Extension du prototypage rapide pour les pipelines d'imagerie radioastronomique : Simulation avancée multi-nœud HPC *

Ophélie RENAUD[1,3] Sunrise WANG[2] Nicolas GAC[1] François ORIEUX[4]

[1]Université Paris-Saclay, ENS Paris-Saclay, CNRS, SATIE, 91190, Gif-sur-Yvette, France.

[2]Université Côte d'Azur, Observatoire de la Côte d'Azur, CNRS Laboratoire J.-L. Lagrange, Nice, France

[3]ENS Rennes, CNRS, IRISA, France

[4]Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France

Résumé – SimSDP est un outil de prototypage rapide conçu pour fournir des estimations précoces des temps de calcul et de la consommation énergétique du futur pipeline de processeurs SDP de SKAO. Cet article étend une preuve de concept précédente en : (1) explorant davantage de parallélisme grâce à une modélisation en flux de données du parallélisme spectral, (2) simulant le comportement des pipelines sur des architectures multicœurs CPU multi-nœuds, (3) améliorant la modélisation des performances et en intégrant une automatisation du processus d'estimation des temps de calcul. Les résultats montrent que cet outil facilite l'orientation des décisions de conception en explorant trois algorithmes d'imagerie radio-interférométrique sur des systèmes HPC, avec une amélioration de la fiabilité des simulations, l'incertitude étant réduite à environ 10 %.

Abstract – SimSDP is a rapid prototyping tool designed to provide early estimates of the computation time and power consumption of SKAO's future SDP processor pipeline. This paper extends a previous proof-of-concept by: (1) exploring more parallelism through dataflow modeling of spectral parallelism, (2) simulating pipeline behavior on multi-core, multi-node CPU architectures, (3) improving performance modeling and incorporating automation of the computation time estimation process. The results show that this tool facilitates the orientation of design decisions by exploring three radio-interferometric imaging algorithms on HPC systems, with an improvement in the reliability of the simulations, the uncertainty being reduced to around 10 %.

1 Introduction

Les radiotélescopes, instruments essentiels à l'observation des phénomènes célestes, visent de plus en plus un traitement en temps réel, c'est-à-dire une capacité à analyser les données aussi rapidement qu'elles sont acquises. Des infrastructures comme le Square Kilometer Array Observatory (SKAO), grâce à leur grand nombre d'antennes réparties sur de vastes zones, génèrent des volumes massifs de données. Ces données doivent être traitées efficacement par des systèmes de calcul à haute performance (HPC), qui peuvent comporter des dizaines de milliers de nœuds, équipés de processeurs multicœurs entre autres. Le processus d'imagerie radio-interférométrique sera effectué par le Science Data Processor (SDP). Ces algorithmes d'imagerie, en constante évolution pour répondre aux besoins des chercheurs, partagent cependant des structures récurrentes, comme l'a démontré l'étude [9]. Ces algorithmes se caractérisent généralement par une série d'itérations qui produisent une image dite *dirty* du ciel, contenant des artefacts instrumentaux, suivis d'une série de boucles mineures qui visent à réduire les artefacts de l'image par un algorithme itératif de déconvolution.

L'optimisation des algorithmes avant leur déploiement effectif sur ces infrastructures complexes est essentielle pour garantir un fonctionnement efficace. Des outils tels que Dask [7], Daliuge [10] et PREESM [6], permettent le déploiement de telles applications en exploitant le modèle de flux de données, où l'application est représentée sous la forme d'un graphe acyclique dirigé (DAG), pour exposer et exploiter le parallélisme intrinsèque des algorithmes. Le graphe se compose de nœuds, appelés acteurs, représentant les calculs, interconnectés par des arêtes représentant les tampons en file *premier entré, premier sorti* (FIFO). Bien que ces outils permettent un déploiement rapide, ils ne sont pas conçus pour simuler précisément le comportement des architectures HPC complexes. Le Simulateur du SDP (SimSDP) se présente comme une solution prometteuse pour simuler l'allocation des ressources sur des systèmes HPC multi nœuds et multicœurs. Toutefois, la précision de ces simulations repose sur une estimation fiable des temps d'exécution des acteurs du graphe flux de données en fonction de l'architecture cible. À ce jour, des méthodes comme celle proposée par [9] offrent un cadre partiellement automatisé pour estimer ces temps d'exécution à partir de fonctions d'ajustement. Bien que cette approche constitue une avancée significative, elle est cependant limitée à des architectures CPU mono nœud.

