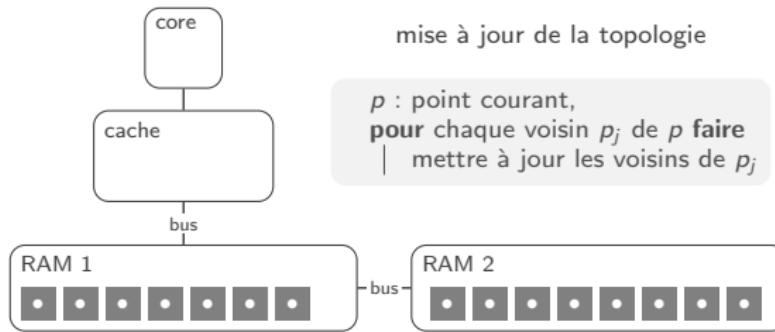


- 1 data-driven : dépendances de tâches non prédictibles,
 - ✗ nombre de tâches et charge par core dynamiques,
 - ✗ les tâches générées peuvent invalider les tâches déjà prévues.
- 2 data-intensive : faible réutilisation de données,
 - ✗ pénalités élevées dues aux indirections mémoire.

maillagelayout mémoire

mise à jour de la topologie

p : point courant,
pour chaque voisin p_j de p faire
| mettre à jour les voisins de p_j

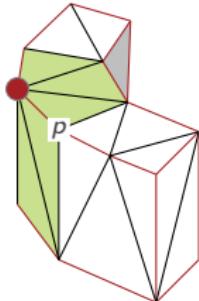


faible réutilisation de données en cache



- 1 data-driven : dépendances de tâches non prédictibles,
 - ✗ nombre de tâches et charge par core dynamiques,
 - ✗ les tâches générées peuvent invalider les tâches déjà prévues.
- 2 data-intensive : faible réutilisation de données,
 - ✗ pénalités élevées dues aux indirections mémoire.

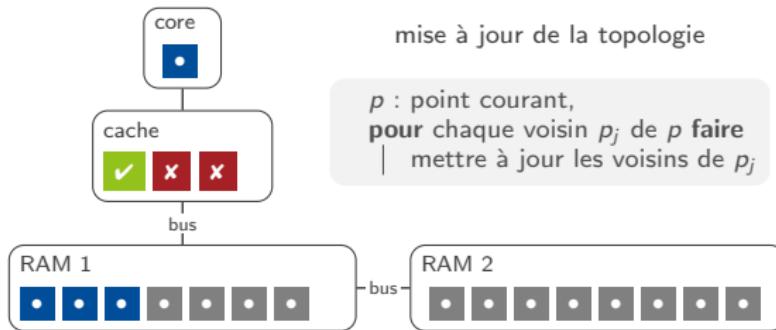
maillage



layout mémoire

mise à jour de la topologie

p : point courant,
pour chaque voisin p_j de p faire
| mettre à jour les voisins de p_j

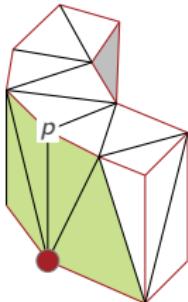


faible réutilisation de données en cache



- 1 data-driven : dépendances de tâches non prédictibles,
 - ✗ nombre de tâches et charge par core dynamiques,
 - ✗ les tâches générées peuvent invalider les tâches déjà prévues.
- 2 data-intensive : faible réutilisation de données,
 - ✗ pénalités élevées dues aux indirections mémoire.

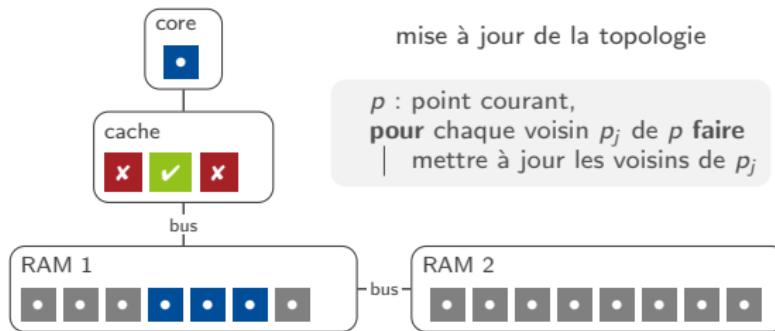
maillage



layout mémoire

mise à jour de la topologie

p : point courant,
pour chaque voisin p_j de p faire
| mettre à jour les voisins de p_j

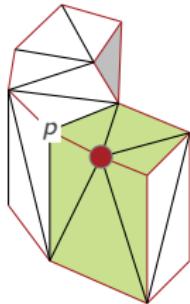


faible réutilisation de données en cache



- 1 data-driven : dépendances de tâches non prédictibles,
 - ✗ nombre de tâches et charge par core dynamiques,
 - ✗ les tâches générées peuvent invalider les tâches déjà prévues.
- 2 data-intensive : faible réutilisation de données,
 - ✗ pénalités élevées dues aux indirections mémoire.

