Table des matières

1 Introduction

Ce rapport présentera mon stage dans le LIP de l'ENS de Lyon où j'ai travaillé sur la mise en place d'un simulateur permettant de tester des heuristiques d'ordonnancement de workflow sur une plateforme de calcul distribué. Ce stage a pris place du 2 mai 2023 au 28 juillet 2023, et il a eu lieu dans les locaux de l'ENS sur le site Monod.

2 Présentation de l'entreprise et contexte du stage

2.1 Présentation de l'ENS

2.1.1 Présentation globale de l'entreprise

Une École Normale Supérieure (ENS) est un établissement d'enseignement supérieur public qui se consacre à la formation de chercheurs et d'enseignants dans les domaines littéraires, scientifiques et technologiques. Les ENS font partie des grandes écoles les plus prestigieuses et sont placées sous l'autorité du ministère de l'Enseignement supérieur et de la Recherche. Ces écoles offrent une formation académique de haut niveau et jouent un rôle important dans la promotion de la recherche et de l'enseignement.

L'ENS de Lyon [1] est l'une des quatre grandes écoles normales supérieures en France. Elle offre une formation en sciences fondamentales, expérimentales, lettres et sciences humaines, axée sur l'enseignement et la recherche. L'ENS de Lyon résulte de la fusion des anciennes écoles normales supérieures de Fontenay-aux-Roses et de Saint-Cloud entre 1987 et 2010. Elle est située à Lyon sur le campus de Gerland et fait partie de l'Université de Lyon.

2.1.2 Échelle de l'organisation

L'école accueille des étudiants non rémunérés, des auditeurs et des étudiants étrangers. Elle compte environ 2 200 étudiants, 1 000 chercheurs et abrite de nombreux laboratoires et centres de recherche. Elle est réputée pour son rôle historique dans les domaines de la pédagogie et de l'éducation.

2.1.3 Différents départements de l'ENS

L'ENS de Lyon propose des formations dans les domaines des sciences, des lettres et des sciences humaines. Elle compte six départements dédiés aux sciences exactes et expérimentales, telles que la biologie, la chimie, les sciences de la terre, l'informatique, les mathématiques et la physique. De plus, elle abrite six départements axés sur les lettres et les sciences humaines, comprenant l'éducation et les humanités numériques, les langues, les littératures et les civilisations étrangères, les lettres et les arts, les sciences humaines, l'économie et les sciences sociales.

2.2 Présentation du LIP

L'École accueille le Laboratoire de l'Informatique du Parallélisme (LIP). Le LIP est organisé en huit équipes de recherche :

— AriC : L'objectif principal d'AriC est de travailler sur l'amélioration des problématiques de performance, d'efficacité et de fiabilité grâce à l'arithmétique informatique. Leur travail porte notamment sur les algorithmes arithmétiques tels que l'arithmétique entière

1. http://www.ens-lyon.fr/

et a virgule flottante, l'arithmétique complexe, l'arithmétique à précision multiple et l'arithmétique sur corps finis, ainsi que sur leurs l'implémentations. Ils s'intéressent également aux méthodes d'approximation, aux réseaux euclidiens et à la cryptologie, ainsi qu'au calcul certifié et à l'algèbre informatique.

- Avalon : L'objectif principal d'Avalon est d'améliorer l'efficacité énergétique des systèmes à grandes échelles en analysant leur consommation d'énergie et en développant de nouveaux mécanismes pour une utilisation plus efficace. Ils étudient également la gestion des ressources distribuées, en mettant l'accent sur la planification des workflows, le traitement des flux de données et la sécurité dans les environnements Cloud.
- Cash : Ils travaillent sur la représentation des programmes parallèles à l'aide du modèle de flux de données. Leur objectif est de concevoir des analyses statiques expressives et évolutives pour les compilateurs, d'optimiser les programmes de flux de données en utilisant le modèle polyédrique, et de développer des techniques de compilation efficaces pour la synthèse de haut niveau (HLS). De plus, ils se concentrent sur la simulation de systèmes à travers des approches de codage à temps lâche.
- DANTE : Leur objectif est d'étudier les structures, la dynamique et les performances des réseaux. Leurs applications incluent les sciences sociales computationnelles, les réseaux de communication, les neurosciences et les sciences des réseaux. Leurs travaux portent sur le traitement des signaux graphiques, l'apprentissage automatique, les réseaux temporels, les corrélations temporelles et structurelles, les phénomènes collectifs, ainsi que les algorithmes distribués pour améliorer les communications.
- HoWNet : L'équipe se concentre sur l'étude des réseaux sans fil, en particulier les réseaux WLAN, véhiculaires et IoT. Ils utilisent des modèles probabilistes, des algorithmes distribués et des approches d'apprentissage automatique. Leur objectif est d'améliorer les performances et l'efficacité énergétique des réseaux, en mettant l'accent sur les interactions avec les réseaux filaires. Les solutions sont validées par des simulations et des tests expérimentaux.
- MC2 : Leur objectif est de comprendre les algorithmes efficaces dans différents modèles de calcul, tels que la complexité algébrique, la dynamique symbolique, l'informatique quantique et la programmation moléculaire. Leurs recherches portent sur la complexité algébrique des circuits arithmétiques, la théorie des graphes pour concevoir des algorithmes efficaces, l'optimisation de l'information quantique avec des codes correcteurs d'erreur, et l'étude de la dynamique symbolique.
- Plume : L'équipe Plume se concentre sur les fondements de la correction de logiciel en combinant des méthodes de construction et de vérification de la correction. Ils utilisent des outils mathématiques tels que la théorie des catégories, le système Coq et la théorie de la complexité. Leurs recherches ont des applications spécifiques dans les domaines de l'extraction de programmes, la vérification de modèles et les connaissances exécutables pour la biologie. Ils contribuent également à la conception de langages de programmation et d'assistants de preuve.
- Roma : L'équipe se concentre sur la résilience des applications face aux pannes des supercalculateurs, en utilisant des modèles et des simulations probabilistes. Les stratégies d'ordonnancement prennent en compte les objectifs des utilisateurs ainsi que les contraintes de la plateforme. Des travaux sont également menés sur les solveurs pour l'algèbre linéaire creuse et les problèmes d'optimisation associés.

Durant mon stage j'ai travaillé au sein de l'équipe Avalon.

2.3 Présentation d'Avalon

2.3.1 Objectifs et travaux d'Avalon

L'équipe de recherche Avalon est une collaboration entre INRIA, le CNRS, l'ENS Lyon, l'Université Claude Bernard Lyon 1 et l'Université de Lyon. Son objectif à long terme est de contribuer à la conception de modèles de programmation prenant en charge différents types d'architectures, de les mettre en œuvre en maîtrisant les différents problèmes algorithmiques impliqués, et d'étudier leur impact sur les algorithmes au niveau des applications. Le défi consiste à déterminer le niveau d'abstraction adéquat pour fournir un modèle de programmation simple au développeur tout en permettant une exécution efficace sur une large gamme d'architectures.

L'équipe se concentre sur la conception de modèles, systèmes et algorithmes permettant d'exécuter des applications sur des ressources tout en respectant les contraintes des utilisateurs (prix, performance, etc.) et des administrateurs système (utilisation maximale des ressources, consommation d'énergie minimale, etc.). Ses domaines d'intérêt incluent le profilage et la modélisation des applications d'un point de vue énergétique, la gestion des données ainsi que la cartographie et la planification des applications.

En résumé, l'équipe Avalon vise à développer des modèles de programmation adaptés à différentes architectures, à optimiser l'utilisation des ressources et à minimiser la consommation d'énergie, tout en simplifiant le processus de développement des applications.

2.3.2 Présentation de quelques projet de recherches

Voici une présentation de trois projets de recherche abordant l'impact environnemental du numérique, la gestion des centres de données et la programmation parallèle pour les scientifiques.

- "Le vrai coût énergétique du numérique" par Anne-Cécile Orgerie, Laurent Lefèvre : Cet article scientifique présente l'impact environnemental des activités quotidiennes que nous considérons souvent comme banales, telles que regarder des vidéos en streaming ou discuter en visioconférence. Il présente les infrastructures complexes de réseaux, de centres de données et d'équipements terminaux qui se cachent derrière ces actions. L'article souligne l'importance de mesurer et de comprendre l'ampleur de l'impact environnemental de ces infrastructures, qui s'avère considérable et complexe à estimer. Il met également en évidence les pistes envisagées pour réduire la consommation énergétique de ces systèmes et pour sensibiliser les utilisateurs au véritable coût environnemental des équipements qu'ils utilisent.
- "Modeling, evaluating and orchestrating heterogeneous environmental leverages for large scale data centers management" par Vladimir Ostapenco, Laurent Lefèvre, Anne-Cécile Orgerie, Benjamin Fichel : Cet article présente une nouvelle approche pour modéliser, évaluer et orchestrer un ensemble étendu de leviers technologiques et logistiques visant à réduire l'impact environnemental des centres de données, qui sont des installations très énergivores.
- "Extensibility and Composability of a Multi-Stencil Domain Specific Framework" par Hélène Coullon, Julien Bigot, Christian Pérez : Cet article aborde le défi de la programmation parallèle dans un contexte où les architectures informatiques sont de plus en plus puissantes, hétérogènes et complexes. Les auteurs proposent d'utiliser des langages spécifiques à un domaine (DSL) pour faciliter la programmation parallèle. Ils présentent le Multi-Stencil Framework (MSF), qui combine un nouveau DSL avec des modèles de component-based programation. Le MSF permet de réutiliser facilement du code, de choisir différentes techniques de parallélisation, d'optimisation et d'implémentation.

2.3.3 Organisation du travail

Conditions du stage Mon stage s'est déroulé les bureaux du LIP, situés au site MONOD de l'ENS de Lyon.

Suivi par mon professeur J'ai travaillé sous la supervision de Yves Caniou, qui m'a conseillé tout au long du stage. Étant donné que nous travaillions dans des lieux différents, nous avons principalement communiqué par e-mail et par visioconférence. Cependant, nous nous sommes rencontrés environ une fois par semaine pour faire le point et discuter de sujets plus complexes en personne. À chaque rencontre, je devais préparer une présentation PowerPoint avec quelques slides pour présenter mes avancées depuis la dernière fois. Pour garder une trace écrite du travail accompli, Yves Caniou m'a également conseillé de prendre des notes chaque jour sous la forme de 5 points principaux.

Réunions / Groupe de travail Tous les membres du LIP se réunissent une fois par semaine pour discuter du travail accompli durant la semaine précédente. Au cours de ces discussions, j'ai pu apprendre beaucoup sur le monde de la recherche et sur les activités des membres du laboratoire.

2.3.4 Clarification vis-à-vis de l'INRIA

INRIA est un institut français de recherche en mathématiques et informatique, qui opère en tant qu'établissement public à vocation scientifique et technologique. Il est sous la tutelle du ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation, ainsi que du ministère de l'Économie et des Finances.

Mon stage a été rémunéré par l'INRIA, mais en pratique, j'ai effectué la totalité de mon stage dans les locaux du LIP.

2.4 Présentation des outils utilisés

LaTeX est un système de composition de documents utilisé principalement pour la création de documents scientifiques et techniques. Il offre un contrôle précis sur la mise en page et permet de produire des documents de haute qualité, tels que des articles de recherche, des thèses et des rapports.

Je l'ai utilisé pour rédiger ce rapport.

Cmake est un outil de gestion de la construction de logiciels qui permet de générer des Makefiles et des projets natifs pour différentes plateforme. Il offre une syntaxe simple et facilite le processus de compilation et d'installation des projets.