Dans cet article, nous proposons une méthodologie pour faciliter l'exploration algorithmique en imagerie radio-interférométrique et l'exploration architecturale de systèmes HPC, avec trois contributions principales.

- **La modélisation du parallélisme spectral** sur les pipelines d'imagerie radio-interférométrique, afin de mieux exploiter les ressources des systèmes HPC.
- **La simulation avancée des architectures** en exploitant SimSDP pour étendre la simulation aux environnements multi nœuds multicœurs.
- **L'automatisation et optimisation des fonctions**

*Ce travail a été soutenu par DARK-ERA (ANR-20-CE46-0001-01).

d'ajustement des temps de calculs en améliorant l'échantillonnage et en explorant un plus grand espace de fonctions via une régression polynomiale, en minimisant l'erreur quadratique moyenne (RMSE) comme critère d'optimisation.

La section 2 présente le modèle générique de flux de donnée du pipeline d'imagerie radio-interférométrique. La section 3 décrit le fonctionnement du simulateur SimSDP et les améliorations apportées aux fonctions d'ajustement pour s'intégrer au système HPC. La section 4 présente les résultats expérimentaux obtenus avec SimSDP sur des systèmes HPC. Enfin, la section 5 conclut ce document.

2 Modèle générique flux de données d'imagerie radio-interférométrique

2.1 Imagerie radio-interférométrique

L'imagerie radio-interférométrique utilise des interféromètres pour mesurer les visibilités, qui sont des échantillons du ciel obtenus à partir de paires d'antennes. Ces visibilités $V(u,v)$ sont décrites par l'équation de mesure radio-interférométrique (RIME) [8] comme

$$V(u,v) = G_{uv} \int \left[D_{uv}(l,m) \frac{1}{\sqrt{1-l^2-m^2}} I(l,m) \right. \\ \left. e^{-2\pi i\left(ul+vm+w\left(\sqrt{1-l^2-m^2}-1\right)\right)} \right] dl\, dm \quad (1)$$

où G_{uv} représente les effets instrumentaux indépendants de la direction, D_{uv} représente les effets instrumentaux dépendants de la direction, (u,v) désignent les coordonnées spatiales continues dans le plan uv, (l,m) sont les coordonnées sur la sphère céleste, w est la coordonnée de la ligne de visée et $I(l,m)$ correspond à la véritable intensité du ciel.

Ainsi, pour transformer ces visibilités en une image du ciel, les algorithmes d'imagerie utilisent généralement deux étapes principales : une boucle majeure et une boucle mineure. L'étude [9] a proposé un modèle générique et modulaire basé sur un flux de données pour ces étapes, permettant de structurer les algorithmes en blocs élémentaires et de tirer parti du parallélisme des architectures modernes. La boucle majeure gère la transformation des visibilités en une image *dirty* du ciel via des opérations équivalentes au *gridding* et de *degridding*. Le *gridding* consiste à projeter les visibilités sur une grille régulière dans le plan uv afin de faciliter l'application de la transformée de Fourier rapide (FFT) pour générer une image *dirty*. Le *degridding* effectue l'opération adjointe projetant les coefficients de Fourier issues de la FFT sur les coordonnées continues des mesures dans le plan uv. Les principaux paramètres sont définis comme suit : le nombre de visibilités (n_v), la taille de la grille de l'image (n_g), le nombre de canaux de fréquence (n_f), la taille des noyaux de *gridding* et de *degridding* ($n_{\mathcal{D}}$ et $n_{\mathcal{G}}$), et le nombre de cycles mineurs (n_m). Le modèle modulaire [9] comprend trois versions algorithmiques pour la boucle majeure, adaptés à différents contextes. Ces algorithmes offrent un compromis entre précision et performance, les complexités suivantes sont valables si l'on considère un canal de fréquence n_f unique :

