Conception et évaluation d'un algorithme de tolérance aux fautes à points de reprises coordonnées pour MPICH2

Prix de la Fondation Louis Leprince-Ringuet
Texte complémentaire

Camille Coti

Table des matières

1 Contexte

1.1 Introduction

Au cours de ma formation à Télécom INT, j'ai choisi de me spécialiser dans l'informatique parallèle et distribuée en choisissant l'option de dernière année "Architecte de Services en Réseaux". Les enseignements que j'y ai reçus m'ont vivement intéressée, si bien que j'ai ressenti en fin de semestre l'envie d'approfondir certains points vus en cours. Dans le même temps, j'étais attirée par la recherche. C'est pourquoi j'ai choisi d'effectuer mon stage de fin d'études au Laboration de Recherche en Informatique.

1.2 Problématique générale

L'industrie a aujourd'hui besoin d'importantes capacités de calcul pour effectuer des simulations numériques dans de nombreux domaines : géophysique, physique des hautes énergies, conception mécanique, énergie, physique nucléaire... C'est pour remplir ces besoins que l'on mutualise des ressources matérielles de stockage et de calcul afin d'obtenir les performances nécessaires pour réaliser ces simulations. Le top500[1] recense semestriellement les cinq cent super-calculateurs les plus puissants au monde et publie leurs performances. C'est ainsi qu'aujourd'hui, plus de 80% des machines les plus puissantes au monde sont constituées d'au moins 500 processeurs, et 30% comprennent au moins 1025 processeurs. La machine la plus puissante à ce jour est composée de 131 072 processeurs.

Même si les composants de ces systèmes sont choisis avec le plus grand soin, les pannes matérielles dans de tels systèmes sont inévitables. Une étude statistique a montré qu'un calcul effectué sur la machine ASCI-Q, composée de "seulement" 4 096 processeurs, et durant 12 heures, a 50% de chances de se terminer. La probabilité de panne croît avec le nombre de composants du systèmes et avec la durée du calcul. Or, les calculs effectués sur de telles machines durent entre plusieurs heures et plusieurs jours. Une défaillance, si elle n'est pas gérée par un mécanisme approprié, engendre la perte totale du calcul en cours. Il est donc indispensable de pouvoir tolérer les inévitables défaillances dans ces systèmes.

Une autre possibilité utilisée par les entreprises pour obtenir des capacités de calcul importantes consiste à utiliser les ressources existantes. En effet, la

[1] http ://www.top500.org

plupart des ordinateurs d'une entreprise ne sont généralement pas utilisés la nuit. De plus, il y a des moments de la journée où leurs capacités de calcul ne sont pas exploitées, comme lors des pauses déjeuner, ou lorsque leurs utilisateurs font quelque chose ne nécessitant pas l'usage d'un ordinateur. On peut alors les mutualiser pour créer ce que l'on appelle une grille d'entreprise. On définit sur chaque machine un signal lui faisant rejoindre la grille, souvent la mise en place de l'écran de veille. Lorsque son utilisateur recommence à s'en servir, elle quitte la grille pour qu'il puisse s'en servir à nouveau de manière exclusive. Les machines de la grille d'entreprise sont donc volatiles, c'est-à-dire qu'elles peuvent joindre et quitter la grille à tout moment. Un mécanisme de tolérance aux fautes permet de gérer cette volatilité.

1.3 Contexte du stage

J'ai effectué ce stage au sein du projet MPI-V, V signifiant "volatile". Le but de ce projet est d'évaluer les algorithmes de tolérance aux fautes dans les applications parallèles communiquant par passage de messages et d'en concevoir de nouveaux. Les protocoles existants sont pour la plupart connus uniquement théoriquement. Leur évaluation passe par une implémentation dans une bibliothèque de programmation parallèle déjà existante (MPICH ou OpenMPI) et par des mesures de performances sur des applications scientifiques typiques. Plusieurs critères de performances sont pris en compte. Tout d'abord, l'impact sur les performances du fait que les fautes sont tolérées : certains protocoles ajoutent une latence, qui doit être la plus faible possible. Ensuite, la perte de calcul engendrée lorsqu'une panne survient. La combinaison de ces deux critères détermine si un protocole sera plus adapté à un environnement où les fautes sont rares, ou si au contraire le système est fréquemment confronté à des défaillances.