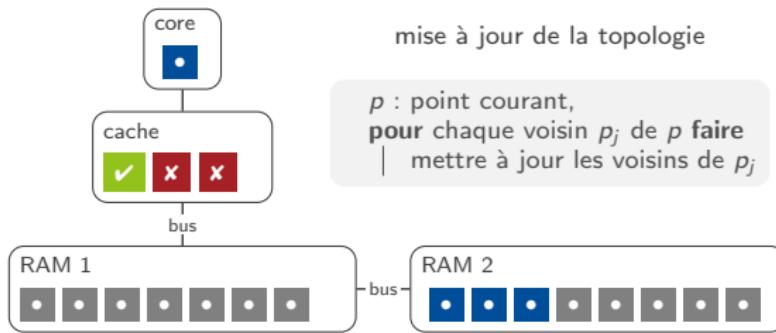
maillage



layout mémoire

mise à jour de la topologie

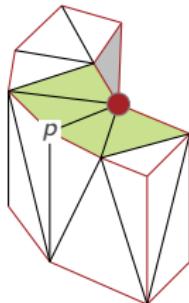
p : point courant,
pour chaque voisin p_j de p faire
| mettre à jour les voisins de p_j



faible réutilisation de données en cache

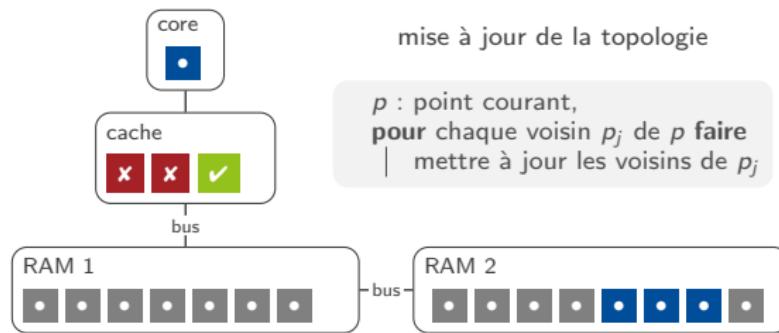


- 1 data-driven : dépendances de tâches non prédictibles,
 - ✗ nombre de tâches et charge par core dynamiques,
 - ✗ les tâches générées peuvent invalider les tâches déjà prévues.
- 2 data-intensive : faible réutilisation de données,
 - ✗ pénalités élevées dues aux indirections mémoire.

maillagelayout mémoire

mise à jour de la topologie

p : point courant,
pour chaque voisin p_j de p faire
| mettre à jour les voisins de p_j



faible réutilisation de données en cache



Pour l'aspect **data-driven** : [LOCK-FREE]

- 1 extraction du parallélisme pour chaque noyau,**
 - formuler les dépendances de données en graphe,
 - extraire les tâches par des heuristiques adéquats.
 - bon compromis entre nombre de tâches extraites et surcoût induit.
- 2 approche de structuration des noyaux,**
 - par vagues : calcul local, écritures groupées en mémoire partagée.
 - minimisation des contentions, portabilité.

Pour l'aspect **data-intensive** : [LOCK-FREE]

- 1 réduction asynchrone ou NUMA-aware de données.**
 - précalcul de références mémoire pour un accès direct ultérieur.
 - pas de recopie de données locales vers la mémoire partagée,
 - préserve la proximité mémoire de données géométriquement voisines.
- 2 synchronisation grain fin pour les primitives topologiques.**
 - mise à jour en deux temps du graphe d'incidence.
 - tâches plus locales, minimise les mouvements de données.

Pour l'aspect **data-driven** : [LOCK-FREE]

- 1 extraction du parallélisme pour chaque noyau,**
 - formuler les dépendances de données en graphe,
 - extraire les tâches par des heuristiques adéquats.
 - bon compromis entre nombre de tâches extraites et surcoût induit.
- 2 approche de structuration des noyaux,**
 - par vagues : calcul local, écritures groupées en mémoire partagée.
 - minimisation des contentions, portabilité.