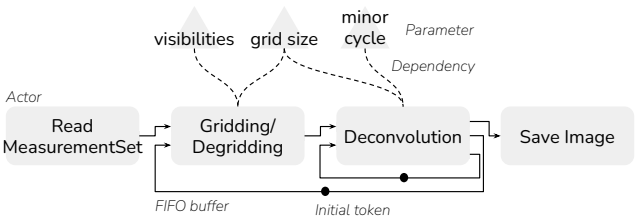

FIGURE 1 : Représentation simplifiée du modèle flux de donnée du pipeline générique d'imagerie radio-interférométrique

1. **transformée de Fourier directe (DFT)** : Calculs précis, mais coûteux en temps de calcul, avec une complexité de $O(2n_v n_g^2)$.
2. **FFT** avec convolution : Méthode rapide grâce à la grille, au prix d'une légère perte de précision, avec une complexité de $O(2n_g^2 \log_2 n_g^2 + n_v(n_{\mathcal{G}}^2 + n_{\mathcal{D}}^2)$.
3. **Grid-to-Grid (G2G)** [5] : Algorithme hybride visant à optimiser la précision tout en maintenant une bonne performance, avec une complexité de $O(2n_g^2 \log_2 n_g^2 + n'_v(n_{\mathcal{G}}^2 + n_{\mathcal{D}}^2))$. Cet algorithme est souvent plus rapide en pratique grâce à une simplification des visibilités (n'_v) et à une soustraction effectuée dans l'espace image plutôt que dans l'espace des visibilités.

La boucle mineure intervient pour nettoyer l'image *dirty* des artefacts introduits par l'instrumentation. Parmi les algorithmes disponibles, **CLEAN** par J.A. Hogbom [3] est le plus couramment utilisé. Cet algorithme identifie d'abord les sources dominantes dans l'image *dirty*, puis modélise ces sources tout en supprimant les artefacts associés à la réponse instrumentale (PSF). Enfin, l'image est mise à jour de manière itérative jusqu'à convergence vers une version idéalement dans artefacts. La complexité de CLEAN est de $O(n_g^2 + n_m)$, où une convolution de n_g^2 PSF est effectuée.

Dans cet article, nous cherchons à révéler les facteurs algorithmiques limitant le passage à l'échelle. Trois paramètres principaux se dégagent de la complexité arithmétique de ces algorithmes : le *nombre de visibilités* n_v, la *taille de la grille* n_g et le *nombre de cycles mineurs* n_m.

2.2 Sémantique du modèle PiSDF

Le modèle de flux de données facilite la comparaison algorithmique et se prête particulièrement bien au cas d'usage SKAO, en raison de ses exigences complexes en matière de traitement de données à grande échelle. Un modèle de flux de données populaire est le flux de données synchrone (SDF) [4], en raison de sa prédictibilité qui permet une analyse statique efficace. Dans ce modèle, les taux de consommation et de production des jetons sont définis par des entiers associés aux ports d'entrée et de sortie des acteurs. Le modèle flux de données synchrone parametré et interfacé (PiSDF)[2], illustré dans la Figure 1, étend le SDF en autorisant des taux définis par des expressions, offrant ainsi une plus grande flexibilité pour configurer n_v, n_g et n_m. Le modèle intègre également la hiérarchie permettant de spécifier le comportement interne des acteurs par un sous-graphe au lieu d'un code C. Le modèle PiSDF définit les interfaces d'un sous-graphe comme des ports de données d'entrée et de sortie de l'acteur hiérarchique parent. Ces interfaces permettent la transmission de jetons entre les

niveaux hiérarchiques et confèrent au modèle une forte modularité facilitant la comparaison des différentes versions des algorithmes de boucle majeure.

