Simulations en Chimie : Les bénéfices des méthodes Monte-Carlo Quantique

Michel Caffarel, Anthony Scemama

Simulations en Chimie : Les bénéfices des méthodes Monte-Carlo Quantique

Michel Caffarel, Anthony Scemama

HPC et chimie sur ordinateur

Author: Michel Caffarel et Anthony Scemama, CNRS-Laboratoire de Chimie et Physique Quantiques, Université de Toulouse

Pour aller plus loin: http://qmcchem.ups-tlse.fr

A la recherche de la méthode qui "passe l'échelle"

L'apparition très récente des supercalculateurs massivement parallèles ouvre une nouvelle voie pour la simulation numérique. Durant la première période de l'informatique les algorithmes mathématiques à la base des simulations ont été développés dans un contexte d'architecture mono-processeur. Avec l'apparition du calcul parallèle et des premiers ordinateurs multi-processeurs dans les années 80, des efforts importants ont été produits pour "paralléliser" les algorithmes, c'est à dire permettre à plusieurs processeurs de travailler simultanément sur une même tâche. Avec l'arrivée ces dernières années des ordinateurs petaflopiques comprenant des centaines de milliers de coeurs (Roadrunner, premier ordinateur petaflopique, apparu dans la liste du TOP500 en juin 2008 comprenait 122 400 coeurs) et bientôt beaucoup plus -les supercalculateurs exaflopiques prévus autour de 2020 devraient comprendre au moins quelques dizaines de millions de coeurs- on assiste à la naissance d'un nouveau paradigme. Plutôt que de s'efforcer à paralléliser toujours plus des algorithmes qui ne s'y prêtent pas forcément, il peut être très profitable de s'écarter des approches usuelles du domaine et de se tourner vers des approches nouvelles, intrinséquement peu efficaces sur des ordinateurs à quelques milliers de processeurs, mais dont la structure algorithmique permet de profiter facilement d'un nombre potentiellement arbitrairement grand de processeurs. On sait que sur une architecture mono-processeur l'algorithme mathématique optimal est celui qui permet de calculer une quantité donnée en un nombre d'opérations élementaires le plus petit possible (par souci de simplicité on ignore ici d'éventuelles contraintes liées aux I/O et/ou à la mémoire centrale). Le temps de restitution, c'est à dire le temps qu'attend l'utilisateur pour obtenir le résultat souhaité, et le temps d'exécution sont alors essentiellement identiques et proportionnels au nombre d'opérations à effectuer. Dans une architecture parallèle on sait que la notion de nombre d'opérations à effectuer devient secondaire, l'objectif étant de réduire le temps de restitution grâce au calcul en parallèle, quitte à effectuer au total un bien plus grand nombre d'opérations élementaires. Quand le temps de restitution peut être rendu inversement proportionnel au nombre de coeurs de calcul utilisés on est alors dans une situation de parallélisme optimal. Quand ce régime peut être étendu à des nombres arbitrairement grands de processeurs, on parle alors de méthodes qui "passent l'échelle". Avec le développement des supercalculateurs au nombre toujours plus phénoménal de processeurs, disposer d'un algorithme qui passe l'échelle devient donc un véritable défi. En effet, c'est s'assurer qu'il existera toujours un nombre minimal de processeurs pour lequel l'algorithme sera supérieur à tout algorithme qui ne passe pas l'échelle, et ceci quelles que soient les performances intrinsèques (mono-processeur) de ce dernier.