Je m'en suis servi pour installer différentes bibliothèques, mais aussi pour générer les Makefiles de mon projet.

Git est un système de contrôle de version distribué qui permet de gérer les modifications apportées aux fichiers d'un projet. Il facilite le travail collaboratif, la gestion des branches et le suivi des modifications. Git est largement utilisé dans le développement logiciel pour assurer la cohérence et la traçabilité des versions des projets.

Je m'en suis servi pour sauvegarder le projet et consulter les anciennes itérations du projet dans le cadre de débogage.

Le Git du projet est cette adresse :
https://forge.univ-lyon1.fr/YVES.CANIOU/budget-stochastic-wf

SimGrid est une bibliothèque libre et open-source utilisée pour développer des simulateurs d'applications distribuées ciblant des plateformes distribuées, qui peuvent à leur tour être utilisés pour prototyper, évaluer et comparer des configurations de plateforme pertinentes, des conceptions de système et des approches algorithmiques. Plus d'information sur SimGrid peuvent être trouvées sur sa page web https://simgrid.org/.

WRENCH est un bibliothèque libre et open-source qui facilite le développement de simulateurs pour les applications informatiques distribuées. Basé sur la bibliothèque de simulation SimGrid, WRENCH permet la simulation d'un environnement de calcul distribué en proposant des objets qui simulent des VM et des clusters avec leur système de gestion de tâches en batch *etc.* Il est notamment utilisé dans les domaines de la recherche, du développement et de l'éducation. Vous pouvez trouver plus d'information sur WRENCH sur sa page web https://wrench-project.org/.

Le sujet de ce stage portait principalement sur le développement d'un outil similaire à celui développé dans l'article scientifique initial, mais en utilisant WRENCH comme base au lieu de SimGrid. Cette évolution était nécessaire car l'outil de simulation basé sur SimGrid était devenu obsolète, au point de ne plus pouvoir être utilisé.

3 Introduction au sujet et mise en contexte

3.1 Sujet du stage et contexte

Article scientifique initial Le sujet de mon stage est basé sur l'article scientifique "Budget-aware scheduling algorithms for scientific workflows with stochastic task weights on IaaS Cloud platforms" [2] par Yves Caniou, Eddy Caron, Aurélie Kong Win Chang, Yves Robert .

Cet article présente des algorithmes pour déployer des workflows scientifiques sur des plates-formes Cloud IaaS avec des coûts et des ressources spécifiques. Les algorithmes doivent respecter un budget et optimiser leurs décisions de planification en fonction de la disponibilité des machines et du budget restant.

Pour réaliser cet article, Aurélie Kong Win Chang a développé des outils permettant à SimGrid de prendre en compte le budget lors d'une simulation. Chaque simulation dispose d'un budget dédié à la création et l'utilisation de VM. En effet, plus le nombre de VM créées est élevé et plus longtemps elles sont utilisées, plus le budget est consommé. On cherche à ne pas dépasser le budget alloué tout en réduisant le temps d'exécution global de l'ensemble du workflow, également connu sous le nom de makespan.

Pour cela, nous avons utilisé des heuristiques d'ordonnancement de tâches spécialisées. Ces heuristiques sont des variantes modifiées d'heuristiques connues telles que MinMin ou HEFT (voir section 4.4.5). Elles prennent en compte le budget lors de l'attribution des tâches à une ressource de calcul. De plus, ces heuristiques sont responsables de la création de nouvelles VM. Une fois de plus, le budget doit être pris en compte lors de la décision de créer ou non de nouvelles VM.

Il est important de noter que le simulateur ,utiliser pour l'article scientifique sur lequel ce stage est basé, était statique. Par conséquent, les heuristiques de l'article ne sont pas applicables à la partie pseudo-statique/dynamique du projet.

Développement d'un simulateur WRENCH Certains éléments du projet précédent qui utilisaient SimGrid étaient devenus obsolètes au point où il n'était plus utilisable. L'objectif de mon stage était de développer un nouveau simulateur qui utiliserait Wrench au lieu de

2. https://inria.hal.science/hal-03029318v1

SimGrid. De plus, des informations telles que le coût des VM ou leur puissance de calcul devaient être mises à jour pour mieux refléter l'environnement actuel.

Il a notamment fallu créer des outils de simulation qui n'étaient pas disponibles nativement dans WRENCH, tels que la prise en compte du budget ou la mise en place de services de cloud dont les capacités sont limitées uniquement par le budget.

Il a également fallu mettre en place des outils externes à la simulation pour pouvoir générer et assurer la reproductibilité des expériences, notamment en mettant en place un format de fichier permettant de sauvegarder les paramètres des expériences et de les reproduire facilement.

Les heuristiques ont également dû être adaptées pour être utilisables dans WRENCH, notamment celles utilisant un budget, qui ont nécessité des ajustements afin d'accéder à diverses informations qui ne sont pas facilement accessibles dans une simulation WRENCH classique.

Enfin, l'une des principales tâches consistait à mettre en place un système permettant d'attribuer un comportement stochastique aux tâches du workflow. Cela représente l'une des parties les plus conséquentes du projet, car WRENCH est par défaut entièrement déterministe en ce qui concerne le temps d'exécution des tâches.

3.2 Différence entre simulation et sous-simulation

Dans ce rapport, nous aborderons différents types de simulations à différentes échelles. Une simulation peut être amenée à lancer elle-même d'autres simulations qui ne généreront que des résultats partiels ou qui seront utiles uniquement dans le cadre de la simulation plus large. Nous appellerons ces simulations auxiliaires des sous-simulation, tandis que les simulations complètes qui génèrent des résultats finaux seront appelées simulation.

3.3 Workflows

3.3.1 Concept de workflows

Les workflows sont des structures utilisées pour représenter et gérer des processus de travail complexes. Ils se composent d'un ensemble de tâches et d'un réseau de dépendances entre elles. L'utilisation de workflows facilite la modélisation et l'automatisation des tâches.

Tâche Une tâche est un ensemble de calculs et d'opérations qu'une unité de calcul sera amenée à effectuer. Elle peut nécessiter l'accès à un certain nombre de fichiers pour se réalisé. De plus, dans la majorité des cas, une tâche générera un fichier contenant les résultats de son exécution.

Dépendances des tâches entre elles Dans un workflow, les tâches peuvent avoir des dépendances entre elles, ce qui signifie que l'exécution d'une tâche peut dépendre de l'achèvement d'une ou plusieurs autres tâches. Ces dépendances doivent être prises en compte lors de la planification et de l'exécution du workflow afin de garantir que les tâches sont exécutées dans le bon ordre et d'éviter les conflits.

Dépendances des tâches a des fichiers En plus des dépendances entre les tâches, les workflows peuvent également comporter des dépendances a des fichiers de données. Certaines tâches peuvent nécessiter l'accès à des fichiers en entrée. La gestion appropriée de ces dépendances a des fichiers est essentielle pour assurer la cohérence des données et la fiabilité de la simulation.

3.3.2 Format des workflows et wfcommons

Lors de la représentation et de la manipulation des workflows, il est courant d'utiliser des formats standardisés tels que Common Workflow Language (CWL), Workflow Description Language (WDL) ou encore WfFormat. Ces formats permettent de décrire de manière portable et indépendante du système de calcul les étapes, les dépendances et les paramètres des workflows. Le format utilisé durant ce projet est le WfFormat, qui se présente sous la forme d'un fichier JSON, et qui a été mis en place par la plateforme Wfcommons.

Wfcommons est une plateforme en ligne qui fournit plusieurs outils liés à la création et au traitement de workflows scientifiques, ainsi qu'une collection de workflows scientifiques disponible pour des tests. L'objectif de Wfcommons est de faciliter le partage et la réutilisation des workflows.

3.3.3 Remarques sur les spécificités de notre simulation

Workflow scientifique Les workflows utilisés dans ce projet sont des workflows scientifiques, caractérisés notamment par des tâches de longue durée pouvant prendre plusieurs heures pour s'exécuter. Il est également important de noter que nous utilisons des workflows scientifiques réels qui ont déjà été utilisés dans le cadre de recherches, plutôt que de générer des workflows aléatoires.

Tâches stochastiques dans les workflows Dans les workflows que nous utilisons, chaque tâche est associée à une durée d'exécution prédéfinie. Cependant, nous nous intéresseront a des tâches pouvant avoir une durée d'exécution qui n'est pas déterministe, mais qui suit une distribution probabiliste (typiquement une loi normale). Ces tâches sont appelées tâches stochastiques et introduisent un élément d'incertitude dans la simulation, car leur temps d'exécution peut varier à chaque exécution. Pour tenir compte de cette variabilité, des techniques de modélisation appropriées doivent être utilisées lors de la simulation des workflows scientifiques. Dans le cadre de notre simulation, toutes les tâches sont considérées comme stochastiques. Cependant, comme mentionné précédemment, les tâches des workflows que nous utilisons ont une durée prédéterminée. L'une des tâches à accomplir est donc de simuler le caractère aléatoire de ces tâches (voir 4.7.1).

3.4 Plateforme

3.4.1 Cloud

Un service de cloud est une plateforme qui fournit des ressources de calcule à la demande via Internet. Il est typiquement possible de demander des ressources supplémentaires en fonction des besoins. Il peuvent fournir différents type de services, certain cloud ne loue que des espaces de stockage tandis que d'autre fournissent des ressources de calcules ou les deux.

Ceux qui fournissent des ressources de calcules sont typiquement séparer en trois types : Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) et Software-as-a-Service (SaaS). Avec IaaS, les utilisateurs gèrent le système d'exploitation et les applications, tandis que le fournisseur s'occupe de l'infrastructure. PaaS permet aux développeurs de se concentrer sur le développement d'applications sans se soucier de l'infrastructure. SaaS fournit des applications entièrement gérées par le fournisseur accessibles via le web.

Dans le cadre de ce stage, le modèle le plus proche du nôtre est le modèle IaaS.

3.4.2 Ressources de calcul

Les ressources de calcule fournit par le cloud sont dans la majorité des cas sous la forme de Virtual Machine (VM). Les VM sont des environnements virtuels isolés qui permettent l'exécution de tâches de calcul de manière indépendante. Ces VM peuvent avoir un ou plusieurs cœurs de calcul sur lesquels exécuter plusieurs tâches simultanément, ou exécuter des tâches multithread qui peuvent exploiter plusieurs cœurs à la fois. Les VM peuvent également disposer d'espaces de stockage tels que de la RAM ou un disque dur.

3.4.3 Coût des VM par rapport au temps d'utilisation

Le coût de l'utilisation des ressources de calcul (ici des VM) est généralement indexé sur le temps d'utilisation, c'est-à-dire que plus longtemps une VM est utilisée, plus le coût sera élevé. De plus, les VM plus puissantes (en terme de calcul effectué par seconde) elles sont plus coûteuses. L'augmentation du prix est proportionnelle au gain de puissance de calcul, ce qui a tendance a rendre peut important le choix de la puissance d'une VM. Par exemple, il reviendra au même prix d'exécuter une même tâche dans une VM à bas prix mais moins puissance (donc pour plus longtemps), et dans une VM plus coûteuse mais plus puissante (donc pour un temps plus court).

La réduction du coût et le respect d'un budget sont les objectifs principaux de notre simulation. Dans le cadre de notre simulation, le coût des ressources de calcule sera indexé sur des services réels disponibles sur le marché, tels que Amazon EC2, Google Cloud ou OVH. Ce travail avait déjà été réalisé lors du projet précédent, mais il était important de le mettre à jour.