2 Environnement humain et managérial

2.1 Le laboratoire

J'ai effectué ce stage au sein de l'unité de recherche Futurs de l'INRIA, située à Orsay, Lille et Bordeaux. Cette unité travaillant en partenariat avec des laboratoires universitaires, mon stage s'est déroulé au Laboratoire de Recherche en Informatique (LRI) de la faculté de sciences d'Orsay (Université Paris XI). Les chercheurs du LRI sont organisés en équipes de recherche. J'ai été pour ma part intégrée à l'équipe Parallélisme.

2.2 Le projet Grand Large

Le projet Grand Large est un projet commun entre l'INRIA Futurs, le LRI et le LIFL (situé à Lille). Il regroupe donc des employés de l'INRIA (Chargés de Recherches et Directeurs de Recherches), de l'Université Paris XI (Maîtres de Conférences et Professeurs) et des doctorants et stagiaires. Les Maîtres de Conférences et Professeurs sont enseignant-chercheurs, c'est-à-dire qu'ils ont une charge d'enseignement en plus de leurs activités de recherches. Les Chargés et Directeurs de Recherches n'ont pas obligatoirement de charge d'enseignement. De plus, tous encadrent des doctorants et des stagiaires.

Le projet Grand Large a pour but d'étudier les systèmes à grande échelle, comme les grilles de calcul. Il est dirigé par Franck Cappello, Directeur de Recherches à l'INRIA Futurs et membre de l'équipe Architectures Parallèles du LRI. Au LRI, ses membres font partie des équipes "Architectures Parallèles", "Parallélisme" et "Théorie des Graphes et Fondements des Communications". Les thématiques de l'équipe Parallélisme sont les algorithmes répartis et le calcul à grande échelle. Le projet MPI-V, dont le but est d'étudier les protocoles de tolérance aux pannes dans les systèmes repartis à grande échelle, se situe donc à l'intersection entre le projet Grand Large et l'équipe Parallélisme.

2.3 L'équipe

La structure hiérarchique au sein des équipes est définie par les responsabilités induites par les postes occupés. Le laboratoire et l'INRIA ont chacun un organe de direction qui lui est propre ; le LRI est dirigé par Michel Beaudouin-Lafon, l'INRIA par Michel Cosnard, et l'unité Futurs de l'INRIA par Claude Puech. À plus petite échelle, les équipes de recherche ont chacune un directeur ; la directrice de l'équipe parallélisme est Brigitte Rozoy, qui est Professeur à l'Université Paris XI. Le projet Grand Large est dirigé par Franck Cappello, Directeur de Recherches à l'INRIA. Il est nécessaire d'être titulaire d'une Habilitation à Diriger des Recherches pour encadrer des doctorants, ceux-ci sont donc encadrés par des Professeurs et Directeurs de Recherches ou par des Maîtres de Conférences ou Chargés de Recherche titulaires d'une HDR. Enfin, les stagiaires peuvent être aussi encadrés par des Maîtres de Conférences ou par des Chargés de Recherches. J'ai été encadrée au cours de ce stage par Thomas Hérault, Maître de Conférences à l'Université Paris XI. Les assistants de projet et les ingénieurs sont situés sous la hiérarchie directe du chef de projet.

À mon arrivée, j'ai été frappée par la bonne ambiance et la cohésion régnant dans cette équipe. On m'a tout de suite expliqué que le tutoiement était de rigueur, quelle que soit la position hiérarchique des uns et des autres : avant d'occuper tel ou tel poste, nous étions avant tout des collègues. Il est évident que le fait de travailler dans la bonne humeur et de bien s'entendre avec ses collègues améliore la qualité du travail fourni. Je pense que c'est particulièrement vrai dans une équipe de recherche, où si les personnes se sentent bien au sein de leur équipe, elles seront plus enclines à collaborer avec leurs collègues plutôt que de travailler seules.

Outre la bonne ambiance générale, plusieurs choses peuvent être notées pour expliquer le caractère soudé de l'équipe. Tout d'abord, la convivialité générale. Tous les midis, nous étions une bonne partie de l'équipe à déjeuner ensemble au restaurant du personnel de la Faculté d'Orsay, du chef de projet aux stagiaires, nous étions tous assis à la même table et pouvions parler ensemble dans un contexte dissocié de notre travail. Les activités communes de manière générale, qui créent une part d'histoire commune entre les membres de l'équipe, m'ont parues très importantes. La participation à l'organisation d'une conférence, des sorties au restaurant, le pique-nique annuel de l'équipe et les séminaires d'équipe sont autant d'évènements vécus ensemble, qui créent des souvenirs communs entre leurs protagonistes.