Chimie sur ordinateur

Nous illustrons ici cette problématique au cas de la chimie, une discipline extrêmement gourmande en simulations numériques. La chimie est au coeur de notre vie quotidienne, peu d'aspects y échappent : développement des médicaments (drug design), nouveaux matériaux (nanosciences), énergies nouvelles, etc. Être capable de comprendre, de prédire et d'innover dans ce domaine est un enjeu de société considérable. En parallèle à la chimie traditionnelle en laboratoire se développe une véritable chimie virtuelle où les processus chimiques sont simulés sur ordinateur à partir des équations microscopiques de la matière. Avec cette nouvelle "chimie sur ordinateur", les scientifiques cherchent à reconstituer le plus fidèlement possible les échanges électroniques complexes à l'origine des liaisons chimiques, des interactions entre atomes et finalement des diverses propriétés chimiques recherchées. Sans entrer dans les détails mathématiques, ce problème constitue un formidable défi mathématique et informatique puisqu'il met en jeu la célèbre équation de Schrödinger de la mécanique quantique dont la solution recherchée -la fonction d'onde- est une fonction particulièrement complexe de l'ensemble des positions de tous les électrons (des milliers, voire plus!) et des noyaux des atomes. Durant les cinquante dernières années, et toujours en relation très étroite avec le développement des caractéristiques matérielles et logicielles des ordinateurs, plusieurs méthodes ont émergé. D'un point de vue algorithmique, elles s'appuient principalement sur des schémas itératifs de résolution de systèmes linéaires de très grandes tailles nécessitant des volumes de calculs et des besoins en I/O et/ou en mémoire centrale considérables. Malheureusement, basées sur la manipulation et le traitement de très grandes matrices, ce type d'approches se prêtent mal au calcul massivement parallèle.

Nous développons dans notre groupe une approche alternative pour la chimie, très différente des approches usuelles mais qui passe naturellement l'échelle. Basée sur une interprétation originale des probabilités de la mécanique quantique, cette approche connue sous le nom de «méthode Monte Carlo quantique» (quantum Monte Carlo, QMC) propose de simuler le monde quantique réel où les électrons possèdent un caractère délocalisé, par un monde fictif où les électrons suivent des trajectoires «classiques» comme les planètes autour du soleil. Afin d'introduire la délocalisation quantique absente de ce schéma, une composante aléatoire est ajoutée au mouvement des électrons. C'est ce caractère aléatoire qui donne son nom à la méthode : à chaque déplacement d'électron un tirage au sort est effectué comme à la roulette du casino de la fameuse ville. Chaque trajectoire ou groupe de trajectoires aléatoires peuvent être distribuées à volonté sur un nombre arbitrairement grand de coeurs de calcul, chacune de ces trajectoires évoluant indépendamment des autres (pas de communications entre elles). La situation est donc idéale du point de vue du calcul parallèle.

La Figure 1 illustre la méthode QMC au cas de la simulation de l'atome d'hydrogène, le plus simple des atomes (un électron "gravitant" autour d'un proton). L'image insérée dans le bloc bleu représente quelques milliers de pas d'une trajectoire de l'électron se déplaçant aléatoirement autour du noyau fixe (en blanc) de l'atome. Les trajectoires obtenues sur des coeurs de calcul indépendants peuvent être superposées à volonté (partie droite de la figure) et la densité électronique exacte (probabilité de présence de l'électron) est reconstruite dans le cas-limite d'un nombre infini de trajectoires.