3.5 Modèle de stockage et de transfère de données

L'exécution d'un workflow sur une plateforme distribuée requiert également des outils de stockage des données, notamment lorsque ces données doivent être utilisées par plusieurs VM. Nous aurions put abordée le stockage des donnée de plusieurs manières.

La méthode la plus simple consiste à avoir un *datacenter* où une seule VM est dédiée au stockage. Toutes les données du workflow, qu'elles soient générées par des tâches ou fournies au début du workflow, sont stockées dans cette VM. Ainsi, si une VM a besoin d'un fichier, elle le trouvera nécessairement dans ce *datacenter*. De même, si une VM doit stocker un fichier, elle le fera sur la VM de stockage. Cette méthode a l'inconvénient de maintenir une VM constamment allumée qui ne sert qu'au stockage. Avec cette méthode, l'architecture réseau suivrais alors une architecture en étoile, comme illustré dans la figure 1 [3].

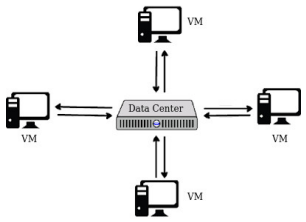

FIGURE 1 – topologie en étoile

Une autre méthode serait de laisser les VM allumées tant qu'un de ses fichier doit encore être utilisé, même si la VM qui l'a généré n'as plus de tâches a exécuter. Cependant, l'efficacité de cette méthode dépend fortement du workflow et peut être pire qu'un *datacenter* dans une mauvaise configuration.

Enfin, il est possible de stocker les fichiers de manière distribuée en les stockant sur les VM qui les génèrent ou les utilisent, mais en permettant la migration des données pour réduire le nombre de VM allumées. Cette méthode est beaucoup plus complexe notamment par rapport a la contention du réseau. Il devient aussi difficile de garantir la pérennité des données dans le temps. En effet, à chaque fois qu'une VM est éteinte, il faut s'assurer que tous les fichiers

3. L'image de la figure 1 provient du site https://waytolearnx.com/2019/06/topologie-reseau-en-etoile.html

qui devront encore être utilisés ont une copie déjà existante dans une autre VM. Dans le cas contraire, il faut utiliser une des ressources disponibles pour stocker ces données, tout en cherchant a minimiser le nombre de migration, qui peuvent prendre du temps.

Dans un souci de simplicité, et étant donné que c'est la méthode qui avait été choisie pour le simulateur de l'article initial, la méthode du *datacenter* a été adoptée pour notre simulateur. Cependant, avec davantage de temps, il aurait été intéressant de tester les autres approches.

3.6 Architecture réseau

Les différents éléments d'une plateforme de calcul sont inter-connectés au sein d'une architecture réseau. L'architecture du réseau a un impact sur le temps total de calcul workflow car elle peut influé sur le temps de transfère des données. Des paramètres tels que la bande passante des connexions ou le fait qu'elles puissent être utilisées par différentes VM en alternance peuvent avoir un impact négatif sur le temps total de calcul et ajouter des coûts.

4 Déroulement d'une simulation

4.1 Modélisation WRENCH

4.1.1 Simulation

Dans la bibliothèque WRENCH, une simulation est représentée par une classe. Tous les éléments à simuler, tels que les services de stockage et les différents fournisseurs de ressources de calcule, sont ajoutés à une instance de la classe Simulation. Une fois que tous les éléments à simuler ont été ajoutés, la simulation peut être lancée en utilisant une fonction membre de la classe. Enfin, les résultats de la simulation sont également générés par une fonction membre de la classe Simulation.

4.1.2 Comput services

Cloud comput service

présentation La classe Cloud Computing Service est l'une des classes qui héritent de la classe Compute Service. Un Cloud Computing Service a pour rôle de créer des VM et de mettre à disposition leurs bare metal services. Ces derniers sont les objets auxquels les tâches seront directement soumises.

Créations des VM Dans un service de cloud computing, il est possible de créer des VM à la demande, que ce soit avant ou pendant la simulation. De plus, il est possible de les éteindre de manière dynamique. Une fois qu'une VM est créée, elle doit être assignée à un Computing Service qui la prend en charge. Le nombre de VM pouvant être assignées à un Computing Service dépend des ressources allouées aux hôtes du service lors de la définition de la plateforme.

Contrôle limité sur les hôtes Il est important de noter qu'il est impossible de contrôler sur quel hôte physique une VM sera créée dans un service de cloud computing. Il est possible de configurer l'algorithme qui effectue le choix de l'hôte, mais cette configuration est très limitée. C'est cette limitation qui a conduit au passage à un Computing Service de type Virtualised Cluster comput service.

Virtualised Cluster comput services

Présentation La classe Virtualised Cluster Compute Service hérite de la classe Cloud Compute Service (et par conséquent de la classe Compute Service), et elle propose donc toutes les actions réalisables par un Cloud Compute Service. Cependant, un Virtualised Cluster Compute Service offre plus de contrôle supplémentaire sur les VM.

Attribution des VM à des hôtes spécifiques Dans un Virtualized Cluster Service, il est possible d'attribuer une VM à un hôte spécifique si celui-ci dispose des ressources nécessaires pour l'héberger. Dans notre simulation, nous considérons que les services proposant des VM ont des ressources pratiquement illimitées, car le budget constitue la véritable contrainte.

Migration des tâches Dans un Virtualised Cluster comput services il est aussi possible de faire migré une VM d'un hôte à un autre. Et ce même si la VM en question est en cours d'exécution d'une tâche, elle seras mise en pause pour la durée de la migration.

Hôtes comme approximations des différents type de services Dans la réalité, les fournisseurs de ressources de calcule proposent souvent des services avec des niveaux de puissance et de prix variables. Cependant, dans WRENCH, le seul niveau auquel la puissance de calcul peut être définie est au niveau des hôtes. Nous utilisons donc les hôtes comme une approximation de ces différents type de services. Par exemple, si une entreprise propose des machines virtuelles à faible coût mais avec une vitesse de calcul lente, et d'autres machines virtuelles avec une meilleure vitesse de calcul mais à un coût plus élevé, nous pouvons représenter cette situation par deux hôtes : l'un avec une meilleure vitesse de calcul et l'autre avec une moins bonne. Ensuite, nous déterminons le coût final de l'utilisation d'une machine virtuelle en fonction de l'hôte (qui a un prix par unité de temps) auquel elle est associée et de la durée d'utilisation.

Limitations de cette approximation Cependant, cette approximation des services par les hôtes limite le réalisme de la simulation sur certains aspects. Les connexions entre les services sont également définies au niveau des hôtes. Ainsi, dans la réalité, un service qui propose des VM peut être réparti sur plusieurs bâtiments, et la proximité entre eux ainsi que la qualité de leur connexion peuvent influencer les temps de transfert entre les VM de même puissance. Dans une simulation WRENCH normale, cette situation aurait été représentée par plusieurs hôtes avec différentes connexions entre eux. Cependant, avec notre approximation, les VM de même puissance sont associées au même hôte, elles sont donc considérés comme faisant partie du même système, et donc elles ne rencontrent aucun problème de latence, de saturation des moyens de communication, de débit faible, etc.

4.1.3 Réseau

Dans une simulation WRENCH, tous les éléments qui représentent des composantes physiques dans la réalité font partie d'un réseau. Les connexions de ce réseau sont spécifiées dans la plateforme, qui est définie par un fichier XML. Il est possible de définir le nombre de connexions pour un composant, la vitesse de la connexion et les éléments qui sont reliés par cette connexion. Il est important de noter que, dans le cadre des Compute Services, ces connexions sont définies via des hôtes qui représentent les ressources physiques du service. Chaque hôte peut avoir ses propres capacités de calcul et connexions.

4.1.4 Storage services

Un Storage services permet de gérer les opérations de stockage et de lecture de données. Il est constitué de plusieurs processus chargés de gérer les opérations concurrentes et utilise des algorithmes prédéfinis pour optimiser les communications réseau et l'accès aux disques. Il est important de noter que ce service ne propose pas par défaut les fonctionnalités avancées des systèmes de fichiers parallèles tels que Lustre, telles que la répartition entre les nœuds de stockage ou les serveurs de métadonnées dédiés. Cependant, le service de stockage peut être personnalisé pour intégrer ces fonctionnalités.

4.1.5 File Registry Service

Le File Registry Service est une classe fournie par WRENCH qui permet de traquer l'emplacement des fichiers afin de les rendre accessibles aux ressources de calcul. En général, une ressource de calcul doit passer par le File Registry Service pour rechercher l'emplacement du fichier, puis effectuer le transfert depuis le Storage Service.

4.2 Simulation du temps de calcul des tâches

Temps de calcul Le temps d'exécution d'une tâche sur une VM donnée dépend de plusieurs paramètres :

- Le temps de démarrage de la VM, s'il n'est pas déjà allumé lorsque la tâche est assignée.
- La vitesse de calcul de la VM, exprimée en opérations en virgule flottante par seconde (FLOPS).
- Le nombre de cœurs de la VM. Il est important de noter que ce paramètre n'affecte le temps de calcul d'une tâche individuelle que si celle-ci est parallèle. Cependant, dans le cadre de notre simulation, aucune tâche ne le sera.
- La quantité de calcul à effectuer pour terminer la tâche, exprimée en FLOPS.

Tous ces paramètres sont utilisés par WRENCH pour calculer le temps total que prendra une tâche depuis sa soumission jusqu'à sa finalisation.

Temps de transfert des données Pour qu'une tâche puisse être lancée, tous les fichiers requis doivent être préalablement copiés sur la VM qui exécutera la tâche. Le temps de transfert de ces fichiers dépend des paramètres suivants :

- La taille des fichiers à transférer.
- Le nombre de fichiers à transférer.
- Le débit maximal des connexions entre le lieu de stockage et la ressource de calcul.
- La disponibilité de la connexion entre le lieu de stockage et la ressource de calcul.

Espaces de stockage La nature de l'espace de stockage des fichiers dans le service de stockage (*Storage Service*) et la VM d'exécution a aussi un impact sur le temps d'exécution. Il peut s'agir de RAM (accès plus rapide) ou de disque dur (accès plus lent). Il n'y a pas de possibilité de paramétrage de la vitesse d'accès au-delà de ces deux options.

4.3 Déroulement d'une simulation statique

4.3.1 Application des heuristiques

Heuristique Une heuristique est un algorithme qui cherche à attribuer des tâches à des ressources de calcul. On mesure la qualité d'une heuristique par sa capacité à réduire le temps total d'un workflow ainsi que sa capacité à respecter le budget de l'expérience.

Application des heuristiques dans WRENCH WRENCH fonctionne selon un paradigme événementiel, ce qui signifie notamment que les heuristiques sont appliquées dès qu'une tâche est terminée. À ce moment-là, elles attribuent généralement toutes les tâches pouvant l'être en fonction des ressources de calcul disponibles. Ces heuristiques peuvent donc être appliquées plusieurs fois au cours d'une même simulation ou sous-simulation.

4.3.2 Déterminisme de la simulation WRENCH et implémentation de la stochasticité

Dans les workflows utilisés par WRENCH, les tâches ont toutes une quantité de calcul prédéterminée, et les hôtes ont une vitesse de calcule constante en fonction du temps. Cela signifie que l'exécution d'un workflow dans WRENCH est purement déterministe. Si l'on lance deux simulations du même workflow sur la même plateforme, on obtiendra exactement le même temps de calcul pour chaque tâche. Or, la tâche à réaliser lors de ce stage est de développer un simulateur pour des tâches stochastiques, dont le temps d'exécution peut varier aléatoirement entre deux simulations.