3 Méthodologie de travail

Mon stage s'est déroulé en deux parties, chacune impliquant un contexte et une méthodologie propres aux objectifs fixés.

3.1 Implémentation d'un protocole

Je suis arrivée dans l'équipe du projet MPI-V au moment de l'implémentation d'un protocole de tolérance aux fautes par points de reprise coordonnés dans une bibliothèque de programmation par passage de messages existante (MPICH2). Il s'agissait de production d'un logiciel. Nous étions alors 7 à travailler sur le développement de ce logiciel. J'ai écrit un composant de ce protocole (le serveur sur lequel les images des processus sont enregistrées), et j'ai dû travailler avec la personne qui s'est occupée de cette prise d'image afin d'intégrer la communication avec le serveur que j'avais développé.

Une fois le logiciel développé, nous l'avons évalué. Pour ce faire, nous avons comparé ses performances avec celles d'un logiciel précédemment développé

par l'équipe de MPI-V, qui était une implémentation d'un autre protocole. Le nouveau logiciel, MPICH2-*Pcl*, est une implémentation bloquante d'un algorithme de prise d'image simultanée sur l'ensemble du système, que nous avons comparé à MPICH-*Vcl*, qui en est une implémentation non bloquante. Le caractère bloquant du protocole *Pcl* implique un arrêt de l'exécution de l'application durant la prise d'image et sa sauvegarde sur un serveur supposé stable, ce qui a pour effet d'arrêter l'exécution pour effectuer un synchronisation de tous les processus. Cependant, il s'agit d'un protocole plus simple que *Vcl*, qui implique de sauvegarder tous les messages envoyés pour compenser l'absence de synchronisation entre les processus au moment de la prise d'image. Nous avons ici voulu voir si les meilleures performances en l'absence de fautes et entre les prises d'images de *Pcl* compensaient le ralentissement au moment de la synchronisation.

Nous avons donc soumis les deux logiciels à des tests, ou benchmarks, constituant des utilisations typiques de ce logiciel. Il s'agit de calculs courants dans les applications numériques scientifiques. Les deux benchmarks que nous avons utilisés font partie d'une suite produite par la NASA et largement reconnue dans la communauté des applications parallèles. Nous avons utilisé d'une part le benchmark CG, qui est un calcul de Gradient Conjugué, impliquant beaucoup de communications de petite taille entre les processus : il met en évidence la latence des communications inter-processus. Nous avons aussi utilisé le benchmark BT, qui est un calcul de matrice triangulaire par blocs et qui implique des communications plus rares mais de grande taille, mettant ainsi en évidence la bande passante du réseau reliant les processus.

Une fois les résultats obtenus, nous avons rédigé un article que nous avons soumis à une conférence. Comme je commençais mon stage, je venais de me documenter sur les protocoles existants en lisant un grand nombre d'articles publiés sur le sujet. On m'a donc demandé d'écrire la partie concernant les travaux effectués antérieurement sur le sujet. J'ai d'abord écrit seule ce qui m'était demandé, puis je l'ai retravaillé avec une personne de l'équipe expérimentée en rédaction scientifique qui m'a beaucoup appris sur la façon d'écrire. La rédaction d'articles implique un savoir-faire qui s'acquiert avec l'expérience : il faut écrire de manière claire, précise et concise les travaux réalisés et les résultats obtenus de manière à apporter toutes les informations au lecteur tout en restant dans les limites de taille imposées.

Cette première phase a occupé le premier mois de mon stage. L'article que nous avons soumis a été accepté par une conférence internationale. Nous

avons alors reçu les commentaires des relecteurs et effectué les modifications demandées avant d'envoyer la version finale. Chacun a retravaillé la partie le concernant au moment où nous avons reçu les commentaires, c'est-à-dire environ deux mois après la soumission de l'article. L'envoi de la version finale de l'article implique des démarches administrative, notamment à propos de copyright : les auteurs doivent transférer leur copyright à l'éditeur afin que l'article soit publié. J'ai été impliquée dans ces démarches, m'occupant de cette formalité légale. Je me suis ensuite consacrée à la deuxième partie. Si j'ai travaillé au sein d'une équipe et de manière très encadrée durant cette première partie, le travail que j'ai effectué durant la suite de ce stage était plus individuel et j'ai alors du faire preuve d'autonomie dans mon travail.