3 SimSDP : un outil pour la simulation multi nœuds

3.1 Fonctionnement de SimSDP

SimSDP, basé sur PREESM [6] et SimGrid [1], repose sur trois étapes principales :

- Partitionnement au niveau des nœuds : Le graphe de flux de données est divisé en sous-graphes, chaque sous-graphe étant affecté à un nœud spécifique du système HPC. Dans ce contexte, chaque sous-graphe représente un pipeline exécuté sur une bande de fréquence distincte, permettant ainsi une simulation indépendante des différentes fréquences.
- Partitionnement au niveau des threads : À l'intérieur de chaque nœud, les tâches sont réparties entre les cœurs de calcul disponibles via des techniques d'ordonnancement par liste.
- Simulation des charges de travail : Cette simulation prend en compte non seulement les communications inter nœuds via Message Passing Interface (MPI), mais aussi les synchronisations intra nœuds gérés par Pthread. Cela permet de modéliser les interactions entre les nœuds ainsi que l'exécution parallèle sur les cœurs du même nœud.

SimSDP génère automatiquement le code MPI pour assurer la communication entre les nœuds et utilise Pthread pour la gestion des synchronisations au niveau des cœurs.

3.2 Parallélisme spectral

Dans cet article, nous considérons le modèle de données sous forme de cube spectral, où chaque dimension correspond respectivement à la fréquence, la position spatiale en ascension droite et la position spatiale en déclinaison. Cette organisation, très courante dans le traitement des données radio-interférométrique, permet de découper les données en bandes de fréquence indépendantes, facilitant ainsi leur traitement en parallèle. Dans notre approche, chaque nœud du système de calcul est responsable de l'exécution d'un pipeline complet (boucle majeure et boucle mineure) pour une bande de fréquence donnée. Cette méthode offre plusieurs avantages : (1) indépendance entre nœuds, les bandes spectrales étant traitées séparément, ce qui réduit les surcharges de synchronisation entre les nœuds ; (2) facilité de parallélisation, grâce à la structure du cube spectral qui se prête naturellement à une répartition de charge équilibrée sur les ressources disponibles ; et (3) optimisation mémoire, chaque nœud manipulant uniquement une fraction des données globales, ce qui réduit les contraintes mémoire locales. La modélisation en flux de données consiste à associer un sous-graphe à chaque bande de fréquence, chaque sous-graphe contenant une réplique complète du pipeline de traitement. Afin de permettre l'interprétation du graphe par les outils basés sur les flux de données, la méthode consiste

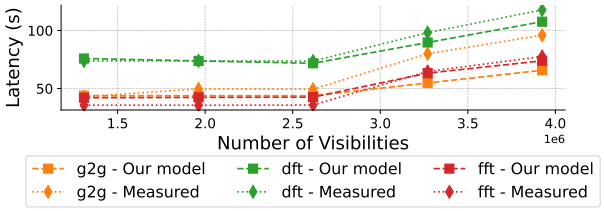

FIGURE 2 : Latences mesurées et simulées en fonction du nombre de visibilités par nœuds, avec n_c = 150 et n_g = 1536 sur 21 nœuds.

à créer une dépendance factice entre les sous-graphes, dans laquelle un pipeline logiciel est intégré.

3.3 Extension du modèle d'estimation des temps de calculs

Pour faciliter la comparaison des pipelines, l'étude [9] que nous cherchons à étendre a mis en place une méthode manuelle pour estimer les temps d'exécution des acteurs. Cette méthode associe à chaque acteur un modèle de timing sous forme de polynôme, dont les paramètres incluent n_v, n_g et n_m. Le processus manuel initial consiste à exécuter un code instrumenté, sauvegarder les temps d'exécution des acteurs et faire varier les paramètres. Une fois les échantillons collectés, le processus calcule des polynômes de degré maximal à 2 (par défaut). Cette approche reste toutefois limitée par le nombre d'échantillons et le degré des polynômes évalués. Cela se reflète dans la fiabilité des simulations des pipelines complets, en particulier dans les configurations extrêmes. Afin de dépasser ces limitations et d'explorer plus efficacement l'espace des paramètres, le processus a été entièrement automatisé. La première étape encapsule les codes instrumentés et paramétrés de chaque acteur et sauvegarde les temps d'exécution pour chaque configuration. La seconde étape ajuste ces données par régression polynomiale sur une ou deux dimensions, en fonction du nombre de paramètres affectant chaque calcul. La méthode explore les degrés de polynôme possibles et sélectionne celui qui minimise la RMSE entre la mesure de référence et le modèle. À noter que l'augmentation du nombre de points permet également d'accroître le degré maximal du polynôme pouvant être évalué. Pour éviter les valeurs aberrantes aux bornes des fonctions d'ajustement, les équations sont bornées avec les valeurs minimales des mesures de références.

