



UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

École Doctorale Sciences Technologie Santé

Rapport français

Pour obtenir le grade de:

Docteur de l'Université de Reims Champagne-Ardenne

Discipline : Informatique

Spécialité : Calcul Haute Performance

Présentée et soutenue par:

Julien LOISEAU

le 18 Mars 2018

Les Architectures Hybrides pour Atteindre l'Exascale

Sous la direction de :

Michaël KRAJECKI, Professeur des Universités

JURY

Pr. Michaël Krajecki	CRéSTIC, Université de Reims Champagne-Ardenne	Directeur
Dr. François Alin	CRéSTIC, Université de Reims Champagne-Ardenne	Co-directeur
Dr. Christophe Jaillet	CRéSTIC, Université de Reims Champagne-Ardenne	Co-directeur
Pr. William Jalby	Université de Versailles Saint-Quentin	Rapporteur
Dr. Benjamin Bergen	Ingénieur au Los Alamos National Laboratory, USA	Rapporteur
Pr. Zineb Habbas	LCOMS, Université de Lorraine	Rapporteur
Pr. Françoise Baude	I3S, Université de Nice Sophia Antipolis	Examineur
Guillaume Colin de Verdier	Ingénieur au CEA	Invité

Ce document comprend un résumé substantiel de ma thèse rédigée en anglais. Il présente tout d'abord la traduction de la table des matières de la thèse. Une traduction de l'introduction générale, une synthèse des trois parties de la thèse ainsi que de la conclusion générale.

Table des matières traduite en français

Cette traduction de la table des matières a été réduite aux Parties, Chapitres et Sections de la thèse.

Introduction

I Exascale et calcul haute performance

1 Modèles pour le HPC

- 1.1 Introduction
- 1.2 Modèle de Von Neumann
- 1.3 Taxonomie de Flynn et modèles d'exécution
- 1.4 Mémoires
- 1.5 Caractérisation des performances dans le HPC
- 1.6 Conclusion

2 Hardware dans le HPC

- 2.1 Introduction
- 2.2 Évolution du modèle de Von Neumann
- 2.3 Architectures du 21ème siècle
- 2.4 Architectures distribuées architectures
- 2.5 Le supercalculateur ROMEO .
- 2.6 Conclusion

3 Software dans le HPC

- 3.1 Introduction
- 3.2 Modèles parallèles et distribués
- 3.3 Software/API
- 3.4 Benchmarks
- 3.5 Conclusion

II Métrique de problème complexes

4 Calcul : Le problème de Langford

- 4.1 Introduction
- 4.2 Algorithme de Miller
- 4.3 Méthode algébrique de Godfrey
- 4.4 Conclusion

5 Communication : Graph500

- 5.1 Introduction
- 5.2 Méthodes existantes
- 5.3 Environnement
- 5.4 Parcours en largeur
- 5.5 Résultats
- 5.6 Conclusions

III Application

6 Modélisation et systèmes complexes

- 6.1 Introduction
- 6.2 Modélisation physique et limitations
- 6.3 Cas appliqués
- 6.4 Conclusion

7 Simulations complexes sur architecture hybride

- 7.1 Introduction
- 7.2 FleCSI
- 7.3 SPH distribué sur architecture multi-core
- 7.4 SPH distribué sur architecture hybride
- 7.5 Résultats
- 7.6 Conclusion

Conclusion

Bibliographie

Introduction

Le monde du calcul haute performance (HPC) va prochainement atteindre une puissance de calcul inégalée avec l'échelle exaflopique. Les États Unis d'Amérique et l'Europe devraient l'atteindre aux horizons 2020-2021 mais la Chine pourrait proposer une telle machine dès 2019. Ces superordinateurs seront 100 fois plus rapides que l'estimation actuelle des performances du cerveau humain avec 10^{16} calculs à virgules flottante par secondes (FLOPS)[?, kurzweil2010singularity]t atteindre une puissance sans précédent d'un milliard de milliard (10^{18}) de FLOPS. Cette aventure a commencé avec les premiers ordinateurs à tube à vides et les besoins de la balistique militaire. De nos jours les supercalculateurs étendent leur domaine d'application et sont des éléments phares pour représenter la puissance d'une nation. Les applications se sont répandues dans tous les secteurs de la science et de la technologie.

Depuis 1962, et en considérant le Cray CDC 6600 tel que le premier superordinateur, la puissance de calcul de ces machines n'a fait qu'augmenter tout en suivant une observation faite par le co-fondateur de l'entreprise Intel, Gordon Moore. Mieux connue sous le nom de "Loi de Moore", cette loi de 1965 explique que, considérant l'évolution constante de la technologie, le nombre de transistors sur un circuit intégré va doubler environs tous les deux ans pour un prix similaire. Ceci va permettre une augmentation de la puissance de calcul que peuvent fournir les ordinateurs et superordinateurs. Plus important encore, comme "*L'argent est le nerf de la guerre*", le prix des puces pour de meilleures performances va diminuer.