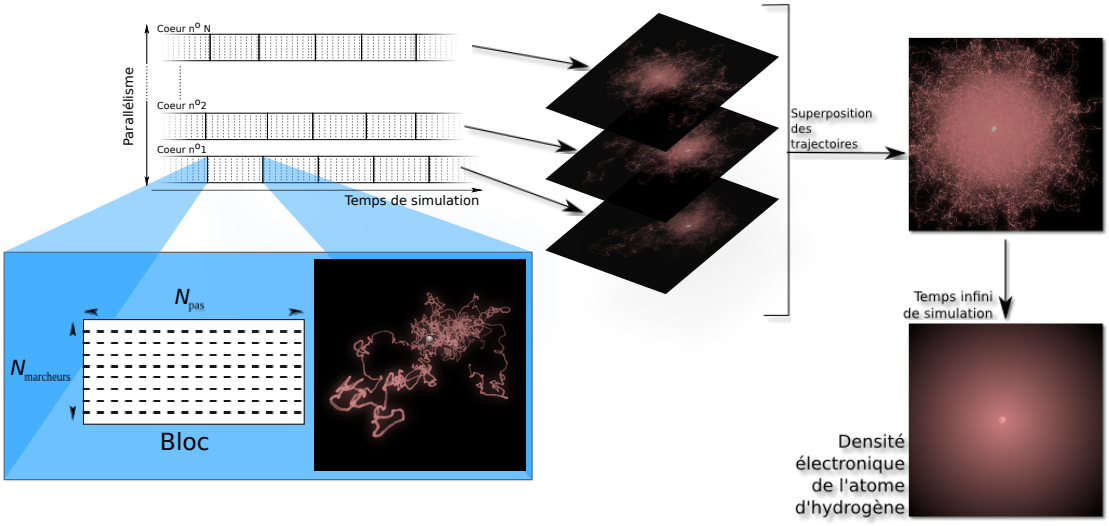

Figure 1 : Simulation QMC d'un atome d'hydrogène. Le noyau (en blanc) est fixe et les trajectoires aléatoires de l'électron sont représentées en rouge. Chaque trajectoire reconstruit la probabilité de trouver l'électron en un point de l'espace. La superposition de toutes les trajectoires donne la densité de probabilité de présence de l'électron proportionnelle au carré de la fonction d'onde, Ψ^2

Aspects computationnels

Déployer efficacement des simulations où des milliers d'électrons effectuent des milliards de pas nécessite d'optimiser au mieux les aspects algorithmiques et informatiques les plus critiques. Détaillons maintenant les principales stratégies que nous avons employées.

Nous définissons un *marcheur* comme l'ensemble des coordonnées *x*, *y* et *z* de chacun des N électrons du système (un vecteur à 3 *N* dimensions). Pour réaliser une trajectoire, un marcheur va effectuer une marche aléatoire dans l'espace à 3 *N* dimensions, de façon à reconstruire la densité de probabilité correspondant au carré de la fonction d'onde. Nous définissons un *bloc* (Figure 1) comme un ensemble de trajectoires courtes réalisées par un groupe de marcheurs pendant un certain nombre de pas (de l'ordre de 10 000). Pour chaque bloc, on calcule les valeurs moyennes des propriétés (énergie, moment dipolaire, etc) à partir de toutes les positions successives de chacun des marcheurs le long de toutes les trajectoires du bloc. Si les blocs sont suffisament longs, les positions finales des marcheurs sont complètement décorrélées des positions initiales, et on peut alors considérer les blocs comme tous indépendants. Dans ce cas, les valeurs moyennes calculées sur les blocs ont une distribution statistique obéissant à une loi Gaussienne. Chaque moyenne de bloc est un échantillon que l'on utilise pour calculer statistiquement les valeurs moyennes des propriétés.

Le but est alors de calculer le plus de blocs possible le plus rapidement possible. Pour cela nous avons adopté deux stratégies. La première est la mise en place d'un système de parallélisation efficace et la seconde est la réduction du temps de calcul de chaque bloc en travaillant sur l'optimisation mono-coeur.

Parallélisme

Lorsque l'on utilise des dizaines de milliers de coeurs, une nouvelle préoccupation devient la tolérance aux pannes. En effet, si un noeud de calcul tombe en panne en moyenne au bout de cinq ans, un calcul utilisant simultanément 2000 noeuds ne pourra pas atteindre 24 heures de calcul sans panne! Nous devons donc disposer d'un système qui survit coûte que coûte. Dans le cas d'algorithmes déterministes le calcul est découpé en tâches, et l'accomplissement de chacune des tâches est indispensable pour obtenir un résultat correct. En cas de panne, certaines tâches ne pourront pas être effectuées. Les bibliothèques MPI sont bien adaptées aux calculs déterministes. En effet, lorsqu'un client MPI ne répond plus, tout le reste de la simulation est tué car certaines tâches ne pourront pas être réalisées. Dans notre modèle, perdre des blocs au cours de la simulation ne change pas les moyennes, mais influe seulement sur la barre d'erreur du résultat. Nous avons donc choisi de ne pas utiliser MPI pour la parallélisation, mais nous avons implémenté un modèle client/serveur, où les clients calculent des blocs et envoient les résultats au serveur qui les stocke dans une base de données. Si un client meurt, tant pis, les autres clients continuent à calculer.