Pour l'aspect **data-intensive** : [LOCK-FREE]

- 1 réduction asynchrone ou NUMA-aware de données.**
 - précalcul de références mémoire pour un accès direct ultérieur.
 - pas de recopie de données locales vers la mémoire partagée,
 - préserve la proximité mémoire de données géométriquement voisines.
- 2 synchronisation grain fin pour les primitives topologiques.**
 - mise à jour en deux temps du graphe d'incidence.
 - tâches plus locales, minimise les mouvements de données.

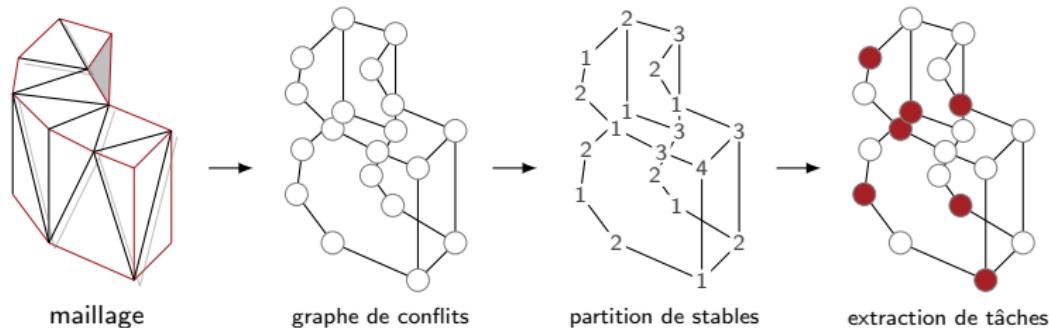
Pour l'aspect data-driven : [LOCK-FREE]

- 1 extraction du parallélisme pour chaque noyau,
 - formuler les dépendances de données en graphe,
 - extraire les tâches par des heuristiques adéquats.
 - ✓ bon compromis entre nombre de tâches extraites et surcoût induit.
- 2 approche de structuration des noyaux,
 - par vagues : calcul local, écritures groupées en mémoire partagée.
 - ✓ minimisation des contentions, portabilité.

Pour l'aspect data-intensive : [LOCK-FREE]

- 1 réduction asynchrone ou NUMA-aware de données.
 - précalcul de références mémoire pour un accès direct ultérieur.
 - ✓ pas de recopie de données locales vers la mémoire partagée,
 - ✓ préserve la proximité mémoire de données géométriquement voisines.
- 2 synchronisation grain fin pour les primitives topologiques.
 - mise à jour en deux temps du graphe d'incidence.
 - ✓ tâches plus locales, minimise les mouvements de données.

Idée : extraire un **stable** maximal de points en **parallèle**.



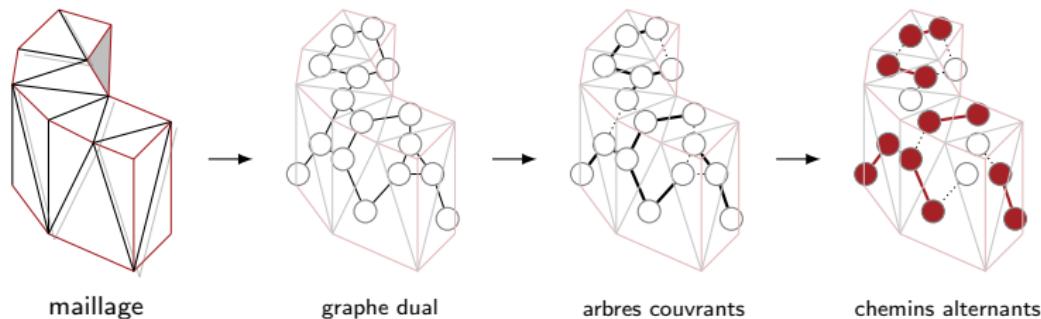
Contraintes :

- ✗ intrinsèquement séquentiel,
- ✗ coût doit rester négligeable.

Notre approche spéculative :

- ✓ bon compromis entre nombre de tâches extraites et coût,
- ✓ tient compte de l'évolution du graphe.

Idée : extraire un **couplage maximal de mailles en parallèle**.



Contraintes :

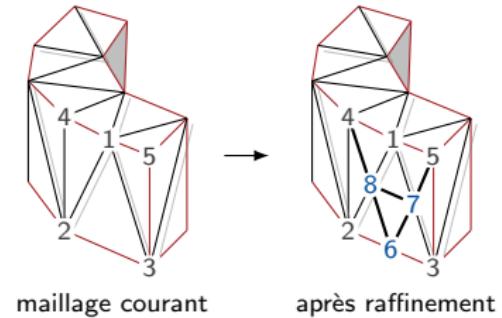
- ✗ très sensible à l'ordonnancement,
- ✗ coût doit rester négligeable.