Cette théorie de Gordon Moore peut être observée avec l'évolution de la puissance des supercalculateurs, la liste TOP500¹. Présenté sur la figure 1, le loi de Moore se poursuit et reste vraie après des décennies d'évolution du matériel informatique.

La diminution de la taille des semi-conducteurs avec des transistors de plus en plus petits n'est pas la seule raison de cette évolution linéaire. Les premiers processeurs étaient bâtis autour d'un unique cœur de calcul (CPU) avec une augmentation du nombre de transistors et une meilleure fréquence d'opération. Ils ont vite rencontré une limitation pour atteindre des fréquences toujours plus élevées du fait de l'énergie requise mais aussi de la chaleur générée devant être canalisée. C'est pourquoi, au début du vingtième siècle, IBM proposa le premier processeur multi-cœur, le Power4. Les constructeurs ont commencé à créer des cartes avec plus d'un cœur pour augmenter la puissance de calcul tout en continuant à réduire la taille des composants. Cela permis de répondre à la constante demande en puissance de calcul et la Loi de Moore fut conservée. Bien entendu cette nouvelle architecture demandait de nouveaux paradigmes et se confronta plusieurs limitations. Un coût supplémentaire en synchronisation entre les cœurs pour l'accès à la mémoire, le partage des tâches mais aussi une complexité des algorithmes parallèles. De nos jours un CPU classique propose de deux à des dizaines de cœurs de calcul sur une seule carte.

Pour atteindre une puissance de calcul encore plus importante et dépasser les contraintes énergétiques, les chercheurs se sont intéressés aux approches many-cores. Ces machines utilisent des centaines, voire des milliers, de cœurs de calcul "simples". Leur fréquence de fonctionnement est plus basse ce qui réduit la consommation mais ils s'exécutent de manière synchrone. Ce synchronisme d'exécution ajouter une complexité lors du développement pour pouvoir atteindre les meilleures performances. Ces architectures

¹<https://www.top500.org>

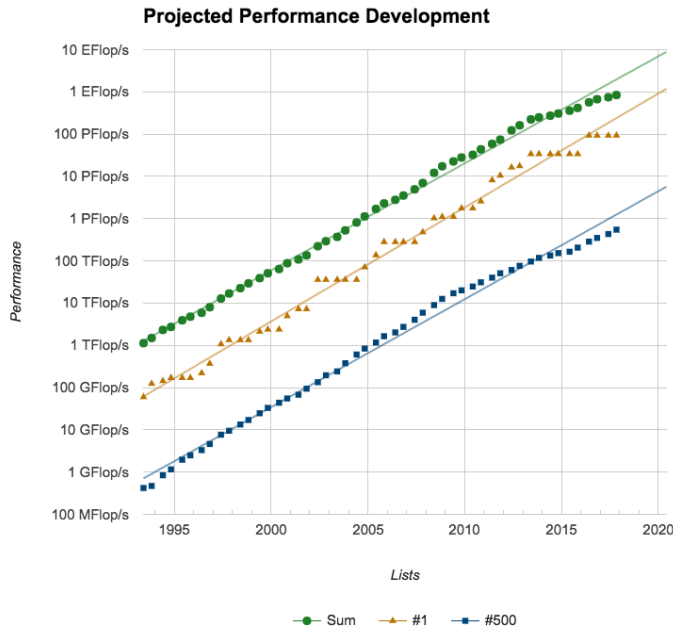


Figure 1: Évolution de la puissance de calcul, liste du TOP500

sont généralement couplées à un CPU qui gère l’envoi des données et les exécutions de fonctions appelées ”Kernels”. On notera que certaines architectures se placent entre les multi-core et many-core, proposant près de cent cœurs de calcul utilisant un réseau de processeurs multi-core, le Xeon Phi d’Intel. Les architecture many-cores sont généralement appelées accélérateurs, ils augmentent sensiblement la puissance de calcul du CPU auquel il est associé. Les plus connus sont donc les General-Purpose Graphics Processing Unit (GPGPU), les Xeon Phi, les Field Programmable Gate Array (FPGA) ou encore les Application-Specific Integrated Circuit (ASIC). L’architecture basée sur un processeur ”Host” avec un ou plusieurs ”Device” apparait et est appelé architecture hybride. Un supercalculateur peut être classique, avec uniquement des processeurs multi-cœurs, hybride avec des accélérateur du même type ou encore hétérogène avec des nœuds présentant différents types de d’accélérateurs.

Depuis 2013 et 2014, beaucoup d’entreprises, comme celle de Gordon Moore avec Intel, pensent que la loi de Moore est terminée. La figure 1 présente l’évolution de la puissance des supercalculateurs. On peut en effet voir sur la partie droite du graphe que l’évolution n’est plus linéaire mais tend à décroître doucement dans le temps. Ceci peut s’expliquer par deux facteurs. Tout d’abord la taille minimale des transistors est peu à peu atteinte et induit des effets de bord difficiles à gérés. De plus, l’augmentation de la surface de silicium, le grand nombre de transistors et les fréquences élevées augmentent sensiblement la consommation énergétique.