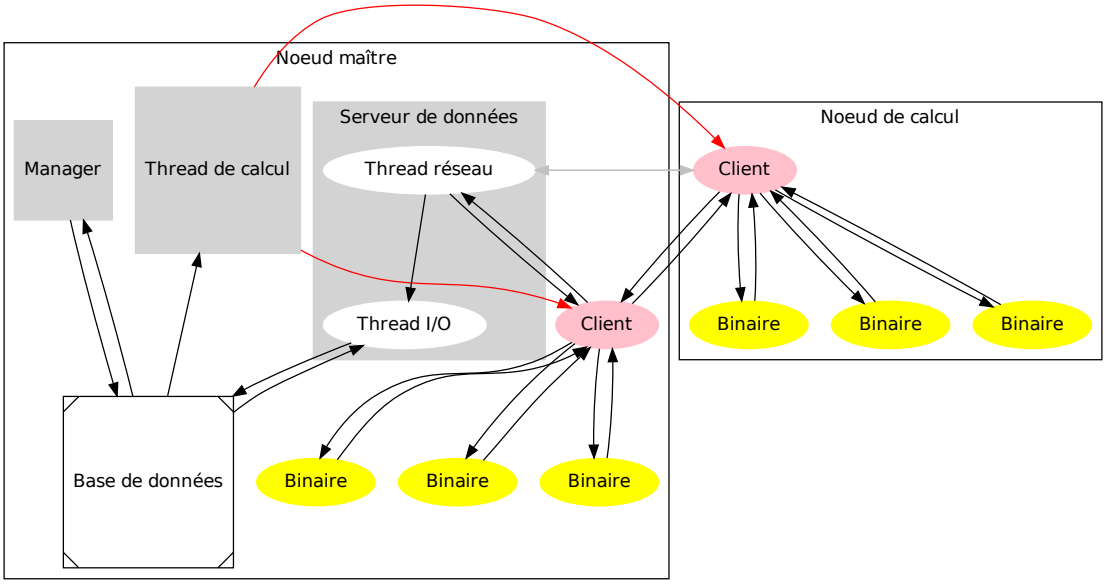

Figure 2 : Modèle client-serveur dans QMC=Chem.

Les clients et les serveurs sont des scripts Python multithreadés qui communiquent entre eux par des sockets TCP (Figure 2). Notons que les communications étant non-bloquantes et peu intensives, les latences dues à la couche TCP ne posent pas de problème. Il y a un seul client par noeud de calcul. Chaque client lance autant de programmes de calcul qu'il y a de coeurs physiques sur le noeud. Le programme de calcul est un binaire Fortran mono-thread connecté au client par un *pipe* unix. Dès que le résultat est communiqué au client, le programme de calcul commence immédiatement à calculer le bloc suivant. Simultanément, lorsque le client est inactif il transmet ses résultats à un autre client dans le but de les acheminer vers le serveur. Le serveur comporte deux threads principales: une thread *réseau* et une thread *I/O*. Le serveur reçoit des données par la thread réseau, les traite et les met dans une file d'attente. Simultanément la thread I/O vide la file d'attente pour mettre les résultats sur disque. Les clients n'accèdent jamais au disque dur local ou au système de fichiers partagé mais plutôt à un disque virtuel en RAM (/dev/shm) afin d'échapper aux pannes liées au stockage temporaire (disque plein, panne physique, etc).