Notre approche :

- ✓ bon compromis entre nombre de tâches extraites et coût,
- ✓ synchronisation à grain fin pour atténuer le déséquilibre.

Mise à jour en deux temps :

- 1 insérer les références,
 - trouver l'offset de chaque liste en incrémentant le degré du point,
 - marquer les listes à réparer.
- 2 épurer les listes d'incidence marquées.



maillage courant

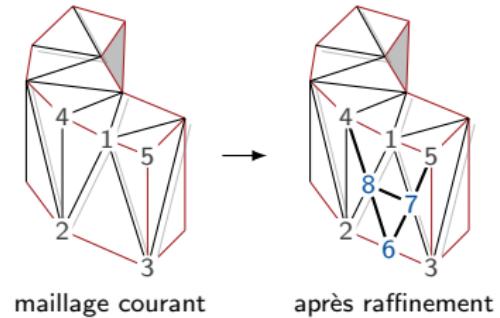
après raffinement

ÉTAT INITIAL				
<u><i>i</i></u>	<u><i>deg</i></u>	<u><i>fix</i></u>	<u><i>voisins</i></u>	
1	4	0	[2,3,4,5,7,8]	
2	3	0	[1,3,4,6,8]	

avant raffinement.

Mise à jour en deux temps :

- 1 insérer les références,
 - trouver l'offset de chaque liste en incrémentant le degré du point,
 - marquer les listes à réparer.
- 2 épurer les listes d'incidence marquées.



maillage courant

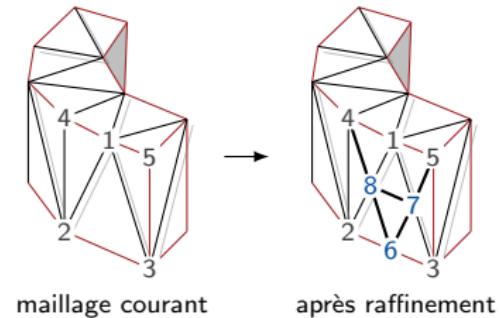
après raffinement

INSERTIONS				
<u><i>i</i></u>	<u><i>deg</i></u>	<u><i>fix</i></u>	<u><i>voisins</i></u>	
1	6	1	[2,3,4,5,7,8]	
2	5	1	[1,3,4,6,8]	

liste mise à jour par *k* threads.

Mise à jour en deux temps :

- 1** insérer les références,
 - trouver l'offset de chaque liste en incrémentant le degré du point,
 - marquer les listes à réparer.
- 2** épurer les listes d'incidence marquées.



maillage courant

après raffinement

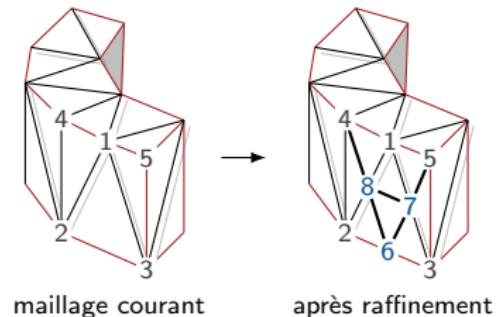
ÉPURATION				
<u>i</u>	<u>deg</u>	<u>fix</u>	<u>voisins</u>	
1	4	0	[2,3,4,5,7,8]	
2	3	0	[1,3,4,6,8]	

liste réparée par un thread.

Mise à jour en deux temps :

- 1 insérer les références,
 - trouver l'offset de chaque liste en incrémentant le degré du point,
 - marquer les listes à réparer.
- 2 épurer les listes d'incidence marquées.

- ✓ primitives bas-niveau peu couteuses,
 - deg : `fetch_add`,
 - fix : `compare_swap`.
- ✓ pas de transferts inutiles de données.



maillage courant

après raffinement

ÉTAT FINAL				
<u>i</u>	<u>deg</u>	<u>fix</u>	<u>voisins</u>	
1	4	0	[2,3,4,5,7,8]	
2	3	0	[1,3,4,6,8]	

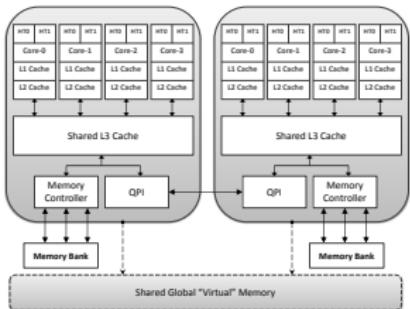
liste réparée par un thread.