Malgré ces dernières technologies, les supercalculateurs actuels sont confrontés à plusieurs limitations dans leur conception et utilisation. Les trois principales limitations qui limitent la puissance de calcul sont : la consommation énergétique, la communication et la mémoire. On trouve plusieurs sous-problématiques avec le réseau d’interconnexion, la résilience et la complexité d’utilisation de ces machines contenant des milliards de com-

posants de calcul.

Dans cette période de questionnement par rapport aux futures architectures du calcul haute performance, cette étude propose plusieurs points de vue. Nous pensons que le futur du calcul haute performance sera bâti autour des architectures hybrides et des accélérateurs utilisant des API et Framework adaptés. Nous considérons que les benchmarks classiques, tel que le fameux TOP500, n'est pas représentatif des problématiques actuelles et des défis du calcul haute performance. Les applications des scientifiques tels que la physique, l'astrophysique, la chimie ou encore la biologie nécessitent des benchmarks basés sur des problèmes lourds en calcul, en communication mais surtout au comportement irrégulier.

Dans cette étude nous proposons notre métrique pour extraire les principales problématiques du calcul haute performance et de les appliquer aux architecture hybrides. Nous montrons comment tirer avantage de ces architectures malgré un groupe de problème qui ne se prête pas à leur modèle d'exécution. Notre métrique est bâtie en rassemblant deux problèmes caractéristiques puis un troisième problème, combinant tout notre les acquis précédents. Les deux premiers ciblent le calcul et la communication avec un comportement très irrégulier avec un grand nombre d'accès mémoire, en utilisant un problème combinatoire académique et le benchmark du Graph500. La dernière étape est un problème de calcul scientifique qui couvre l'ensemble des problématiques précédentes, considéré difficile à implémenter sur des architecture hybrides. Les résultats montrent le net avantage des architectures hybrides et confortent notre thèse.

Cette thèse se décompose en trois parties. La première partie explore l'état de l'art du calcul haute performance depuis les lois fondamentales jusqu'au machines actuelles. Nous présentons les lois d'Amdahl et de Gustafson ainsi que les notions d'accélération et d'efficacité. Les processeurs classiques, les GPGPU et les autres accélérateurs sont présentés en compléments des modèles d'exécution et de mémoire. Les principales méthodes de classement des supercalculateurs et les problématiques inhérentes sont détaillées. Nous montrons que ces classements ne sont pas suffisants pour représenter les problématiques actuelles.

Dans la seconde partie nous présentons notre métrique Le problème de Langford est décrit sous son aspect lourd en calcul et irrégulier. Ce premier élément montre l'avantage des accélérateurs, dans notre cas les GPGPU, sur ce type de problème ne mettant pas en jeu de communications. Ce travail a mené à la publication d'un article de journal[KLAJ16b] ainsi que plusieurs conférences, présentations et posters[DJK⁺14, LJA16, JDA⁺14]. Notre implémentation de ce problème nous a permis de battre un record en termes de temps de calcul sur les dernières instances.

Le problème du Graph500 est ensuite proposé pour cibler, cette fois, la problématique des communications conjointement à l'irrégularité. Nous présentons notre implémentation et la logique utilisée pour obtenir des résultats GPGPU bien supérieurs aux architectures classiques. Ce travail a mené à la publication d'un papier de conférence[KLAJ16a] ainsi que plusieurs présentations et posters[LJA15, LJA15].

Dans la troisième et dernière partie nous considérons un problème confrontant à la fois la puissance de calcul et la communication avec un comportement hautement irrégulier. Nous proposons une analyse de ce problème et montrons qu'il combine tous ces aspects. Nous détaillons ensuite notre méthode pour implémenter ce problème sur les architectures

modernes et en particulier hybrides. Notre choix, explicité dans l’analyse, s’est porté sur la méthode Smoothed Particle Hydrodynamics (SPH). Cette application commence avec le développement du framework FleCSI au Los Alamos National Laboratory (LANL) pour répondre à la demande des scientifiques des domaines appliqués. Le but est de mettre en place un outil efficace pour les physiciens et les astrophysiciens appelé FleCSPH. Ce travail a mené à la production de présentation ainsi que de posters[DBGH⁺16, LLMB17].

La dernière partie résume l’ensemble du travail et les résultats obtenus. Elle montre l’avantage certain des architectures hybrides sur l’ensemble des limitations abordées dans notre métrique.

1 Modèles du HPC

Cette partie présente l’état de l’art de la théorie et des technologies qui font le calcul haute performance d’aujourd’hui. Elle décrit les outils nécessaires à la compréhension de la problématique et de notre méthodologie scientifique. Le calcul haute performance ne trouve pas de définition stricte. Elle est définie en trois chapitres avec la théorie, les technologies et enfin les logiciels, API et framework pour les utiliser.