On pourrait penser qu'il suffit d'appliquer un degré d'aléa au workflow que la simulation utilise. Cependant, avec cette méthode, l'heuristique d'ordonnancement aurait connaissance du temps réel de la tâche (après application de l'aléa), car elle utilise les informations du workflow pour calculer le temps estimé de la tâche. Or, nous souhaitons que les heuristiques prennent une décision basée sur une durée estimée, alors qu'il pourrait y avoir une variation entre le temps estimé et le temps réel en pratique. Il a donc fallu adopter une méthode plus complexe, qui est présentée dans les paragraphes suivants et dans le pseudo-code 1.

Pour pouvoir implémenter la stochasticité, il faut faire appel à plusieurs sous-simulations (voir 3.2). Malheureusement, WRENCH a des variables statiques qui empêchent explicitement le lancement de plusieurs classes Simulations dans le même contexte. Pour contourner ce problème, nous avons dû utiliser un fork pour chaque sous-simulation, ce qui a compliqué le transfert de données entre les deux sous-simulations. La classe SavedOrdonnancement (voir 4.7.3) a été mise en place en partie pour résoudre ces problèmes.

Génération de l'ordonnancement à partir du temps théorique Pour pouvoir simuler le fait que l'heuristique n'ait pas accès au temps réel de l'exécution d'une tâche, il faut commencer par faire une première sous-simulation (qu'on appellera SSM1), celle-ci suivra un workflow "théorique" (qu'on appellera WF1) dont les temps de calcul sont des estimations du temps réel. La SSM1 générera l'ordonnancement théorique du WF1, comme si les taches n'était pas stochastiques. Autrement dit comme si toutes les tâches avaient finalement eu exactement le temps d'exécution prévu initialement.

Application d'aléa On génère ensuite un deuxième workflow (qu'on appellera WF2) en appliquant de l'aléa a WF1. Cet ajout d'aléa simule la variation de temps dans l'exécution des tâches "en pratique". Autrement dit, on peut considérer que WF1 contient des temps de calcul estimés, et WF2 les temps de calcul réels qui seraient normalement mesurés au cours de l'exécution.

Implémentation de la sous-simulation du workflow stochastique On crée ensuite une deuxième sous-simulation (SSM2) qui utilisera WF2. Mais SSM2 ne calculera pas de nouvel ordonnancement, elle se contentera de suivre celui qui a été généré par la SSM1. De cette manière, on sépare la prise de décision, et les informations réelles sur le temps d'exécution d'une tâche. On peut faire cette séparation car il n'y a pas de retour dynamique du temps réellement pris par une tâche.

La réaplication par SSM2 de l'ordonnancement généré par SSM1 suit l'algorithme présenté dans le pseudo code 2 :

Algorithm 1: Double simulation

```
workflowInit : Le workflow initial de la simulation (WF1).
launchSimulation2(workflow, ordonnancement) : Lance la deuxième simulation qui
 applique l'ordonnancent donné sur le workflow donné..
pid ← fork()
if pid of parent child process then
    ordonnancementSimulation1 ← launchSimulation1(workflow)
    send ordonnancementSimulation1 to parent process
if pid of parent process then
    wait for child process to finish
    ordonnancementSimulation1 ← get ordonnancementSimulation1 from the parent
     process
    workflowAlea ← randomize(workflow)
    simResults ← launchSimulation2(workflowAlea, ordonnancementSimulation1)
    return simResults
```

Algorithm 2: Applicaiton de l'ordonnancement générer par SSM1

```
previousSimulationTaskList : la liste des tâches dans l'ordre exécuté par SSM1.
for taskID ∈ previousSimulationTaskList do
    for task ∈ readyTasks do
        if taskID == task.getID then
            earliestTask ← null
            for cs ∈ computeServices do
                taskToDoNextForThisVm ←
                 getEarliestNonScheduledTaskForThisVM(cs)
                if core_utilization_map[cs] == 0 & taskToDoNextForThisVm ==
                 task.getID then
                    assignTaskToService(task, cs)
return
```

4.4 Déroulement d'une simulation pseudo-statique/dynamique

4.4.1 Différence entre pseudo-statique et dynamique

Dans une simulation pseudo-statique, on prend en compte les résultats de la simulation au cours de celle-ci. Par exemple, si les premières tâches de la simulation ont coûté en moyenne plus cher que ce qui était attendu(car les taches ont duré plus longtemps), on attribuera et créera des machines virtuelles de manière plus économe afin de respecter le budget. Une simulation dynamique se comporte de la même manière, à ceci près qu'il est également possible de supprimer ou d'ajouter des tâches pendant la simulation.

Ces deux types de simulation n'ont pas été implémentés au moment de la remise de ce rapport. De plus, s'il avait été envisagé de réaliser une simulation avec des heuristiques d'ordonnancement dynamique au début du stage, il semble peu probable que j'aie assez de temps pour le faire.

4.4.2 Simulation en phases phases multiples

Les simulations pseudo-statiques et dynamiques utilisent le même principe qu'une sous-simulation statique, mais elles lancent beaucoup plus de sous-simulation. Elles doivent en lancer une à chaque fois qu'elles reçoivent un retour d'information sur le temps d'exécution des tâches, notamment lorsqu'une tâche se termine ou lorsqu'une tâche aurait dû se terminer mais ne l'a pas encore fait.

4.4.3 Problématiques liées à la minimisation du temps de démarrage des VM

Dans les exemples suivants, on considérera que le temps de transfert des fichiers entre les VM est nul.

Minimisation du temps de démarrage des VM Un des éléments à simuler qui était ignoré dans la simulation statique est le temps de démarrage des VM. En effet, dans nos simulations statiques, on considérait les VM comme déjà lancées au démarrage de la simulation. C'était le cas car nos heuristiques statiques étaient aussi non budgétaires, et donc ne pouvaient pas créer de VM en cours d'exécution. Cependant, dans nos simulation pseudo-dynamique ou dynamique, de nouvelles VM peuvent être créées au cours de la simulation en fonction des tâches à exécuter car nous utilisons des heuristiques dynamique budgétaires.

Si une VM est lancée au cours de la simulation, le temps qu'il lui faut pour devenir opérationnelle peut avoir un impact sur le temps total de calcul. Par exemple, dans le workflow décrit dans la figure 2. Si l'on considère que toutes les tâches ont la même durée, le temps de mise en place de la seconde VM s'ajoutera systématiquement au temps total de calcul, comme illustré dans la figure 3. Dans cet exemple, on voit que le temps de démarrage de la VM 2 s'ajoute au temps total de calcul. Le temps de démarrage d'une VM est de l'ordre de deux minutes, ce qui peut avoir un impact important sur le temps total de calcul. Pour minimiser cet impact, on peut décider de lancer les VM en avance pour qu'elles soient opérationnelles au moment où on en a besoin. Cependant, il faut faire attention à ne pas les démarrer trop tôt pour minimiser les coûts d'utilisation des VMs.

Pour résoudre ce problème, on considérera que lors de la première sous-simulation, le temps de démarrage des VM est nul. De cette manière, on peut enregistrer le temps "idéal" de démarrage des VM, c'est-à-dire l'heure à laquelle elles devraient être fonctionnelles pour minimiser les coûts des VM et le temps de calcul total. Lors de la deuxième sous-simulation, on lance le démarrage des VMs au temps indiqué par la première sous-simulation, auquel on soustrait le temps de démarrage de la VM en question (voir la figure 4).

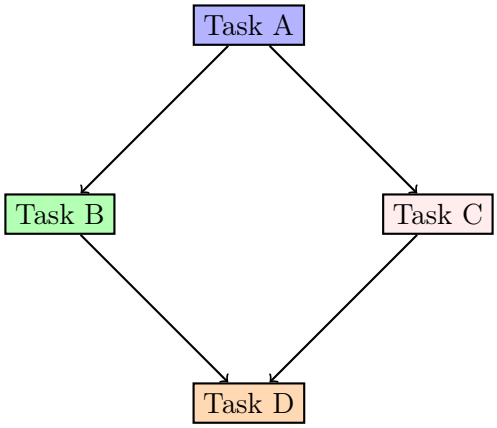

FIGURE 2 – Example de Workflow

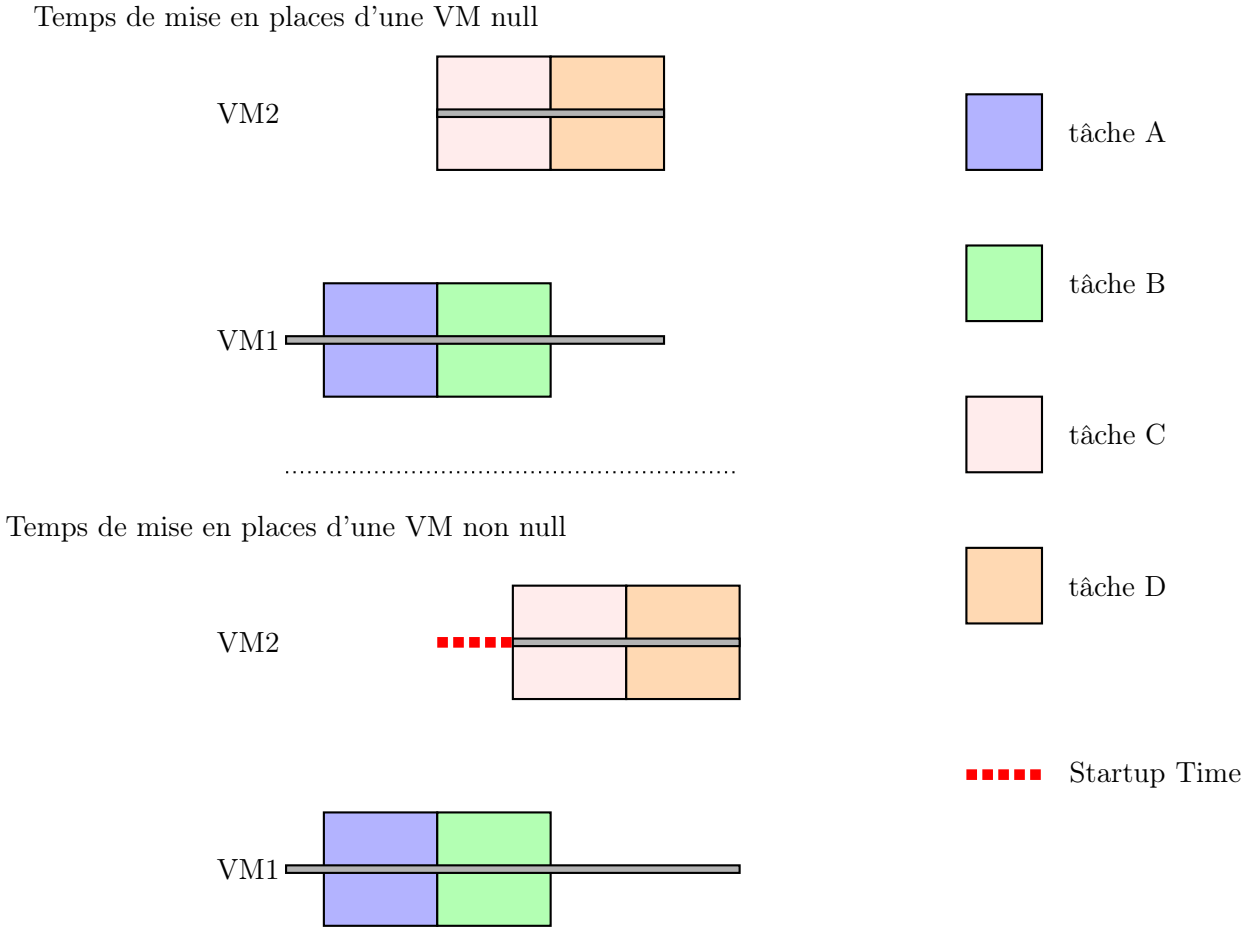

FIGURE 3 – Différence de prise en comte du temps de démarage des VM

Différents cas de figure liés à la stochasticité des tâches Nous travaillons avec des tâches stochastiques, ce qui signifie que l'heure idéale de démarrage d'une VM est également variable à cause de l'aléa des tâches. Par exemple, dans le workflow décrit dans la figure 2, avec des tâches A, B, C et D, où les tâches B et C dépendent des fichiers générés par la tâche A, et où la tâche D dépend des fichiers générés par les tâches B et C. L'ordre d'exécution des tâches est décrit dans la figure 4, on voit la VM 1 qui exécutera la tâche A, puis la tâche B une fois que la tâche A sera terminée. Lorsque la tâche A sera terminée, la VM 2 sera créée pour exécuter la tâche C, puis elle exécutera la tâche D. Dans cette situation, nous décrirons trois cas de figure possibles :