En résumé, nous avons proposé une manière :

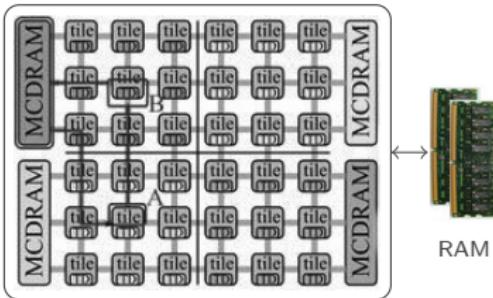
- ① d'extraire les tâches parallèles à l'aide de graphes,
- ② de synchroniser les threads pour la mise à jour de la topologie.

Nous allons maintenant évaluer leur efficacité une fois combinés.

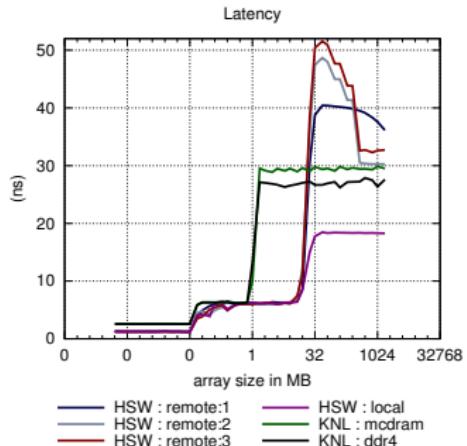
Intel Haswell et Skylake [HSW,SKL]



Intel Knights Landing [KNL]



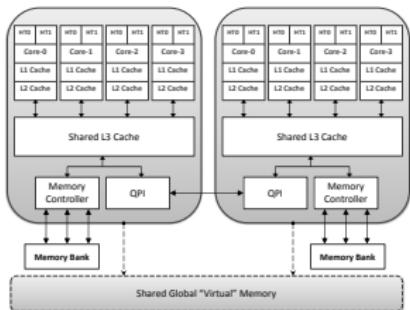
archi	puces	NUMA	cores	GHz	HT	RAM/core
HSW	2	4	32	2.5	2	4.0 GB
SKL	2	2	48	2.7	2	7.9 GB
KNL	1	4	68	1.4	4	1.5 GB



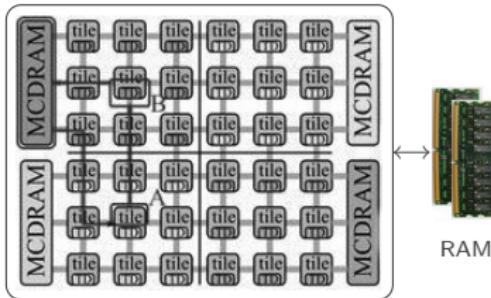
Instances de maillages :

- 1 engine : 1 826 000 points et 3 652 058 mailles,
- 2 shock : 1 014 890 points et 2 029 772 mailles.

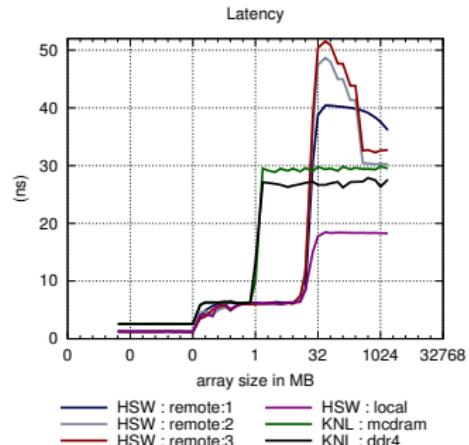
Intel Haswell et Skylake [HSW,SKL]



Intel Knights Landing [KNL]



archi	puces	NUMA	cores	GHz	HT	RAM/core
HSW	2	4	32	2.5	2	4.0 GB
SKL	2	2	48	2.7	2	7.9 GB
KNL	1	4	68	1.4	4	1.5 GB