Facteur d'accélération en fonction du nombre de noeuds

Puisque les blocs sont indépendants, aucune synchronisation n'est nécessaire entre les clients, et cela permet d'obtenir un facteur d'accélération presque idéal en fonction du nombre de noeuds. Cependant, il y a deux étapes critiques qui vont nuire à l'idéalité du facteur d'accélération : l'initialisation et la terminaison.

Pour avoir une initialisation rapide, il faut démarrer les clients aussi vite que possible sur les noeuds de calcul. La méthode qui nous paraît la plus rapide est l'utilisation d'un lanceur MPI qui va envoyer par *MPI_broadcast* un fichier tar contenant le binaire statique Fortran, les scripts Python et le fichier d'input sur tous les noeuds. Lorsqu'un noeud reçoit l'archive, il la décompresse et lance le client qui démarre immédiatement les programmes de calcul. Lorsque tous les clients ont démarré, le lanceur MPI se termine. Ainsi, chaque noeud de calcul démarre dès que possible. Pour un calcul d'un millier de noeuds, nous avons mesuré un temps d'initialisation de l'ordre d'une vingtaine de secondes.

Pour la terminaison, étant donné que tous les noeuds de calcul sont désynchronisés, il faudrait attendre que chacun des noeuds ait fini de calculer son bloc. Dans ce cas, il peut en résulter un temps d'attente important où beaucoup de noeuds attendent que les derniers aient terminés. On ne peut pas non plus tuer violemment tous les clients, car les résultats des blocs en cours seraient perdus. Nous avons donc permis aux programmes de calcul d'intercepter le signal SIGTERM afin d'écourter le bloc en cours et de transmettre le résultat de ce bloc au client. Ainsi, la terminaison d'un client est presque immédiate et aucune seconde de calcul n'est perdue. Pour un calcul d'un millier de noeuds, nous avons mesuré une terminaison de l'ordre de 10 secondes.

Tous ces éléments mis ensemble nous permettent d'obtenir la courbe de la Figure 3. À nombre constant de noeuds le temps d'initialisation et de terminaison est constant, donc plus le temps de calcul est long, plus l'efficacité parallèle est bonne.

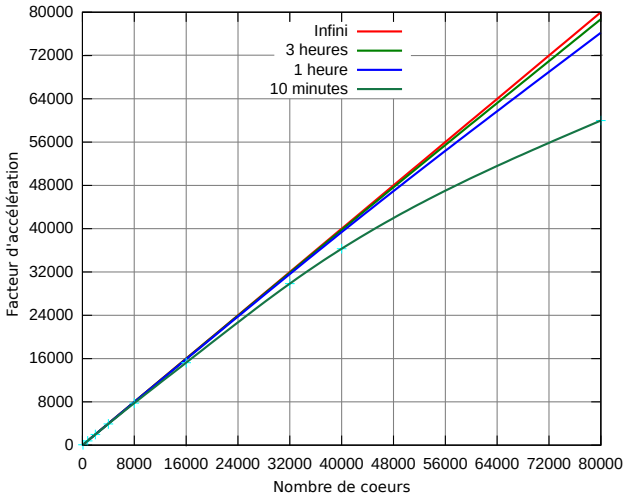

Figure 3 : Facteur d'accélération parallèle en fonction du nombre de coeurs de calcul.

Optimisation mono-coeur

Chaque seconde gagnée dans le programme de calcul se répercutera sur la totalité de la simulation car aucune synchronisation n'est bloquante. Ainsi, nous avons entrepris un travail important d'optimisation mono-coeur de notre programme QMC=Chem en profitant de l'expérience et des outils développés par le groupe du Prof. W. Jalby au Lab. Exascale Computing Research (Intel-CEA-UVSQ-GENCI) à l'Université de Versailles-Saint-Quentin.