Le premier chapitre présente la machine de Von Neumann, base de toutes les machines actuelles, et des premiers processeurs mono-cœurs. Ils ont ensuite évolué avec une approche parallèle et une augmentation de la fréquence et du nombre de cœurs de calcul. Ceci nous permet d’introduire la taxonomie de Flynn et en particulier les modèles d’exécution SIMD et SIMT qui représentent les architectures massivement parallèles et en particulier les GPGPU. Les modèles de mémoires sont ensuite introduits avec le NUMA, CC-NUMA et NoRMA. Nous introduisons ensuite la notion de performance avec tout d’abord les FLOPS, opérations à virgule flottantes. Nous définissons enfin la scalabilité et l’efficacité en HPC. Cela nous amène aux lois de Amdahl et Gustafson permettant de définir les notions de *Strong* et *Weak scaling*.

Le second chapitre propose une présentation chronologique des technologies qui composent les supercalculateurs d’aujourd’hui. Nous présentons en détail les architectures multi-cœurs avec le calcul *out-of-order*, le *pre-fetching*, etc. Les architectures many-cores sont présentées avec les GPGPU de NVIDIA et AMD, les PEZY ainsi que les FPGA. Une fois les composants de base des supercalculateurs introduits nous présentons la topologie d’interconnexion et les principaux supercalculateurs au monde. Nous présentons enfin le supercalculateur sur lequel l’ensemble des tests de cette étude ont été réalisés, le méso-centre ROMEO de l’Université de Reims Champagne-Ardenne.

Le dernier chapitre de cette première partie nous permet de mettre en avant les logiciels, API et framework dédiés au HPC que nous avons utilisés pour mettre en place nos implémentations multi-CPU et multi-GPU. Nous faisons un lien entre les modèles parallèles et distribués comme PRAM, *Fork-Join* et BSP avec les outils de développements disponibles.

Tous ces éléments nous permettent de présenter les benchmarks actuels des supercalculateurs. Le plus connu est le TOP500, utilisé comme classement mondial des machines contenant les 500 plus puissants supercalculateurs publics au monde. Nous montrons que ce benchmark, et les autres, ne sont pas représentatif des besoins et des problèmes actuels rencontrés dans les problèmes de production avec une irrégularité sous-jacente non représentée dans ces benchmarks. IL faut donc mettre en place une métrique permettant de déterminer le comportement des architectures dans les conditions représentatives de calcul, communication et d’irrégularité.

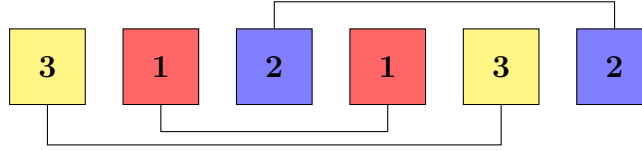


Figure 2: $L(2,3)$ arrangement

2 Métrique de problème complexes

Cette seconde partie détaille notre travail méthodologique et notre métrique, adaptée aux problématiques actuelles. Nous rappelons tout d'abord les différents murs auquel le calcul haute performance est confronté et nous décidons de cibler les deux principales limitations, le calcul et la communication tout en gardant des exemples représentatif de l'irrégularité à la fois de calcul, communication mais aussi de l'utilisation mémoire. Il convient de comparer sur ces problèmes les architectures classiques et hybrides à leur meilleur niveau d'optimisation.

2.1 Calcul: Problème de Langford

Pour cibler l'aspect de calcul sans communication et en environnement irrégulier nous avons choisis le problème académique de dénombrement combinatoire de Langford. Ce problème est présenté sous deux approches, l'une généralisable à l'ensemble des problématiques de *Constraint Satisfaction Problem* (CSP) et l'autre dédié à la résolution de ce problème.

- Une résolution arborescente, appelée méthode de Miller, nous permet de mettre en place un premier travail sur les parcours d'arbre, irréguliers par nature.
- La méthode algébrique, dite de Godfrey, nous montre l'avantage des architectures hybrides avec un comportement irrégulier induit par l'utilisation de grand entiers.

C. Dudley Langford a donné son nom à un problème combinatoire classique [Gar56, Sim83]. En observant son fils jouer avec des cubes de couleurs différentes, il a remarqué des arrangements spécifiques de trois paires de cubes (jaune, rouge et bleu) : la paire rouge - 1 - était séparée par un cube, la paire bleue - 2 - séparée par deux cubes et enfin la paire jaune - 3 - séparée par trois cubes, tel que représenté sur la figure 2.

Ce problème a ensuite été généralisé à un ensemble de n couleurs et un nombre s de cubes de la même couleur. $L(s, n)$ consiste en la recherche du nombre de solution au problème de Langford à une symétrie près.

Méthode de Miller

La méthode de Miller propose une résolution arborescente du problème de Langford en explorant l'ensemble des cas possibles. Les branches impossibles de l'arbre sont éliminées au fur et à mesure du parcours. Avec l'explosion combinatoire le nombre de branches à explorer devient vite bien trop grand et ne permet pas de résoudre les dernières instances. Néanmoins elle fournit un parfait exemple de problème irrégulier et lourd en calcul et ne semble pas s'adapter à l'architecture SIMD des accélérateurs du fait de l'irrégularité du parcours.