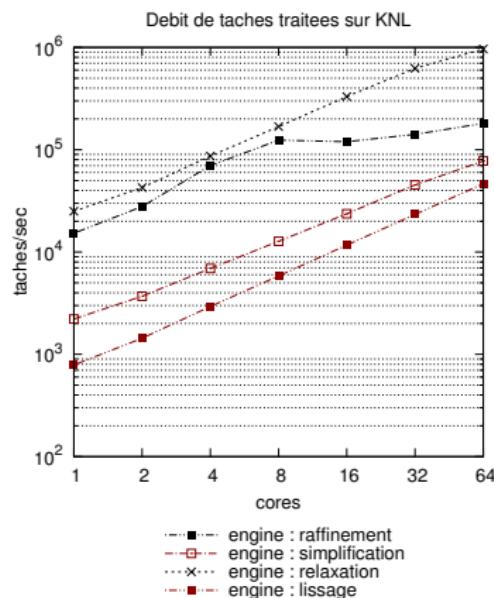
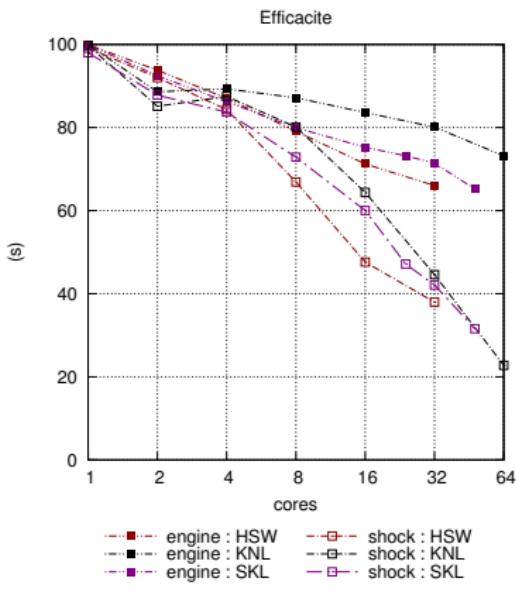


Instances de maillages :

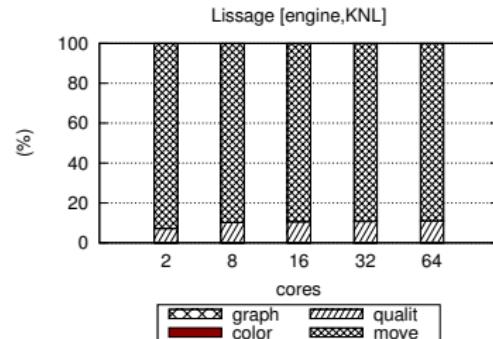
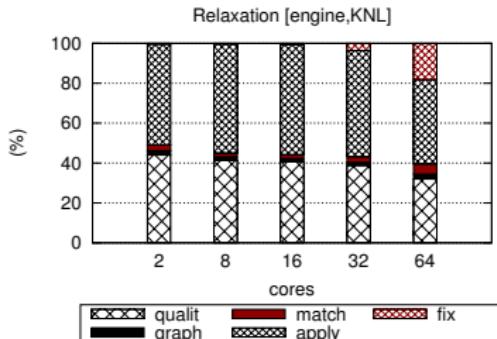
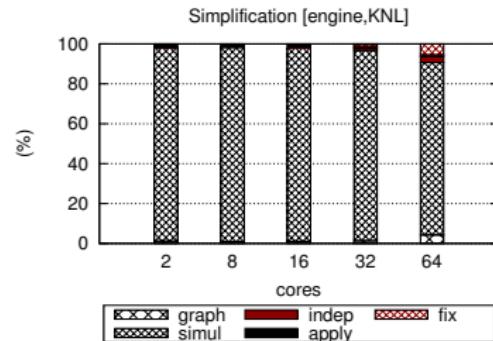
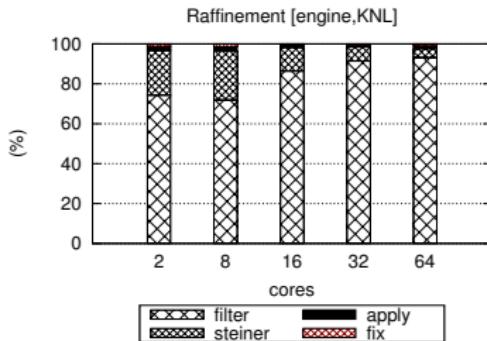
- 1 engine : 1 826 000 points et 3 652 058 mailles,
- 2 shock : 1 014 890 points et 2 029 772 mailles.

Évolution de l'efficacité $E_p = \frac{T_1}{pT_p}$ et débit de tâches traitées,

- 1 deux cas-tests [engine, shock], trois machines [HSW, SKL, KNL].
- 2 instances de tailles fixes, augmentation du nombre de threads.



Répartition du temps de restitution par phase pour chaque noyau sur KNL.