3.2 Conception et évaluation d'un protocole

Les protocoles de tolérance aux pannes connus pour le moment concernent surtout les clusters. Si certains s'adaptent plutôt bien aux grilles, il est nécessaire de les évaluer, et de prendre en compte la topologie des grilles pour en concevoir d'autres plus adaptés. C'est dans cette optique que j'ai travaillé durant la deuxième partie de mon stage.

J'ai commencé par me documenter en lisant des articles concernant le protocole que j'allais tenter d'adapter à la grille (protocole de journalisation causale des messages) et des articles concernant les protocoles adaptés aux systèmes à grande échelle. Ce protocole présente une faiblesse qui le rend incompatible avec les systèmes à grande échelle et les grilles de calcul en particulier, puisqu'il a besoin d'un composant central, présent jusque là en un seul exemplaire dans le système (c'est-à-dire non réparti et devant supporter à lui seul la charge liée à tout le système). J'ai trouvé un article particulièrement intéressant proposant une hiérarchie dans les composants du système.

Cependant, cette architecture hiérarchique ne replissait pas les objectifs fixés : le protocole proposé semblait bien capable de supporter une grande échelle, mais il n'était pas du tout tolérant aux défaillances, et introduisait une sensibilité aux pannes plus importante en déplaçant le point central de fautes.

L'élément critique des protocoles causaux s'appelle un enregistreur d'évènements. Il a été montré que, dans un cadre d'optimisation des performances du protocole, il est indispensable. Des protocoles astucieux ont été proposés, tels que LogOn ou Manetho, proposant de réduire la quantité d'informations

transportées par le protocole, sans qu'aucun n'atteigne les performances obtenues en introduisant un enregistreur d'évènements.

Cependant, les protocoles causaux de journalisation des messages sont des protocoles de cluster : ils ne sont pas adaptés à la topologie des grilles, ni à la grande échelle. De plus, ils nécessitent que l'enregistreur d'évènements soit situé sur une machine que l'on considère comme stable car on ne tolère pas les pannes sur ce composant.

J'ai donc repris l'idée de hiérarchie proposée par l'article que j'avais lu, sans toutefois la suivre dans son intégralité. Cet article propose une hiérarchie complète d'enregistreurs d'évènements de proximité (ou proxys) en forme de pyramide, remontant jusqu'à un sommet. Ce sommet constitue un point central de faute, et potentiellement un goulet d'étranglement. J'ai repris l'idée de proxys, en ne gardant qu'un seul niveau hiérarchique : chaque cluster dispose de son enregistreur d'évènement de proximité, et les proxys mettent à jour leurs informations auprès des autres proxys en suivant un protocole pouvant s'apparenter à un algorithme de cohérence de cache. Ainsi, on tend vers une réplication d'un enregistreur d'évènement global dans chaque cluster.

Une fois le protocole conçu, je l'ai évalué. Ne disposant pas du temps nécessaire pour en faire une implémentation complète, à l'image de l'implémentation du protocole *Pcl* à laquelle j'ai participé dans la première partie de mon stage, j'ai réalisé un prototype permettant d'avoir une idée des performances de ce protocole.

Le prototype se compose de deux parties : l'enregistreur d'évènements, et le pseudo-client.

L'enregistreur d'évènements est l'élément le plus proche de celui qui serait inclus dans une implémentation complète. Les différences avec une implémentation "réelle" concernent les détails de l'intégration dans une architecture de bibliothèque MPI, comme la communication des ports d'écoute, et le comportement au moment du redémarrage d'un processus ou d'un cluster.

Le pseudo-client reproduit l'exécution d'une application MPI à partir d'un fichier de traces généré par l'exécution d'une application MPI. Ce fichier de traces a été obtenu en exécutant une application MPI en utilisant une bibliothèque MPICH2 instrumentée afin d'écrire des informations permettant

de jouer des messages similaires à ceux envoyés lors de l'exécution d'une application. Ainsi, j'ai pu tester le protocole en faisant jouer un benchmark par ce prototype sans disposer d'une implémentation complète de la norme MPI implémentant également ce protocole.