- La tâche A prend le temps prévu initialement. Dans ce cas décrit par la figure 4, aucun changement n'a à être fait par rapport à la première sous-simulation, car dans la deuxième sous-simulation la VM 2 commence au même moment que lors de la première sous-simulation.
- La tâche A prend moins de temps que prévu. Dans cette situation, il y a deux sous-situations possibles :
  - La différence de temps entre la fin prévue de la tâche A est plus faible que le temps de démarrage de la VM. Dans ce cas décrit par la figure 5, aucun changement n'a à être fait, car au moment de la fin de la tâche A dans la deuxième sous-simulation, la VM 2 est déjà en cours de démarrage. Ainsi, la tâche sera exécutée à l'heure prévue initialement.
  - La différence de temps entre la fin prévue est plus grande que le temps de démarrage de la VM. Cette situation est décrite dans la figure 6. Contrairement à la situation précédente, au moment de la fin de la tâche A dans la deuxième sous-simulation, la VM 2 n'est pas encore lancée ou en cours de démarrage. Il est donc possible de lancer le démarrage de la VM 2 plus tôt que prévu pour réduire le temps total de calcul. Ce changement aurait également un impact sur la tâche D, dont le temps de départ devrait être révisé à la baisse en fonction du temps gagné par rapport à la prédiction de la première sous-simulation.
- La tâche a pris plus de temps que prévu. Cette situation est décrite dans la figure 7. On voit que la VM 2 de la deuxième sous-simulation s'est lancée au même moment que dans la première sous-simulation, alors que la tâche C n'est pas prête au lancement. Il faudrait aussi ajouter au temps prédit de départ de la tâche D la différence entre le temps prévu d'exécution de la tâche A et son temps réel d'exécution.

Minimisation du temps de calcul de la simulation Il se peut que la variation de la durée d'une tâche ait un effet important sur l'ordonnancement des tâches, au point où simplement modifier les temps de départs des VM ne soit pas suffisant pour obtenir un ordonnancement efficace. Ce genre de situation peut arriver si l'ordre ou les VM d'exécution doivent être changés pour être plus efficaces. Dans ce contexte, il faudrait relancer tout le processus de calcul de l'ordonnancement avec les deux sous-simulations distinctes, mais en utilisant les valeurs réellement obtenues jusqu'à présent. Cependant, ce processus peut être lourd en termes de calcul. Des alternatives seront utilisées pour alléger le calcul. Par exemple, on pourrait relancer une nouvelle sous-simulation uniquement si la variation de la durée de la tâche dépasse une certaine limite préalablement fixée. De cette manière, on évite de refaire la simulation si elle a peu de chances d'être différente de celle que l'on a déjà réalisée. On peut également ignorer les parties de la simulation qui ont déjà été ordonnancées. De plus, il est possible de lancer la nouvelle simulation uniquement sur une partie du workflow total, cette méthode est appelée le "Window scheduling". De cette manière, on évite de calculer des éléments de l'ordonnancement qui seraient dans un futur lointain et qui ont peu de chances de rester corrects. Cependant, une implémentation naïve de cette méthode risque de faire des erreurs en raison de son manque