Dans notre algorithme, nous devons évaluer à chaque pas Monte Carlo (des milliards de pas sont effectués!) la fonction d'onde, ses dérivées par rapport à chacune des coordonnées électroniques et son Laplacien. Ces opérations font intervenir des produits de petites matrices (< 1000 x 1000) dont l'une est dense et l'autre est creuse.

L'outil d'analyse statique MAQAO développé par nos collègues de Versailles nous a aidé à écrire une routine de produit matrice dense x vecteur creux dont les boucles les plus internes atteignent théoriquement les 16 flops/cycle en simple précision sur les processeurs Sandy Bridge d'INTEL de la machine CURIE. Pour cela, nous avons effectué les modifications suivantes dans le but de favoriser le plus possible la vectorisation :

- Tous les accès à la mémoire sont consécutifs
- Tous les tableaux sont alignés sur 32 octets en utilisant des directives du compilateur
- La dimension la plus interne des tableaux multi-dimensionnels est toujours un multiple de 8 éléments, afin que chaque colonne du tableau soit alignée sur 32 octets
- Déroulage de la boucle externe (*unroll and jam*) pour réduire le nombre d'écritures en mémoire
- Distribution des boucles pour ne pas dépasser les 16 registres et donc réduire les accès au cache L1.

En pratique, étant donné que nous faisons de l'ordre de N^2 opérations (N, nombre d'électrons) pour N^2 accès à la mémoire, le calcul est inévitablement limité par les accès à la mémoire et nous avons mesuré jusqu'à 61% de la performance crête du processeur dans cette routine de produit matrice x vecteur.

Lors de l'installation de la machine CURIE en décembre 2011 et grâce au soutien des ingénieurs de BULL, constructeur de la machine, nous avons pu réaliser notre tout premier calcul QMC grandeur nature sur l'ensemble des coeurs de de la machine. La machine étant en cours d'installation, les calculs ont subi des interruptions et des pannes. Ce qui initialement nous avait apparu comme une gêne est devenu finalement une chance puisque le fait que, malgré ces difficultés, nos simulations aient pu être menées à bien a validé en pratique la grande robustesse de notre schéma.

Simulations petaflopiques pour la chimie d'Alzheimer

Grâce aux moyens exceptionnels mis à notre disposition ces deux dernières années par GENCI et le consortium européen PRACE notre groupe a pu démontrer la faisablilité pratique de nos simulations pour des systèmes difficiles à atteindre par les méthodes usuelles du domaine. Les applications abordées ont concerné la chimie de la maladie d'Alzheimer qui met en jeu des interactions particulièrement délicates à reproduire. On sait que cette maladie est associée à une dégénérescence du cerveau liée à l'apparition de plaques composées d'agrégats de molécules (peptides) dites β-amyloïde. Comprendre pourquoi et comment ces molécules s'agrègent est un des enjeux majeurs de la compréhension des mécanismes les plus fondamentaux à l'origine de l'apparition de cette maladie grave. En collaboration avec le groupe de biochimie expérimentale du Prof. P.Faller (LCC, Toulouse), nous avons pu démontrer que la précision chimique nécessaire pour décrire quantitativement ces systèmes pouvait être atteinte en utilisant un supercalculateur petaflopique comme CURIE (TGCC-GENCI-CEA à Bruyères-le-Chatel). La figure 4 illustre comment les trajectoires aléatoires des électrons se déploient dans le cas d'un des systèmes chimiques simulés.