J'ai ensuite testé ce protocole dans plusieurs configurations, afin d'en évaluer différents aspects. Tout d'abord, il s'agissait d'évaluer l'impact du protocole sur les performances de l'application. Pour une application s'exécutant sur un petit (inférieur à 100) nombre de processeurs, dans un cluster, le protocole causal de base, utilisant un seul enregistreur d'évènements, convient. Je me suis donc assurée que le fait d'utiliser plusieurs enregistreurs d'évènements n'avait pas d'effet négatif sur les performances en augmentant le nombre d'enregistreurs d'évènements pour une application s'exécutant sur un cluster et utilisant 49 processeurs. En effet, les performances ne sont pas affectées de manière significative : le protocole permet au moins de tolérer les fautes par réplication de l'enregistreur d'évènements.

Ensuite, l'objectif suivant de ce protocole est de pouvoir être utilisé dans des systèmes à grande échelle. J'ai donc mesuré les performances d'une application s'exécutant sur 144 processeurs. En effet, si l'on augmente le nombre d'enregistreurs d'évènements utilisés, les performances s'améliorent, puis ne varient plus à partir du moment où les enregistreurs d'évènements sont suffisamment nombreux pour ne pas être surchargés.

Enfin, une dernière famille d'expériences m'a permis de vérifier l'intérêt du protocole sur une grille en comparant les performances en utilisant un enregistreur d'évènements dans un cluster et celles en en plaçant un dans chaque cluster. J'ai effectué ces expériences en utilisant une petit nombre de processus (25), échelle à laquelle le nombre d'enregistreurs d'évènements n'a pas d'importance. Ces expériences ont mis en évidence le gain de performances apporté par l'enregistreur d'évènements local.

Le protocole que j'ai conçu remplit donc ses objectifs, qui étaient :
- Tolérer les pannes sur l'enregistreur d'évènements par réplication
- Permettre d'utiliser des systèmes de grande taille en diminuant la charge sur l'enregistreur d'évènements
- Améliorer les performances sur les grilles en limitant le nombre d'invocations d'un composant distant

4 Difficultés rencontrées

J'ai commencé mon stage au moment du développement du logiciel MPICH2-*Pcl*. Je n'avais pas participé à la conception du protocole, ni à celle de l'architecture du logiciel. Il a fallu que je m'intègre dans un projet en cours et que je participe à son développement bien que n'ayant pas participé aux phases préliminaires. Cependant, cette intégration a été facilitée par les autres membres du projet, qui étaient m'ont bien expliqué ce que j'avais à faire à mon arrivée et sont restés très disponibles pour répondre à mes interrogations.

Ensuite, j'étais totalement novice en matière de tolérance aux fautes au début de mon stage. J'avais certes suivi dans le cadre de ma formation à Télécom INT des cours portant sur les algorithmes répartis, mais il s'agissait maintenant d'approfondir ces bases en acquérant les connaissances spécifiques au domaine dans lequel je travaillais. Je me suis donc documentée, effectuant une recherche bibliographique en même temps que le début de mon stage. On m'a d'abord indiqué des articles à lire, puis j'en ai trouvé par moi-même. Les articles précédemment publiés dans le cadre du projet MPI-V constituent une bonne base de départ, puis j'ai regardé ce qui avait été publié par d'autres personnes sur la tolérance aux pannes. J'ai utilisé les moments où je n'étais pas en train de faire du développement (synchronisations avec le reste de l'équipe, intégration de composants...) pour les lire.

Enfin, la première partie de mon stage a été marquée par une date limite avant laquelle nous devions soumettre l'article présentant les résultats de nos expériences. Nous nous étions fixé cette conférence pour objectif, et il était impossible de soumettre l'article après la date butoir. Nous devions donc finir le développement du logiciel dans un temps très court, et effectuer nos expériences sur une plate-forme que nous n'étions pas les seuls à utiliser. S'agissant d'une plate-forme expérimentale comptant un grand nombre d'utilisateurs, dont beaucoup ont les mêmes dates limites, elle était particulièrement sollicitée à ce moment-là. Alors qu'elle est d'ordinaire occupée à 70% en moyenne, son taux d'utilisation est monté durant ces quelques semaines à plus de 95%. Il a alors fallu non seulement effectuer les réservations longtemps à l'avance, et donc prévoir les besoins des expériences, mais aussi accepter d'utiliser la plate-forme aux moments où elle était disponible, c'est-à-dire souvent la nuit. Nous nous sommes donc relayés 24H/24, certains effectuant des expériences, d'autres analysant les résultats obtenus, les autres s'occupant de rédiger l'article. C'est grâce à cette organisation que nous avons réussi à soumettre notre article à temps. Nos efforts ont été récompensés, car la conférence à laquelle nous l'avons soumis l'a été accepté.