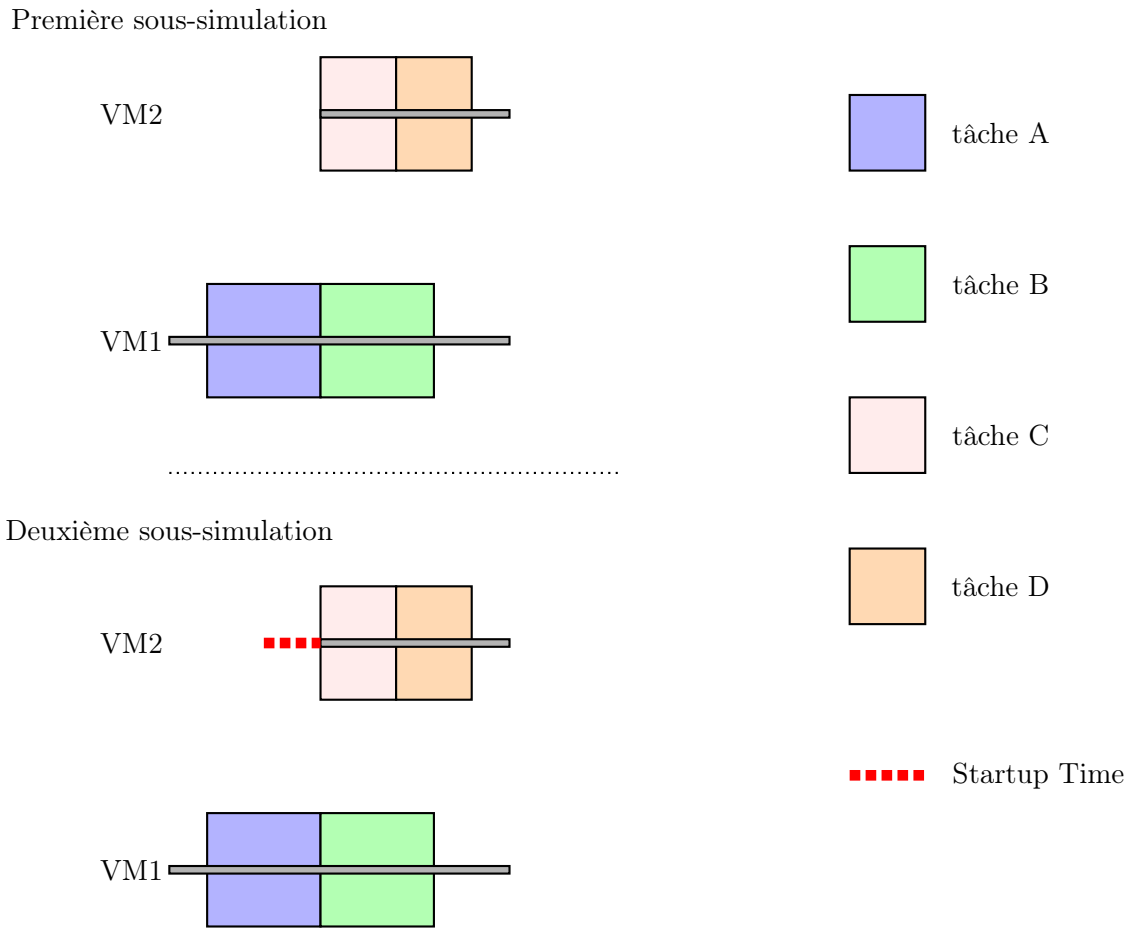

FIGURE 4 – Simulation Sans Alea

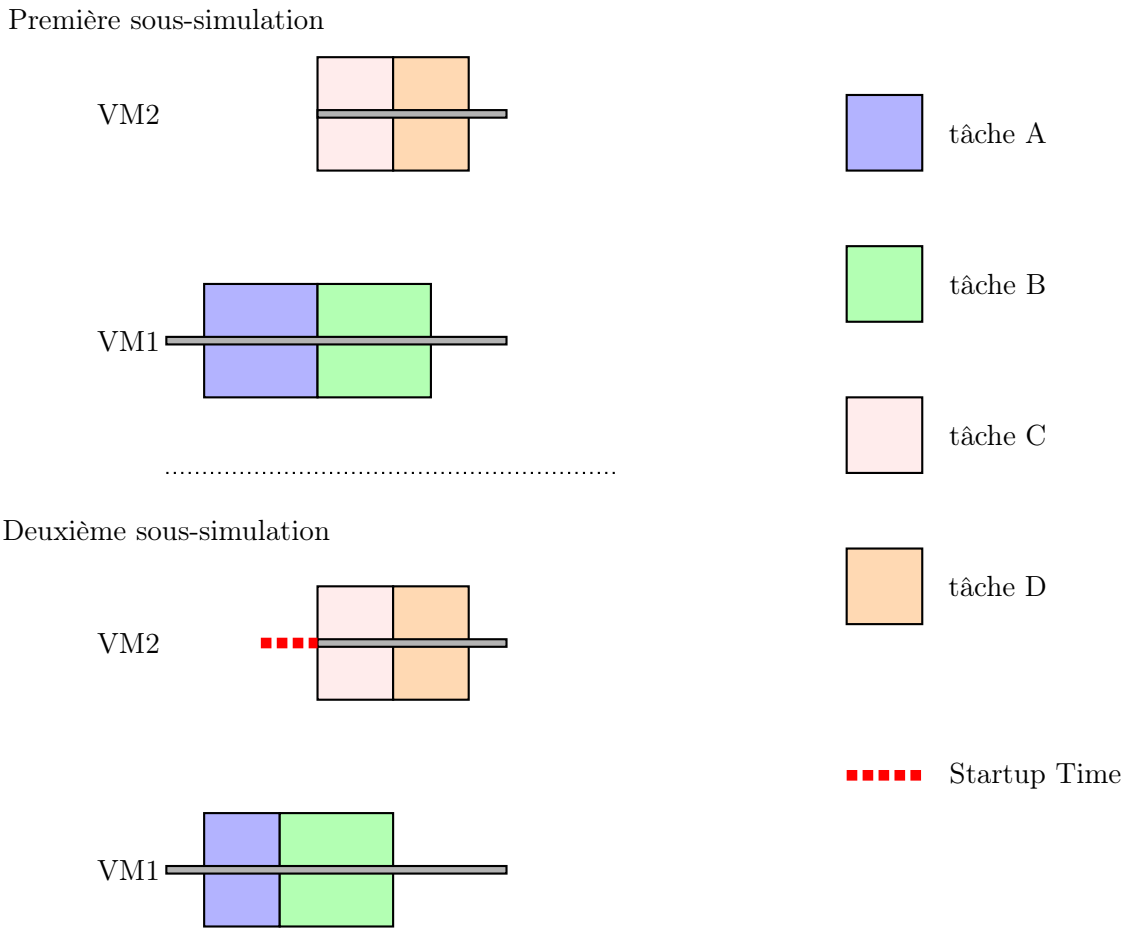

FIGURE 5 – Simulation Tache A Plus courte de peut

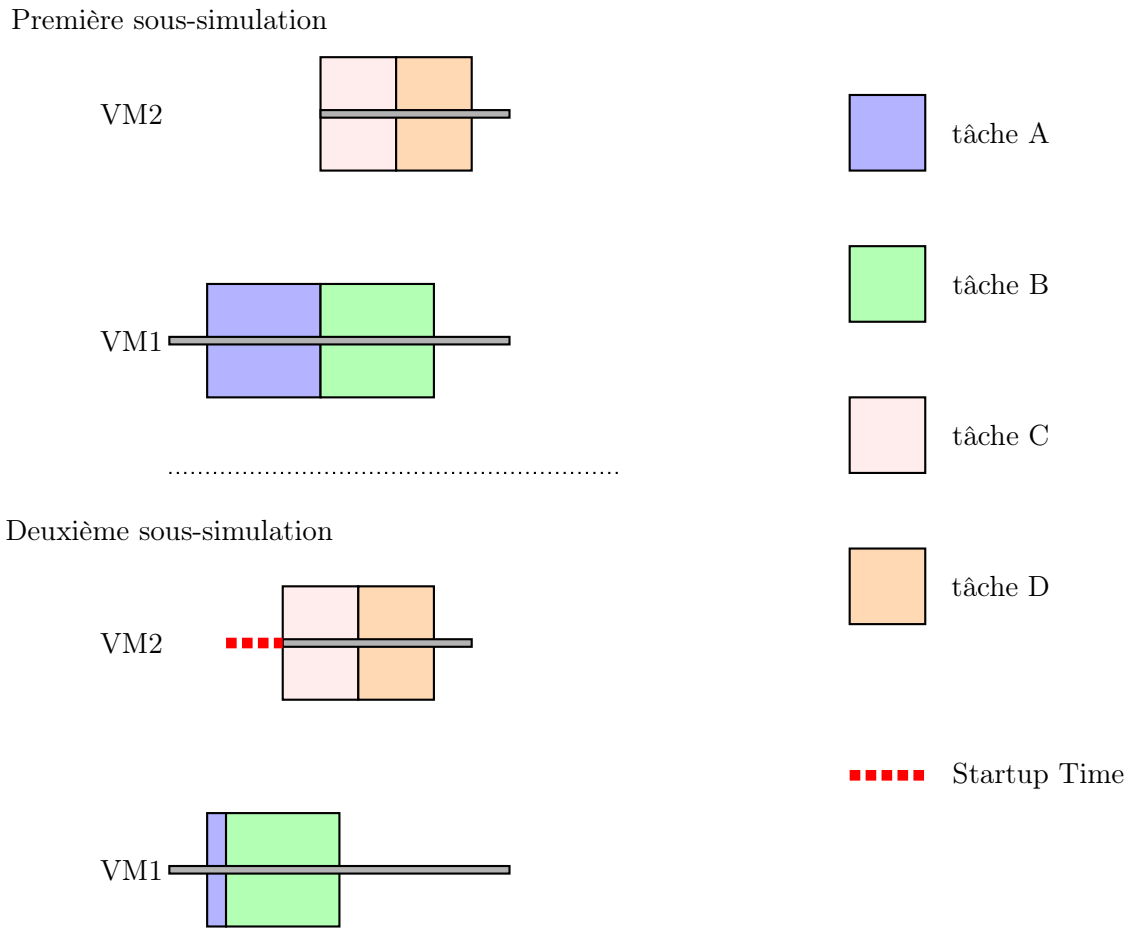

FIGURE 6 – tâche A plus courte de peu

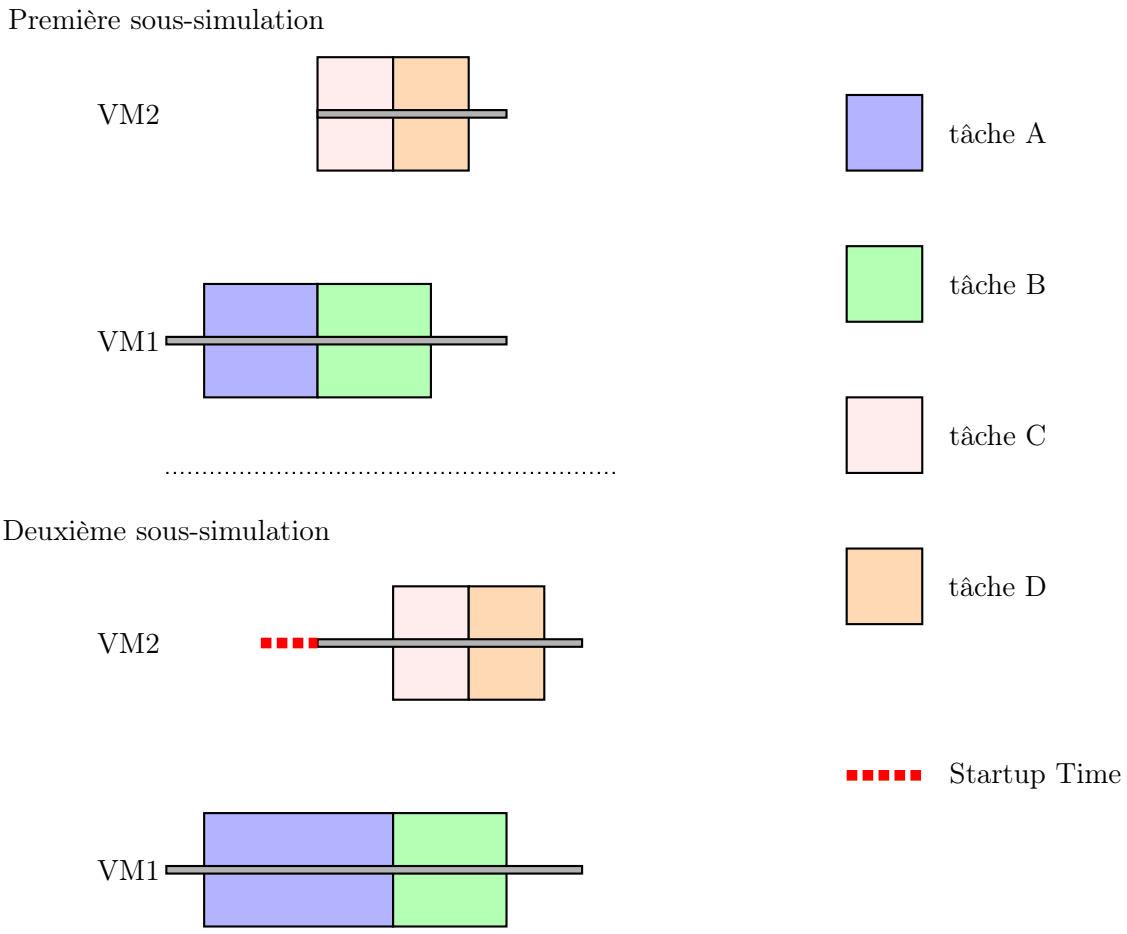

FIGURE 7 – Simulation Tache A plus Long

de perspective sur l'intégralité de la simulation. Il est notamment possible qu'elle ne priorise pas le chemin critique, la suite de tâches à calculer qui définit une limite minimale au temps de calcul total.

4.4.4 Description des fonctions et variables utilisées

- readyTaskList : Une liste de toutes les tâches pouvant être attribuées à une ressource de calcul. Toutes les tâches dont elles dépendent ont été exécutées et tous les fichiers dont elle a besoin existent.
- computeServices : Une liste des ressources de calcul.

- assignTaskToService(task, cs) : Cette fonction attribue la tâche "task" à la ressource de calcul "cs". Elle supprime également la tâche de la liste des tâches disponibles ainsi que la liste des Compute Services disponibles.
- estimateCompletionTime(task, cs) : Calcule une estimation du temps que prendrais une tâche sur une ressource de calcul donnée.
- checkResources(core_utilization_map) : Vérifie qu'il y a au moins une ressource de calcul disponible.
- getEarliestNonScheduledTaskForThisVM(cs) : Renvoie la prochaine tâche qu'il faudrait faire exécuter sur la VM "cs" pour respecter l'ordonnancement de la première sous-simulation.
- calculateExecutionTime(task, cs) : Calcule le temps total d'exécution d'une tache en prenant en compte le temps de calcule et de transfère des données.
- task.setEST() : Associe un temps avec une tâche dans le but de triée les tâches plus tard.

4.4.5 Heuristiques de références

Utilisation de ces heuristiques Ces heuristiques sont largement connues et utilisées par de nombreux acteurs du domaine des calculs distribués. Cependant, elles ne prennent pas en compte le budget, contrairement aux heuristiques que nous utiliserons pour générer les résultats attendus de ce stage. Ces heuristiques de référence ont été implémentées afin de tester l'infrastructure mise en place avec WRENCH. Elles servent également de base pratique pour tester les heuristiques budgétaires qui s'appuieront sur ces heuristiques de référence.

RoundRobin L'heuristique RoundRobin (qui est décris dans le pseudo code 3) parcourt les ressources de calcul et attribue les tâches dès qu'un service a un cœur disponible. Elle reprend son parcours à partir de la dernière ressource de calcul parcourue lors de la dernière distribution afin d'homogénéiser l'utilisation des ressources de calcul.

MCT L'heuristique MCT (Minimum Completion Time), qui est décrite dans le pseudo code 4, cherche la tâche nécessitant le moins de travail à exécuter et l'attribue à une ressource de calcul quelconque.

MinMin L'heuristique MinMin (qui est décris dans le pseudo code 5) consiste à calculer le temps de calcul de toutes les tâches pour chaque VM disponible, puis à attribuer les tâches en fonction des paires les plus rapides. Le pseudo code suivant décrit plus précisément son fonctionnement.

Algorithm 3: RoundRobin

```
1  readyTaskList : La liste des tâches qui sont prêtes pour être attribué à une VM
2  computeServices : La liste des VMs
3  while readyTaskList is not empty do
4      RRindex ← 0
5      for task ∈ readyTaskList do
6          assignTaskToService(task, computeServices[RRindex])
7          RRindex ← RRindex + 1
8  return
```

Algorithm 4: MCT

```
1  readyTaskList : La liste des tâches qui peuvent etre attribuer a une VM
2  computeServices : La liste des VMs
3  for cs ∈ computeServices) do
4      if taslList is empty then
5          return
6      selectedTask ← NULL
7      minCompletionTime ← ∞
8      for task ∈ readyTaskList do
9          if task.getFlops() < minCompletionTime then
10             minCompletionTime ← completionTime
11             selectedTask ← task
12     assignTaskToService(selectedTask, cs)
13 return
```

Algorithm 5: MinMin

```
1  readyTaskList : La liste des tâches qui peuvent être attribuer a une VM
2  computeServices : La liste des VMs
3  while readyTaskList is not empty do
4      for task ∈ readyTaskList do
5          minCompletionTime ← ∞
6          selectedService ← null
7          for cs ∈ computeServices do
8              completionTime ← estimateCompletionTime(task, cs)
9              if completionTime < minCompletionTime then
10                 minCompletionTime ← completionTime
11                 selectedService ← cs
12         assignTaskToService(task, selectedService)
13 return
```

HEFT L'algorithme HEFT (Heterogeneous Earliest Finish Time) est décrit dans le pseudo code 6. Pour chaque tâche, il estime le temps de calcul nécessaire sur chaque machine (lignes 4 à 11). Cette estimation est basée sur le temps de calcul de la tâche sur la machine ainsi que le temps de transfert des données nécessaires vers cette machine. Les tâches sont ensuite triées par ordre décroissant de leur temps d'exécution estimé le plus long sur n'importe quelle machine(ligne 12). Les tâches les plus longues sont planifiées en premier. Ensuite, les tâches sont assignées à la machine disponible qui minimise le temps de fin estimé de cette tâche(lignes 13 et 14).