Notre implémentation propose un découpage de l'arbre de recherche par niveaux. Un premier programme génère des tâches qui sont transmises à l'ensemble des programmes esclaves. Une fois leur calcul terminé ils transmettent le résultat et se mettent en attente

n	CPU (8c)	GPU (4c) + CPU (4c)	n	CPU (8c)	GPU (4c) + CPU (4c)
15	2.5	1.5	17	29.8	7.3
16	21.2	14.3	18	290.0	73.6
17	200.3	120.5	19	3197.5	803.5
18	1971.0	1178.2	20	–	9436.9
19	22594.2	13960.8	21	–	118512.4

(a) Regularized method (seconds) (b) Backtrack (seconds)

Table 1: Comparison between multi-core processors and GPUs for regularized and backtrack method

d’une nouvelle tâche de travail. Nous proposons et détaillons la mise en place de cette distribution sur architectures classiques utilisant MPI-OpenMP ainsi qu’avec l’utilisation de GPGPU avec MPI-OpenMP-CUDA. Nous exposons les choix d’implémentation pour la répartition et l’utilisation de l’accélérateur. Pour la résolution des tâches nous proposons une approche régulière, explorant chaque branche et feuille, mais aussi l’approche *backtrack* classique.

Les résultats sont très bon et nous ont permis d’aborder des instances du problème de Langford encore non calculées avec cette méthode, au-dessus de l’instance $L(2, 19)$. Ces résultats sont réalisés sur 20 nœuds du supercalculateur ROMEO de l’Université de Reims Champagne-Ardenne soit 40 CPU-GPU. Les résultats sont présentés sur les figures ?? Avec la méthode classique le GPU apporte un speed-up de 1.6 fois plus rapide en comparant le *backtrack* sur CPU avec un parcours complet de l’arbre sur GPU. Dans ce cas le GPU réalise une exploration de 200000 fois plus de noeuds mais est tout de même plus rapide que le CPU.

Dans le cas *backtrack*, les tests ont été réalisés sur 129 nœuds du supercalculateur ROMEO, un total de 258 CPU-GPU. Malgré cette méthode qui ne semble pourtant pas s’adapter à l’architecture synchrone et du modèle SIMD du GPU permet une accélération de 15 fois plus rapide en utilisant le GPU. Nous avons repoussés les limites de la méthode en résolvant les instances $L(2, 20)$ en moins de trois heures et $L(2, 21)$ en 33 heures.

Méthode Godfrey

Nous avons ensuite travaillé sur la méthode la plus rapide actuellement connue pour dénombrer les arrangements du problème de Langford. La méthode algébrique de Godfrey propose une sommation en explorant un sous ensemble des combinaisons possibles du problème avec :

$$\sum_{(x_1, \dots, x_{2n}) \in \{-1, 1\}^{2n}} \left(\prod_{i=1}^{2n} x_i \right) \prod_{i=1}^n \sum_{k=1}^{2n-i-1} x_k x_{k+i+1} = 2^{2n+1} L(2, n) \quad (1)$$

Plusieurs optimisations sont possibles et permettent de réduire le nombre de calculs à réaliser rendant cette méthode bien plus intéressante que l’approche classique arborescente. Nous donnons le détail de ces optimisations ainsi que notre implémentation de la méthode. L’idée reste la même que pour la méthode de Miller, les tâches sont cette fois-ci basée sur le positionnement de paires au format binaire. Chaque CPU ou GPU travaille sur une tâche et les résultats sont ensuite rassemblés. L’irrégularité avec cette résolution vient de l’utilisation de grands entiers nécessitant une représentation sur plusieurs mots

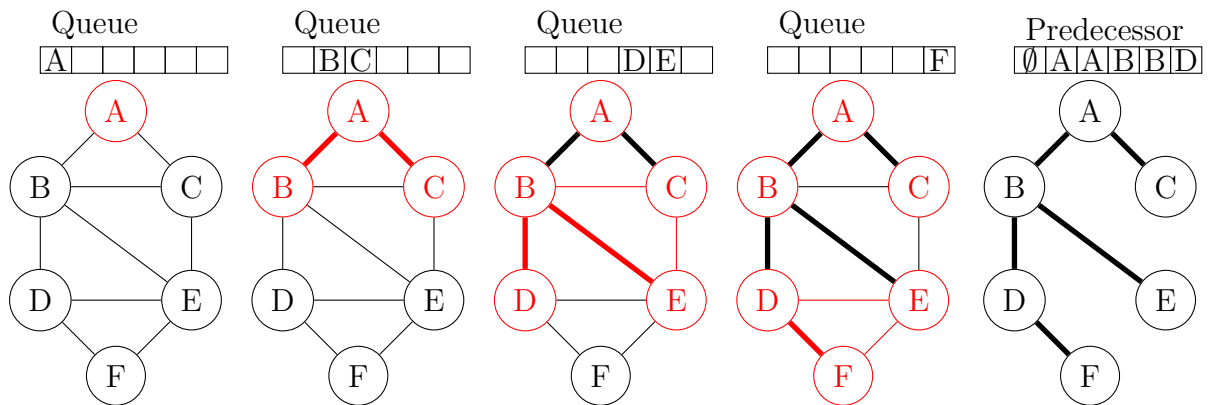


Figure 3: Example of Breadth First Search in an undirected graph

mémoires et des propagations de retenues qui ne peuvent être anticipées. La thèse présente le détail de nos choix et implémentation de cette méthode.