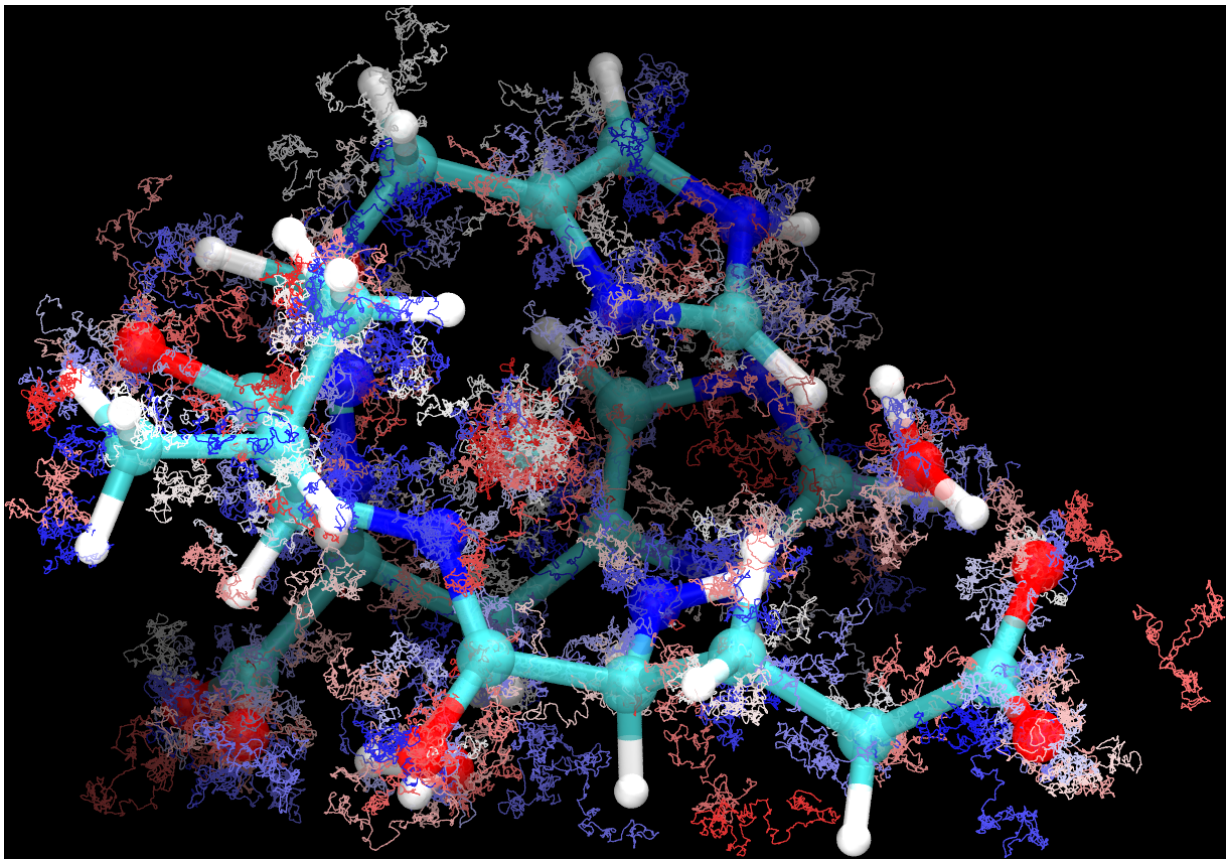

Figure 4 : Trajectoire stochastique des électrons autour des noyaux d'un fragment de β-amyloïde.

Lors de ces simulations, un débit de calcul réel de presque 1 petaflops/s soutenu pendant plusieurs heures (précisément, 960 TFlops/s en simple et double précision) a pu être mesuré.

Même si beaucoup reste à faire, nos premières simulations probabilistes sur une architacture massivement prallèle ouvrent des perspectives importantes. Bien que les volumes de calcul nécessaires -des millions d'heures CPU mono-processeur- soient encore beaucoup trop importants pour en faire des méthodes de routine, l'arrivée d'ici à 2020 des machines exaflopiques mille fois plus puissantes ainsi que les supercalculateurs qui suivront, permet d'espérer transformer ces techniques émergentes en un outil de simulation accessible à la communauté scientifique la plus large.