Algorithm 6: HEFT

```
1  readyTaskList : La liste des tâches qui peuvent être attribuer a une VM
2  computeServices : La liste des VMs
3  while checkResources(core_utilization_map) and readyTaskList is not empty do
4      taskTable ← NULL for task in readyTaskList do
5          minCompletionTime ← ∞
6          selectedCS ← NULL
7          for cs in computeServices do
8              executionTime ← calculateExecutionTime(task, cs)
9              if executionTime < minCompletionTime then
10                 minCompletionTime ← executionTime
11         taskTable.add(task, selectedCS, minCompletionTime)
12     Sort tasks in taskTable based on decreasing order of task's EST
13     for taskAndCSintaskTable do
14         assignTaskToService(taskAndCS.task, taskAndCS.cs)
15 return
```

4.4.6 Heuristique avec prise en compte du budget

Au moment du rendu de ce rapport, seules les heuristiques qui ne prennent pas en compte le budget ont été mises en place.

Certaines heuristiques prennent en compte le budget à la fois pour demander au service de cloud de créer de nouvelles VM et pour ordonnancer les tâches. Leur priorité est généralement de créer un ordonnancement qui respecte le budget en premier lieu, puis d'améliorer le temps total de calcul par la suite. Les heuristiques avec budget qui seront implémentées pendant ce stage sont des variations de MinMin et HEFT, qui sont décrites dans le projet initial [4]. Elles fonctionnent de manière similaire à leurs équivalents non budgétaires, mais en plus d'estimer le temps de calcul d'une tâche, elles estiment également son coût total. Elles utilisent également un système de cagnottes pour décider s'il faut créer ou non une nouvelle VM pour une tâche donnée.

4.4.7 D'autres heuristiques

Il existe de nombreuses autres heuristiques qui n'ont pas été mises en place dans le cadre de ce stage, notamment d'autres heuristiques prennent en compte les ressources nécessaires pour exécuter les tâches. Par exemple, l'heuristique "Sufferage" calcule la différence entre le temps

4. HEFT et MinMin sont décrites dans la section "4.1 MinMinBudg and HEFTBudg" à la 7ème page

d'exécution actuel d'une tâche et le temps d'exécution restant pour déterminer sa priorité. Mais il existe aussi des heuristiques qui prennent en compte d'autres paramètres, tels qu'une *deadline* associée aux tâches. Par exemple, l'heuristique « EDF » (Earliest Deadline First) sélectionne la tâche ayant l'échéance la plus proche pour être exécutée en premier.

4.5 Expérimentation

4.5.1 Plateforme expérimental de calcul

Pour simuler une grande quantité de scénarios je dois utiliser des plateformes qui permettent de lancer un grand nombre de simulations a la fois.

Graal J'utiliserai notamment Graal, qui sert principalement de serveur web pour l'équipe de recherche, mais qui dispose également d'une grande puissance de calcul. Il est utilisé par l'équipe Avalon dans le cadre de leurs recherches.

Grid5000 Il est probable que j'utilise également Grid5000, une infrastructure de recherche et développement dédiée aux systèmes informatiques distribués à grande échelle. Grid5000 propose des clusters inter-connectés répartis sur plusieurs sites à travers la France, et il constitue un outil essentiel pour la recherche. Au total, Grid5000 compte 800 nœuds de calcul utilisant plus de 15 000 cœurs.

4.5.2 Architecture logicielle de traitement des expériences

Nos expériences sont indépendantes les unes des autres, nous pouvons donc les exécuter en parallèle pour gagner du temps. Pour pouvoir lancer un grand nombre d'expériences simultanément, j'ai dû adapter le système qui avait été mis en place pour l'article de recherche. Il consiste à avoir trois dossiers qui organisent les simulations à mener. Un dossier contient les simulations qui n'ont pas encore été exécutées, un autre contient les simulations en cours d'exécution, et le dernier contient les simulations dont l'exécution s'est terminée. Cette méthode permet de contrôler et d'observer l'avancement des exécutions même lorsque des exécutions sont en cours. Il devient également possible de relancer facilement celles n'ayant pas pu terminer pour un problème quelconque, ou d'en ajouter de nouvelles "à la volée".

4.5.3 Image docker

Pour simplifier le lancement de ces simulation sur des machines externe il est prévue d'implémenter une image docker. Elle permettrait de répliquer un environnement d'exécution qui serais portable dans sur la plupart des environnements utiliser sans avoir a se préoccuper de l'installation des différents dépendances du projet.

4.6 Résultats et analyse des expériences

Je suis toujours dans la phase de développement, donc je n'ai aucun résultat d'expérience pour l'instant. Pour mieux expliquer la manière dont les résultats seront présentés et comment ils pourront être analysés, j'ai pris comme exemple des graphes provenant de l'article scientifique initial. Les graphes générés par ma simulation seront présentés de manière similaire, les résultats devraient également correspondre pour la simulation statique.

Différentes données générées par la simulation La simulation WRENCH permet de récupérer beaucoup d'informations sur la manière dont elle s'est déroulée, mais nous avons dû développer des outils pour certaines informations, comme le budget, qui ne sont pas proposées par WRENCH. Nous séparerons ces résultats par heuristique afin de pouvoir les comparer entre elles. Dans notre cas, nous nous intéresserons principalement à la consommation du budget, ainsi qu'à la question de savoir s'il a été respecté ou non. Nous examinerons également le temps total requis pour exécuter un workflow. D'autres informations peuvent être utiles pour mieux comprendre le comportement de certaines heuristiques, telles que le pourcentage d'utilisation des machines virtuelles (VM) ou le nombre de VM créées.

4.6.1 Présentation des résultats du précèdent projet

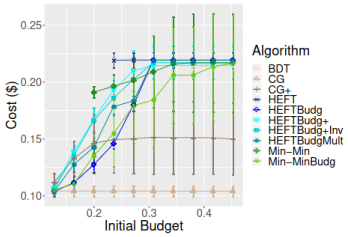

FIGURE 8 – Coût / Budget

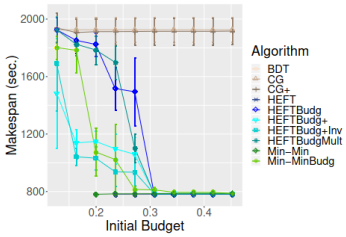

FIGURE 9 – Temps / Budget

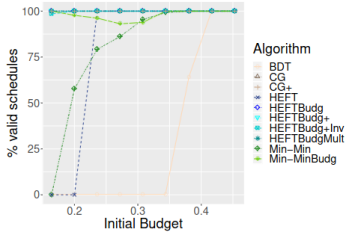

FIGURE 10 – Validité / Budget

Coût / Budget Dans la Figure 8, on peut voir l'évolution du coût (le budget utilisé) en fonction du budget autorisé. On remarque que presque toutes les heuristiques voient leur coût augmenter lorsque le budget augmente. C'est un résultat attendu, car les heuristiques sont plus conservatrices de leurs ressources lorsque leur budget est faible

Temps / Budget Dans la Figure 9, on peut observer l'évolution du temps total de calcul en fonction du budget autorisé. On constate que la plupart des heuristiques deviennent plus rapides lorsque le budget augmente, mais certaines heuristiques évoluent de manière plus rapide. Par exemple, HEFTBudg+ nécessite une augmentation de budget relativement faible pour obtenir une grande amélioration. En revanche, HEFTbudg nécessite une augmentation de budget plus importante pour obtenir une amélioration significative du temps.

Ce résultat est également assez prévisible, dans la mesure où un budget plus grand permet aux heuristiques d'utiliser davantage de ressources.

Validité / Budget Dans la Figure 10, on peut voir l'évolution de la validité des heuristiques (le pourcentage des ordonnancements qui respectent leur budget) en fonction de l'augmentation du budget autorisé. Les résultats sur ce graphique sont plus hétérogènes, mais on remarque que la majorité des heuristiques fournissent systématiquement un ordonnancement valide. Toutefois, il est surprenant de constater que l'heuristique MinMinBudg peut perdre en validité alors que son budget augmente.

4.7 Classes outils

Commentaires sur l'architecture des classes J'ai dû créer différentes classes pour ajouter des fonctionnalités à WRENCH. Ces classes sont assez simples, et la plupart d'entre elles servent simplement à stocker et traiter des informations obtenues via WRENCH. Elles ne

s'inscrivent pas dans une architecture de classe complexe, il ne m'a donc pas semblé pertinent de les présenter sous la forme d'un diagramme de classes.

4.7.1 RandomWorkflowModification

Cette classe stocke les variables utilisées pour la génération de l'aléa : la déviation standard, la moyenne et la seed. Elle utilise ces informations pour générer un nouveau workflow à partir d'un workflow original en appliquant de l'aléa au nombre de calculs nécessaires pour exécuter une tâche.

4.7.2 ReadConfig

Cette classe permet de sauvegarder les paramètres d'une simulation sous forme d'un fichier JSON. Elle permet également de charger ces paramètres à partir d'un fichier.

Cette classe facilite la reproductibilité des résultats pour les rendre vérifiable. Le format JSON est préféré car il offre une lisibilité accrue par rapport aux paramètres des expériences de l'article de recherche, qui étaient définis dans une ligne de commande.

4.7.3 SavedOrdonancement

Cette classe permet de sauvegarder l'ordonnancement d'une sous-simulation sous forme d'une liste de tâches et de Computing Services dans l'ordre où ils ont été attribués.

Bien qu'il aurait été possible d'utiliser les outils de récupération des résultats fournis par WRENCH, les fichiers générés peuvent être volumineux et le traitement de ces fichiers peut prendre du temps. C'est d'autant plus le cas pour les simulations pseudo-statique et dynamique qui doivent obtenir les résultats de plusieurs sous-simulation.

La classe SavedOrdonancement offre également la possibilité de sérialiser et de désérialiser ses données sous forme d'une chaîne de caractères. Cette fonctionnalité a été mise en place pour permettre la communication de données entre différentes sous-simulation lorsqu'elles sont exécutées dans des forks différents.

5 Conclusion

5.1 Récapitulatif du rapport

Dans ce rapport, j'ai fait un récapitulatif de mon environnement de travail au sein du LIP et de l'ENS, en expliquant leur contexte institutionnel et en présentant leurs activités de recherche. J'ai ensuite exposé le sujet de mon stage qui est de mettre en place un simulateur utilisant Wrench pour simuler l'exécution d'un workflow sur une plateforme de calcul distribuée en utilisant différentes heuristiques. Pour atteindre cet objectif, j'ai dû apprendre à utiliser Wrench et développer mes propres outils. J'ai développé le simulateur de manière à pouvoir lancer plusieurs sous-simulations qui produisent des résultats intermédiaires, lesquels sont utilisés pour générer les résultats finaux de l'expérience. L'objectif principal de ce simulateur était de mettre à jour les résultats décrits dans un article scientifique développer dans l'équipe Avalon du LIP. Je doit encore introduit une nouvelle méthodes de simulation qui n'étaient pas mentionnées dans l'article initial, la simulation pseudo-statique. J'ai expliqué comment les résultats des expériences seront présentés, en prenant comme exemples les résultats de l'article scientifique initial. J'ai également été chargé de la gestion de l'architecture du projet ainsi que des outils de contrôle de version tels que Git et des outils de construction tels que CMake.