Les résultats sont très bon et montrent une nouvelle fois l'avantage des architectures hybrides confronté au calcul irrégulier, sans communications. Il nous a permis de recalculer les dernières itérations du problème, $L(2, 27)$ et $L(2, 28)$ dans un temps record avec respectivement 1.9 jours et 23 jours avec une stratégie de répartition Best-Effort sur le supercalculateur ROMEO (environ 70% du cluster). Cette stratégie nous permet de bénéficier des nœuds de calcul non utilisés par les chercheurs mais nécessite des redémarrages et interruptions de tâches. Les résultats avec une répartition classique de l'ensemble du cluster auraient été encore meilleurs.

Ce premier élément de notre métrique, confrontant les architectures au problème de calcul dans un environnement irrégulier montre nettement l'avantage des accélérateurs. Ils permettent une montée à l'échelle dans les deux méthodes abordées et s'adaptent même à des problématiques qui ne semblent pas nativement adaptées à leur architecture.

2.2 Communication: Graph500

La deuxième partie de notre métrique vise la problématique de communication en conservant le comportement irrégulier. Nous avons choisi le benchmark du Graph500². Ce problème de parcours en largeur d'un très grand graphe est utilisé comme base pour le classement des supercalculateurs avec une métrique spécifique, le nombre d'arête du graphe traversées par secondes, les TEPS.

L'algorithme d'un parcours en largeur sur un graphe est détaillé sur la figure ???. Il comprend beaucoup de phase de communications pour échanger les frontières et réaliser des réductions entre les processus. La parcours du voisinage de chaque sommet ne comprend pas de calcul uniquement des accès mémoire irréguliers et des tests avec copie.

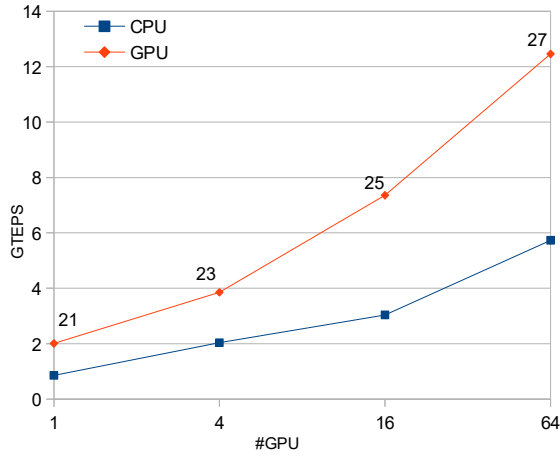
Nous proposons l'implémentation multi-CPU et multi-GPU de ce benchmark. Notre travail se base sur les travaux de NVIDIA avec Merill [MGG15] ainsi que le meilleur algorithme distribué actuellement avec le code utilisé sur les architectures BlueGene/P et BlueGene/Q [CPW⁺12] de IBM.

Notre approche propose une représentation compressée de la matrice d'adjacence appelée *Compressed Sparse Row* et *Compressed Sparse Column* en complément d'une représentation binaire des listes de frontière.

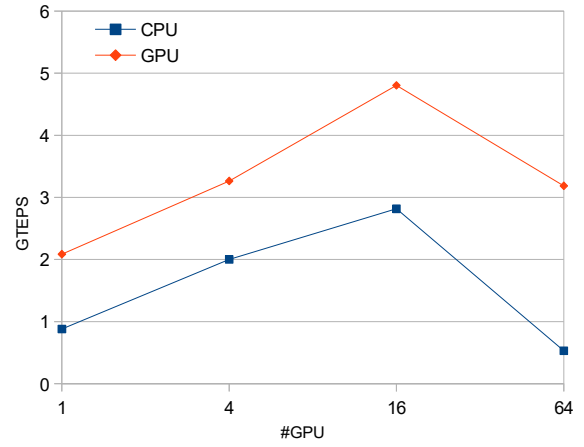
²<https://www.graph500.org>

Les communications de la version GPU se vont directement dans la mémoire de ceux-ci en utilisant la technologie GPU-Direct de NVIDIA.

Notre implémentation montre une nouvelle fois l'avantage des architectures hybrides face aux architecture classiques.



(a) CPU *vs* GPU weak scaling. The number of CPUs is the same as the number of GPUs.



(b) CPU *vs* GPU strong scaling. The *SCALE* is showed on the GPU line. The number of CPUs is the same as the number of GPUs.