1 Étude de noyaux surfaciques locality-aware

- Rappels de nos objectifs
- Problématique de localité et choix de noyaux
- Primitive de projection de points
- Noyau de lissage mixte diffusion-descente
- Résultats numériques

2 Portage des noyaux sur accélérateurs manycore

- Rappels de nos objectifs
- Problématique d'irrégularité et approche proposée
- Extraction du parallélisme à grain fin
- Synchronisation pour les primitives topologiques
- Résultats numériques

3 Conclusion

En résumé, nous avons proposé :

- 1 des noyaux locaux efficients pour l'adaptation de surfaces triangulées ;
- 2 un portage efficient de noyaux sur machines NUMA et manycore ;
- 3 et une bibliothèque **trigen** écrite en C++ ayant impliqué :
 - x lignes de code en 2D, et y en 3D ;
 - le déploiement et profiling sur deux types d'architectures ;
 - l'implém. d'heuristiques de graphes [CFG+12,GM00,RGK15] pour une étude comparative ;
 - l'intégration du schéma de synchronisation de PRAGMATIC [RG13] pour comparaison.
 - la génération de cas-tests et fine-tuning : matrices RMAT [CZF04], modèles de CAO ;
 - le post-traitement semi-automatisé pour la sortie de résultats ;
 - et le plus fun : le débugging 😊

[CFG+12] Catalyurek et al. Graph Colouring Algorithms for Multi-core and Massively Multithreaded Architectures. JPC, 2012.

[GM00] Gebremedhin and Manne. Scalable parallel graph coloring algorithms. Concurrency : Practice and Experience, 2000.

[RGK15] Rokos et al. A Fast and Scalable Graph Coloring Algorithm for Multicore and Manycore Architectures. Euro-Par, 2015.

[RG13] Rokos and Gorman. Pragmatic - Parallel anisotropic adaptive mesh toolkit. Facing the Multicore-Challenge III, 2013.

[CZF04] Chakrabarti, Zhan and Faloutsos. R-MAT : A recursive model for graph mining. SDM, 2004.

En résumé, nous avons proposé :

- 1 des noyaux locaux efficients pour l'adaptation de surfaces triangulées ;
- 2 un portage efficient de noyaux sur machines NUMA et manycore ;
- 3 et une bibliothèque **trigen** écrite en C++ ayant impliqué :
 - x lignes de code en 2D, et y en 3D ;
 - le déploiement et profiling sur deux types d'architectures ;
 - l'implém. d'heuristiques de graphes [CFG+12,GM00,RGK15] pour une étude comparative ;
 - l'intégration du schéma de synchronisation de PRAGMATIC [RG13] pour comparaison.
 - la génération de cas-tests et fine-tuning : matrices RMAT [CZF04], modèles de CAO ;
 - le post-traitement semi-automatisé pour la sortie de résultats ;
 - et le plus fun : le débugging 😊

[CFG+12] Catalyurek et al. Graph Colouring Algorithms for Multi-core and Massively Multithreaded Architectures. JPC, 2012.

[GM00] Gebremedhin and Manne. Scalable parallel graph coloring algorithms. Concurrency : Practice and Experience, 2000.

[RGK15] Rokos et al. A Fast and Scalable Graph Coloring Algorithm for Multicore and Manycore Architectures. Euro-Par, 2015.

[RG13] Rokos and Gorman. Pragmatic - Parallel anisotropic adaptive mesh toolkit. Facing the Multicore-Challenge III, 2013.

[CZF04] Chakrabarti, Zhan and Faloutsos. R-MAT : A recursive model for graph mining, SDM, 2004.

Publications dans des actes de conférences avec comité de lecture :

- 1 [RLP16] Hoby Rakotoarivelo, Franck Ledoux and Franck Pommereau.
Fine-grained locality-aware parallel scheme for anisotropic mesh adaptation.
25th International Meshing Roundtable, 2016, Washington DC, USA. [rang B]
- 2 [RLP+17] Hoby Rakotoarivelo, Franck Ledoux, Franck Pommereau and Nicolas Le-Goff.
Scalable fine-grained metric-based remeshing algorithms for manycore and NUMA
architectures. 23rd International Conference on Parallel and Distributed Computing [Euro-Par],
2017, Santiago de Compostella, Spain. [rang A, 30% d'acceptation]
- 3 [RL18] Hoby Rakotoarivelo and Franck Ledoux.
Accurate manycore-accelerated manifold surface remesh kernels.
27th International Meshing Roundtable, 2018, Albuquerque NM, USA. [réponse mi-juillet]

Communications orales :

- 1 [LR18] Franck Ledoux and Hoby Rakotoarivelo.
Accurate manycore-accelerated anisotropic surface remeshing, MeshTrends.
13th World Congress in Computational Mechanics, 2018, New-York NY, USA.