Durant ce stage, j'ai eu l'opportunité d'interagir avec des chercheurs et des ingénieurs évoluant dans le domaine de la recherche. Ces échanges m'ont apporté de précieuses connaissances sur la réalité du travail de chercheur ainsi que sur les méthodes de travail scientifique. J'ai également pu observer de près la collaboration au sein d'une équipe au sein d'une entreprise.

Ce stage m'a également permis de faire l'expérience pour la première fois d'un projet à plein temps sur une durée prolongée. Grâce à cette expérience, j'ai pu appréhender les défis liés à l'organisation et au développement d'une application. J'ai également pris conscience de l'importance de gérer efficacement mon temps et d'estimer correctement la charge de travail nécessaire pour chaque partie du projet.

Le travail effectué avec WRENCH m'a également fait réaliser l'importance et la complexité de bien comprendre le fonctionnement des outils utilisés dans un projet. De nombreux problèmes rencontrés étaient liés à une connaissance incomplète de la bibliothèque WRENCH.

6 Annexe

6.1 Architecture du projet

L'architecture du projet dans le Git [5] est amenée à changer, donc en fonction de la date à laquelle vous consultez ce rapport, il est possible que certaines informations ne correspondent plus tout a fais du Git.

Pour simplifier la compilation du projet, j'ai utilisé CMake pour générer automatiquement des makefiles, ainsi que pour résoudre les problèmes de lien vers les fichiers et les bibliothèques.

L'architecture du projet se présente comme suit :

```
.
├── README.md
├── internship
├── src
├── tools
└── testFiles
    ├── testCompareFiles
    └── testOrdonancement
```

Présentation :

- `README.md` : une description du projet, de son organisation et de comment l'utiliser.
- `internship` : contient les documents qui sont liés à mon stage, tels que le fichier `.tex` qui permet de générer ce rapport ou `"suivi de travail.odp"` que j'ai utilisé pendant le stage pour discuter de mon avancement avec Yves Caniou.
- `src` : contient le code du simulateur.
- `tools` : contient les outils qui ne font pas partie du code, mais qui sont utilisés dans le cadre de ce projet, tels que `conversionScript.py` qui convertit des workflows d'un ancien format vers celui utilisé par ce projet.
- `testFiles` : contient les ressources nécessaires aux tests unitaires qui sont utilisées pour tester la validité de différents ordonnancements.

6.2 Intérêt personnel dans ce stage

L'une des raisons qui m'a fait choisir ce stage est mon intérêt potentiel pour la recherche comme choix de carrière. Grâce à ce stage, j'ai pu mieux comprendre le métier de chercheur en interagissant avec les chercheurs que j'ai côtoyés au LIP.

5. Vous pouvez retrouver le Git du projet ici : https://forge.univ-lyon1.fr/YVES.CANIOU/budget-stochastic-wf

6.2.1 Difficultés rencontrées

La difficulté majeure de ce projet était de comprendre le fonctionnement de WRENCH et comment l'appliquer à notre problématique. Il y a eu beaucoup de travail que j'ai dû faire et refaire car je n'avais pas compris comment un élément de WRENCH fonctionnait. Ces incompréhensions venaient en partie de la documentation de WRENCH, qui pourrait être plus claire sur certains points, mais elles venaient principalement de mon manque de familiarité avec certains concepts abordés. Mon manque d'expérience avec des bibliothèques plus complexes à utiliser a également joué un rôle dans mes problèmes d'appréhension de WRENCH.

Différences d'utilisation par rapport à l'usage prévu de WRENCH Une autre source de difficulté était la différence entre ce pour quoi WRENCH a été développé et les objectifs du stage. La plupart des classes outils (voir section 4.7) ont été développées pour accéder à des informations que WRENCH ne fournit pas normalement, car elles sont considérées comme inutiles dans ce contexte.

Manque de perspective sur WRENCH et le simulateur De plus j'ai eut des dificulter lier a ma conprehention partielle de WRENCH, ce sont ces difficulter qui m'ont pris le plus de temps a résoudre de manière globale car il existe relativement peut de resources pour mieux apréhender le fonctionement de cette librairie. Par example un des problèmes majeurs que j'ai rencontrer a la fin de mon stage etait lier au fait que je pensais qu'il etait possible de detereminer le nom d'un Computing Service (CS) lors de sa creation. Je pensais que c'etait possible car on peut renomer l'host d'un CS et typiquement le CS prendras le nom de sont host dans la majoriter des cas. Dans mes première simulations ce problème n'est jamais survenus. Mais dès lors qu'on crée des VM (et les CS associer) avec un temps de démarage non null, cette diférence de nom a beacoup plus de chance d'arivée. Or mon programe se repose largement sur le nom des CS pour determiner sur quel VM une tache seras executer. Et trouvée une alternative au nom des CS pour selectionner correctement la VM voulue s'est avéré compliquer, non pas parce que WRENCH ne le permet pas mais parce que la majoriter de mon programe se reposait sur des propriété des CS. Si j'avais été conscient de cette limitation dès le debut de mon stage, la mise en place de la simualtion pseudo static aurais été beacoup plus simple. Mais je manquais de perspective sur les outils fournis par WRENCH ainsi que sur ce qu iserais néssesaire au bon fonctionement du projet, ce qui m'as amener a faire de mauvais choix de designe du simulateur qui auront finit par me couter cher en temps et en effort.

Prise de contact avec les créateurs de WRENCH En cherchant à mieux comprendre le fonctionnement de WRENCH, j'ai notamment contacté les développeurs de WRENCH, qui m'ont indiqué un lien Slack où j'ai demandé des informations. J'ai notamment communiqué avec Henri Casanova, professeur à l'Université d'Hawaï, qui a répondu à la plupart de mes questions portant sur le fonctionnement interne de WRENCH.

Ma première question portai sur l'organisation des tâches prêtes données par le simulateur qui me semblait aléatoire. Les tâches prêtes étaient en réalité temporairement stockées dans un `std::set` qui ne se préoccupe pas de l'ordre de ses éléments. En fonction des ajouts et soustractions faites en interne par WRENCH, l'ordre des tâches prêtes pouvait donc varier.

Une de mes autres questions portait sur l'absence de fonctions permettant d'obtenir la vitesse de la connexion entre deux hôtes. Il s'avère que cette absence est intentionnelle et que les créateurs de WRENCH n'avaient pas mis en place cette fonctionnalité car ils pensaient qu'elle n'était pas typiquement accessible dans une situation réelle. Dans la réalité, il faut souvent obtenir cette information via des tests mis en place par l'utilisateur qui calcule le débit d'une connexion. Il est donc attendu des utilisateurs de WRENCH qu'ils fassent le

même genre de tests s'ils veulent obtenir cette information.

Ma dernière question portait sur la taille du fichier décrivant la plateforme, qui peut être importante car il doit notamment décrire individuellement chaque connexion entre chaque hôte s'il y en a une directe. C'est un problème sur lequel l'équipe de WRENCH est en train de travailler, notamment sur une alternative en C++ pour décrire la plateforme au lieu d'utiliser un fichier XML. La mise en place de la plateforme se ferait via des fonctions prédéfinies qui pourraient par exemple définir un hôte ou encore lier plusieurs hôtes via une connexion. Cette méthode aurait l'avantage d'utiliser des fichiers plus courts et de pouvoir automatiser beaucoup de choses qui devaient être faites explicitement avec l'ancienne méthode.

6.3 Timeline du stage

6.3.1 Bornes kilométriques atteints dans le projet

La durée de travail sur les différentes parties du simulateur, ainsi que leur état de complétion est détaillée dans le diagramme de Gantt (voir figure 5). Les durées indiquées sont proches de la réalité mais pas exactes, et bien sûr, les travaux présentés après le rendu de ce rapport sont des prédictions.

6.3.2 Travail qui aurait pu être fait avec plus de temps

Certains aspects du stage ont dû être ignorés en raison de la limitation de temps de ce projet. Par exemple, l'analyse des différentes méthodes de stockage et de transfert de données (voir section 3.5) aurait pu être intéressante. Il était également initialement prévu de mettre en place une simulation dynamique (voir section 4.4), mais il est rapidement devenu clair que ça ne serait pas réalisable dans les délais impartis.

S'il était prévu que je termine la mise en place de la simulation pseudo-statique lors de mon stage, je n'y suis malheureusement pas parvenu. J'ai partiellement mis en place le système permettant de gérer la création des VM de manière dynamique au cours de la simulation. Idéalement, j'aurais dû aussi implémenter de nouveaux tests unitaires pour vérifier les comportements des heuristiques pseudo-dynamiques. Enfin, il aurait fallu que je réutilise le code utilisé par le papier de recherche original afin de générer des graphes utilisant les résultats obtenus par notre simulation.

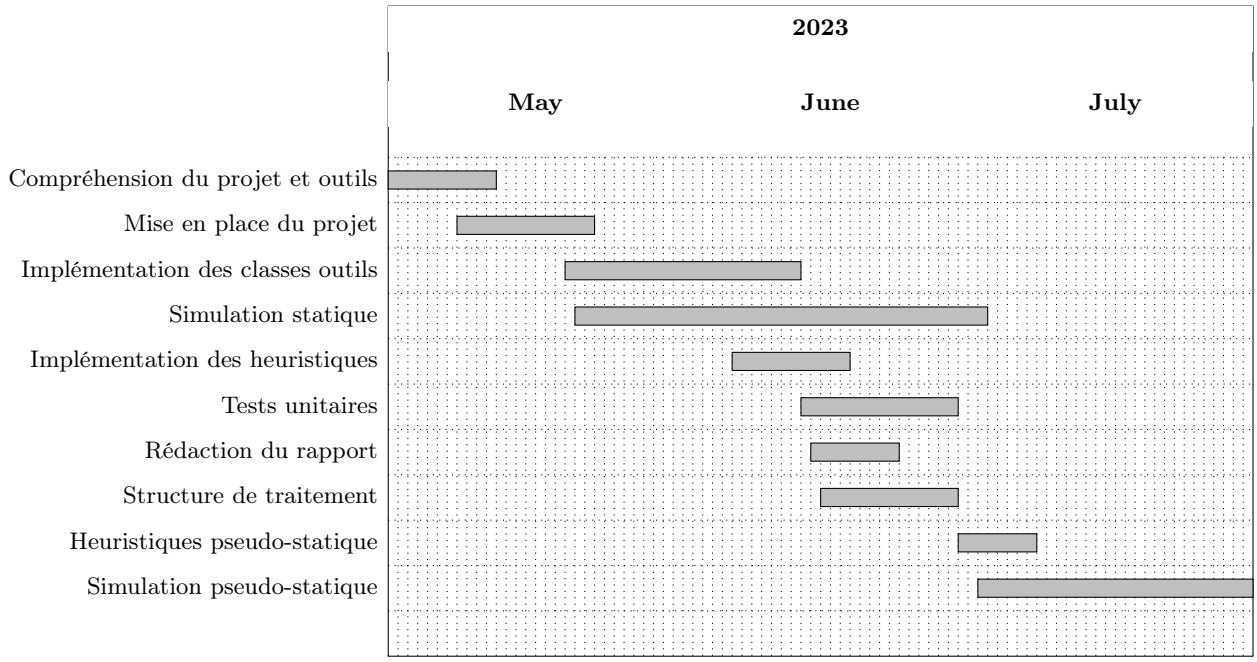

FIGURE 11 – Diagramme de Gantt pour le stage