Figure 4: Weak and Strong scaling between CPU and GPU

Les résultats présentés en figure ?? et figure ?? montrent respectivement le *weak scaling* et le *strong scaling* de nos deux versions. La version CPU se base sur le meilleur algorithme fournis par le groupe Graph500 sur leur page internet.

3 Application

Conclusion

La première partie de cette étude décrit les outils et la théorie nécessaire pour comprendre et atteindre la performance dans le calcul haute performance. Nous avons présenté plusieurs architectures et leurs avantages en montrant l'objectif d'un supercalculateur à échelle exaflopique aux environs 2020. Nous pensons que la solution actuelle reste l'utilisation d'architectures hybrides munies de machines many-cores tels que les GPGPU ou les FPGA.

Nous avons défini une métrique ciblant les principales limitations du calcul haute performance avec le calcul, la communication et le comportement irrégulier. L'objectif de cette métrique est de confronter les architectures classiques et hybrides à des problèmes réels.

Cette métrique est séparée en deux parties. Dans un premier temps nous avons visé des problèmes classiques, académiques et même un benchmark. Nous avons comparé les architectures classiques aux hybrides sur un premier problème lourd en calcul et un second lourd en communications. Dans les deux cas nous voulions retrouver une spécificité des codes appliqués, l'irrégularité de calcul, communication et accès mémoire.

Le premier problème choisit est le problème de dénombrement combinatoire de Langford. Nous avons étudié ce problème avec deux méthodes différentes, respectivement un parcours arborescent et une méthode algébrique. La résolution arborescente montre de très bons résultats sur accélérateurs avec jusque 80 pourcents du travail réalisé par les GPGPU avec une stratégie de répartition des tâches efficace. La solution algébrique propose des calculs régularisés mais des accès mémoires aléatoires, irréguliers, par l'utilisation de grands entiers. Dans ce cas le GPU est capable de gérer jusque 65 pourcents de l'effort global de calcul. Notre version basée sur les architecture hybrides nous a permis de battre un record de temps pour le calcul des dernières instances en utilisant le supercalculateur ROMEO de l'Université de Reims Champagne-Ardenne en mode Best-Effort.

Le second problème pour notre métrique est le benchmark du Graph500. C'est un candidat parfait pour considérer uniquement les communications, sans calculs. En effet, les seules opérations nécessaires sont des tests et des copies de zone de mémoire qui se réalisent de manière irrégulière. Ce problème respecte aussi la problématique d'irrégularité. Nous proposons une implémentation basée sur les derniers travaux à la fois de NVIDIA mais aussi BlueGene/Q et BlueGene/P. Ce travail nous a permis de classer le supercalculateur ROMEO à la 105ème place du Graph500 dans la liste de Novembre 2016. Cela montre le haut niveau de parallélisme et de scalabilité qui peut être obtenu avec des GPGPU.

Ces deux premières approches montrent un grand avantage aux architectures hybrides malgré des problèmes inadaptés au modèle d'exécution SIMD. Pour compléter notre métrique, nous avons besoin de confronter les deux types d'architectures à un problème présentant à la fois les problématiques de calcul et de communication avec un comportement irrégulier. Nous avons choisi une application de simulation complexe. Le problème

que nous avons choisi et répondant aux problématiques est la méthode Smoothed Particle Hydrodynamics (SPH) ainsi que le calcul de la gravitation appliqué à des simulations d'astrophysique. Nous avons développé, en partenariat avec le Los Alamos National Laboratory, un framework appelé FleCSPH dédié aux topologies arborescentes utilisées pour les simulations de physique et d'astrophysique. Nous montrons avec cette dernière métrique les avantages des architectures hybrides. Même en conservant une approche de code de production, pas entièrement porté sur GPU, les performances restent bien supérieures que sur les architectures classiques. Cette étude a montré que dans tous les cas étudiés, les architectures hybrides sont la meilleure solution actuelle pour atteindre l'échelle exaflopique. Les limitations sont résolues en utilisant des stratégies qui peuvent être généralisées à d'autres problèmes et de nouvelles méthodes de résolution doivent encore être trouvées.

Les architectures hybrides semblent être la meilleure approche pour bâtir des architectures exaflopiques mais l'Informatique est une science évoluant très rapidement. L'arrivée d'une nouvelle technologie pourrait changer la donne et permettre un meilleur rapport puissance de calcul et consommation énergétique. La principale autre solution réside pour le moment dans les architectures ARM. Le projet MontBlanc, que nous avons présenté en partie I, est le projet Européen de supercalculateur implémentant ces processeurs à jeu d'instruction réduit. Le projet Japonais avec RIKEN va proposer le post-K³, successeur du K Computer. Il sera bâti sur des processeurs ARMv8 et utilisera la même topologie en tore à six dimensions.

Une autre solution pour des problèmes spécifiques semble émerger avec l'Informatique Quantique. Les *bits* des processeurs classiques sont remplacés par des *qbits*, quantum bits, qui permettent de représenter plus de deux valeurs. Ces machines sont basées sur des probabilités et ne sont utilisables que sur une sous-catégorie de problèmes. Cette technologie est toujours en développement mais dans certain centre de calcul, comme ROMEO à l'Université de Reims Champagne-Ardenne, il est d'ores et déjà possible de travailler sur simulateur pour en comprendre le comportement.