À court-terme (6 mois) :

1 Finalisation de trigen,

- inclus dans mon travail au CMLA, ENS Cachan.
- sera open-source.

2 Vérifier formellement les briques critiques :

- heuristiques pour l'extraction des tâches,
- synchronisations basées sur les primitives atomiques.
- vérifier le caractère wait-free (par un modèle en réseau de Petri).

À moyen-terme (1 an et demi) :

1 Analyse approfondie des algorithmes :

- relaxation : vers un noyau avec garantie de convergence ?
- lissage : intégrer les contraintes de projection dans la formulation du problème.
- ratio d'approximation des algorithmes de graphes.

2 Étendre au parallélisme à granularité multiple :

- machines à mémoire à la fois partagée et distribuée,
- restructurer les noyaux en tenant compte de la hiérarchie mémoire.
- rééquilibrage dynamique et NUMA-aware de charges.

À court-terme (6 mois) :

1 Finalisation de trigen,

- inclus dans mon travail au CMLA, ENS Cachan.
- sera open-source.

2 Vérifier formellement les briques critiques :

- heuristiques pour l'extraction des tâches,
- synchronisations basées sur les primitives atomiques.
- vérifier le caractère wait-free (par un modèle en réseau de Petri).

À moyen-terme (1 an et demi) :

1 Analyse approfondie des algorithmes :

- relaxation : vers un noyau avec garantie de convergence ?
- lissage : intégrer les contraintes de projection dans la formulation du problème.
- ratio d'approximation des algorithmes de graphes.

2 Étendre au parallélisme à granularité multiple :

- machines à mémoire à la fois partagée et distribuée,
- restructurer les noyaux en tenant compte de la hiérarchie mémoire.
- rééquilibrage dynamique et NUMA-aware de charges.

A plus long terme :

1 Réflexions sur l'extensibilité de l'approche :

- autres problèmes irréguliers : machine-learning, n -corps etc.
- ils n'ont pas tous les mêmes caractéristiques,
- définir des classes d'équivalences,
- proposer une méthodologie par classe.

2 Retour à l'analyse formelle d'applications industrielles :

- parallélisme massif : codes de production plus difficiles à vérifier,
- problèmes de la montée en charge des algorithmes de vérification,
- quantifier le lien entre degré d'abstraction et interprétabilité des propriétés.

In fine, nous avons voulu montrer :

- 1 l'approche de co-design numérique-parallèle sur un cas concret ;
 - penser à la parallélisation dès le choix même d'un noyau [HPA+15].
 - (a) nouvelle application sur machine existante
 - (b) application existante sur nouvelle machine

numérique

$$\begin{aligned} A &= Ax^2 \\ A' &= x^2 \\ A' &= x^2 \\ A' &= \frac{x^2}{x^2} - 12 & 6L^2 & -12 & 6L \\ & 6L & 2L^2 & 12 & 2L^2 \\ & -12 & 6L^2 & -6L^2 & -6L^2 \\ A &= x^2 \\ x &= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \end{aligned}$$

$$x = \sum_{k=0}^{n-1} \binom{n}{k} x_k \alpha^{n-k} \quad x = x_0 + x_1 \alpha + \dots$$

algorithme parallèle

```

19 unsigned int levolve(
20     const size_t len1 = s1.size() / s1.stride(),
21     const size_t len2 = col1.size() / col1.stride(),
22     vector<unsigned int> &col1,
23     for (unsigned int i = 0; i < prevCol.size(); ++i)
24     prevCol[i] = i;
25     for (unsigned int i = 0; i < len1; ++i) {
26         for (unsigned int j = 0; j < len2; ++j) {
27             col1[i] = (unsigned int) std::min((prevCol[i] + j) * s1.stride() +
28                                             prevCol[j], (prevCol[i] + j + 1) * s1.stride());
29         }
30     }
31     return prevCol[0];
32 }
```

architecture



↔ (a)

↔ (b)

- 2 le changement de paradigme requis par l'exascale [BBD+13].
 - si cela se fait, ce sera aussi ardue pour les ordinateurs quantiques.

[BBD+13] Barrett et al. On the role of co-design in HPC. Transition of HPC Towards Exascale Computing, 2013.

[HPA+15] Halappanavar et al. Co-design lessons learned from implementing graph matching on multithreaded architectures. Computer Journal, 2015.


Commissariat à l'énergie atomique et aux énergies alternatives
Centre DAM-Île-de-France | 91297 Arpajon Cedex
T. +33 (0)1 69 26 40 00

Établissement public à caractère industriel et commercial | RCS Paris B 775 685 019