Durant la deuxième partie de mon stage, la principale difficulté à laquelle j'ai été confrontée est relativement prévisible chez les stagiaires puisqu'il s'agit de mon manque d'expérience. En effet, je me trouvais à travailler seule face à un projet d'une taille inédite pour moi. J'ai donc rencontré un certain manque de méthode devant le développement du prototype de mon protocole. En a découlé une perte de temps considérable. J'ai donc eu besoin de plus de temps qu'une personne plus expérimentée pour développer le prototype permettant d'évaluer le protocole, mais je pense que ces piétinements et retours en arrière ont eu un intérêt pédagogique, et que dans le futur je perdrai moins de temps.

5 Conclusion : Résultats obtenus

5.1 Résultats du stage

5.1.1 Implémentation d'un protocole

La première partie de mon stage, portant sur l'implémentation d'un protocole de tolérance aux pannes dans une bibliothèque MPI, a permis de comparer en situation d'utilisation typique deux protocoles de tolérance aux pannes. Nous avons ainsi mis en évidence que le fait de synchroniser l'ensemble des processus d'une application à grande échelle ou s'exécutant sur une grille de calcul était trop coûteux en termes de performances, cependant la simplicité de ce protocole lui permet d'offrir de meilleurs performances sur les clusters équipés de réseaux rapides (ici myri2000). Notre travail a été reconnu par la communauté de recherche dans laquelle il s'inscrit, puisque l'article présentant ces résultats a été accepté dans une conférence internationale présentant un taux d'acceptation de 22%. On m'a de plus demandé d'en faire la présentation lors de deux journées de rencontre des utilisateurs de la plate-forme expérimentale que nous avons utilisée.

5.1.2 Conception et évaluation d'un protocole

La deuxième partie de ce stage a été consacrée à la conception d'un protocole de tolérance aux pannes adapté aux grilles de calcul et aux systèmes à grande échelle, puis à son évaluation. Le protocole que j'ai conçu remplit ses objectifs. L'enregistreur d'évènements des protocoles de journalisation de messages n'ayant plus à être situé sur une machine considérée comme stable, on peut maintenant considérer des systèmes constitués d'un plus grand nombre de processus qu'auparavant, et ce protocole est adapté à la topologie des grilles de calcul. Si l'implémentation qui m'a servi à l'évaluer

n'est que prototypaire, elle permet de se faire une idée de ses performances. Maintenant que les objectifs de ce protocole se sont avérés remplis, il est possible qu'il soit implémenté dans une bibliothèque MPI dans le futur afin de le comparer à des protocoles existants et de mesurer ses performances de manière plus précise. Ce protocole pourrait par la suite faire l'objet d'un article dans une conférence scientifique.

5.2 Apports personnels

Le stage de fin d'études est souvent la première expérience professionnelle "significative" des élèves-ingénieurs. C'est en effet la première fois que nous devons occuper un poste dans une entreprise ou un laboratoire, avec une mission correspondant à notre formation, sur une durée d'au moins 5 mois, donc suffisamment longue pour que l'on ait le temps de se rendre compte si ce que l'on est en tain de faire nous plaît vraiment. C'est une occasion de voir si un domaine, un environnement de travail et un type d'entreprise (PME, grande entreprise, laboratoire...) nous plaît vraiment, si l'on est prêt à s'y engager ou s'il vaut mieux que l'on se tourne vers autre chose par la suite.

Dans mon cas, il s'agissait de savoir si j'étais prête à m'engager dans la recherche. La réponse, après six mois dans un laboratoire public, est positive. C'est vraiment ce que j'ai envie de faire, et c'est pourquoi je souhaite poursuivre dans ce domaine. Dans la continuité de mon stage, j'ai commencé un doctorat à l'INRIA Futurs (toujours dans le projet Grand Large) portant sur la tolérance aux pannes dans les systèmes de calcul à grande échelle dans la cadre du projet européen QosCosGrid[2]. Les résultats obtenus dans le cadre de ce projet pourront être utilisés par les entreprises partenaires de ce projet.

[2]Quasi-Oportunistic Supercomputing for Complex Systems in Grid Environments