Cette étude s'est intéressée aux moyens d'attendre la puissance nécessaire pour atteindre l'échelle exaflopique. Les principales limitations sont la puissance de calcul, la communication et la demande énergétique que nous avons déjà abordés dans ce travail. Mais une fois l'exaflop atteint d'autres problèmes vont survenir. Nous avons cité le problème du réseau d'interconnexion qui va augmenter lorsque les supercalculateurs vont contenir des millions de composants, rendant difficile la conservation d'une bande passante suffisante entre eux. Une autre problématique est l'augmentation des erreurs de calculs due aux influences environnementales. Elles sont de l'ordre d'une erreur par semaine sur un calculateur petaflopique mais vont atteindre de l'ordre d'une par heure pour des machines exaflopiques. Il faudra inventer de nouvelles méthodes de vérification à la volée. De la même manière les algorithmes devront être résistants aux défaillances matérielles. La problématique de résilience, le rétablissement après la perte d'un élément de calcul, nécessitera la mise en place d'algorithmes robustes et de parallélisme de tâches.

Enfin, la dernière problématique sera la complexité. Les Informaticiens ont besoin d'inventer de nouvelles manières de penser les algorithmes parallèles. D'une part la mise en place de nouveau framework et API pour gérer et cibler tous les types de topologies et d'architectures. D'autre part la production de nouvelles méthodes non hérités du calcul séquentiel pour repenser les algorithmes et calculs.

³<http://www.fujitsu.com/global/Images/post-k-supercomputer-overview.pdf>

Bibliographie

- [CPW⁺12] F. Checconi, F. Petrini, J. Willcock, A. Lumsdaine, A. R. Choudhury, and Y. Sabharwal. Breaking the speed and scalability barriers for graph exploration on distributed-memory machines. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pages 1–12, Nov 2012.
- [DBGH⁺16] Nicola De Brye, Daniel George, Glenn Hordemann, Hyun Lim, Julien Loiseau, Jonah Miller, and Johnathan Sharman. Domain partitioning and problem space representations for compact binary mergers. In *Poster at the 2016 Co-Design Summer School, Los Alamos.*, 2016.
- [DJK⁺14] Hervé Deleau, Christophe Jaillet, Michaël Krajecki, Julien Loiseau, Luiz Angelo Steffenel, François Alin, and Lycée Franklin Roosevelt. Towards the parallel resolution of the langford problem on a cluster of gpu devices. In *CSC14: The Sixth SIAM Workshop on Combinatorial Scientific Computing*, page 66, 2014.
- [Gar56] Martin Gardner. *Mathematics, magic and mystery*. Dover publication, 1956.
- [JDK⁺14] Christophe Jaillet, Hervé Deleau, Michaël Krajecki, Luiz-Angelo Steffenel, François Alin, and Julien Loiseau. Langford problem: Massively parallel resolution on a multigpu cluster. In *CSC14: The Sixth SIAM Workshop on Combinatorial Scientific Computing*, 2014.
- [KLAJ16a] Michaël Krajecki, Julien Loiseau, François Alin, and Christophe Jaillet. Bfs traversal on multi-gpu cluster. In *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, 2016.
- [KLAJ16b] Michaël Krajecki, Julien Loiseau, François Alin, and Christophe Jaillet. Many-core approaches to combinatorial problems: case of the langford problem. *Supercomputing Frontiers and Innovations*, 3(2):21–37, 2016.
- [LAJK15] Julien Loiseau, François Alin, Christophe Jaillet, and Michaël Krajecki. Parcours de grands graphes sur architecture hybride cpu/gpu. In *Modèles et Analyses Réseau: Approches Mathématiques et Informatiques (MARAMI) 2015. 14-16 Octobre Nîmes. Journée Fouille de Grands Graphes (JFGG)*, 2015.

- [LJAK15] Julien Loiseau, Christophe Jaillet, François Alin, and Michaël Krajecki. Massively parallel resolution of combinatorial problems on multigpu clusters. In *GTC Technology Conference 2015.*, 2015.
- [LJAK16] Julien Loiseau, Christophe Jaillet, François Alin, and Michaël Krajecki. Résolution du problème de langford sur architecture massivement parallèle cpu/gpu. In *Compas' Conference 2016. Lorient, June 2016, France*, 2016.
- [LLMB17] Julien Loiseau, Hyun Lim, Nick Moss, and Ben Bergen. A parallel and distributed smoothed particle hydrodynamics implementation based on the flecsu framework. In *SuperComputing conference, Denver CO*, 2017.
- [MGG15] Duane Merrill, Michael Garland, and Andrew Grimshaw. High-performance and scalable gpu graph traversal. *ACM Transactions on Parallel Computing*, 1(2):14, 2015.
- [Sim83] James E. Simpson. Langford sequences: perfect and hooked. *Discrete Math*, 44(1):97–104, 1983.