4 Résultat expérimentaux

Le simulateur SimSDP a été utilisé pour simuler le déploiement et générer le code du pipeline générique et modulaire d'imagerie radio-interférométrique [9] étendu sur 1 et 21 canaux, conformément au jeu de données de test GLEAM, couvrant une plage de fréquences de [72 ; 231] MHz avec une résolution native des canaux de 7,68 MHz. Nous évaluons plusieurs versions du pipeline : une version contenant uniquement des cycles majeurs, utilisant les algorithmes G2G, DFT, FFT, sans cycle mineur, et une version combinant G2G pour la description des cycles majeurs et CLEAN pour les cycles mineurs. Les simulations ont été réalisées sur des systèmes de

Pipeline	Err.(%) Pire cas	Err.(%) S. Wang et al.	Err.(%) Our model
DFT	32	23	8
FFT	43	21	12
G2G	68	35	22
G2G - CLEAN	56	31	17

TABLE 1 : Erreur relative des simulations par rapport aux mesures expérimentales sur un nœud.

1 à 21 nœuds, équipés de 6 cœurs CPU chacun. Les configurations de paramètres simulés et mesurés sont les suivantes : n_v = {1308160 ; 1962240 ; 2616320 ; 3270400 ; 3924480}, n_g = {512 ; 1024 ; 1536 ; 2048 ; 2560} et n_m = {50 ; 100 ; 150 ; 200 ; 250}, représentant ainsi 125 configurations au total. Les codes générés par l'outil sont paramétriques et déployés sur Grid5000 afin d'évaluer les performances dans des environnements réels [†]. Les implémentations des pipelines, les scripts et les codes générés sont disponibles en ligne [‡].

4.1 Analyse algorithmique

La Figure 2 compare les latences des simulations obtenues avec nos fonctions d'ajustement aux mesures expérimentales des codes des pipelines déployés sur 21 nœuds en fixant n_m = 150 et n_g = 1536, tout en faisant varier n_v sur chaque nœud. Les expérimentations montrent que le pipeline DFT s'adapte beaucoup moins bien que les autres en ce qui concerne le *nombre de visibilités*. Toutefois, la distribution des données sur 21 nœuds permet de mieux répartir la charge de travail et de repousser ces limites à des valeurs plus élevées. Ainsi, il devient envisageable de traiter des jeux de données de plus grande taille, jusqu'à atteindre l'échelle de SKAO. De plus, nos résultats montrent une latence globalement plus faible et mieux maîtrisée pour le pipeline FFT, bien qu'il reste sensible à la configuration des paramètres. En revanche, le pipeline G2G présente une forte variabilité liée à un paramètre supplémentaire : la densité des visibilités, non pris en compte dans les fonctions d'ajustement.

4.2 Analyse de fiabilité des simulations

Le tableau 1 résume l'erreur relative entre les latences simulées et mesurées, calculée à l'aide de la RMSE, obtenue avec : les modèles de pire cas des valeurs de paramètres, les modèles polynomiaux quasi linéaires initiaux [9], et nos propres modèles polynomiaux. Cette comparaison a été réalisée pour les pipelines déployés sur un nœud. Exécuter les 125 configurations prend environ 4 heures par pipeline, tandis que la simulation ne prend qu'environ 20 minutes, soulignant ainsi l'intérêt d'un modèle prédictif fiable pour limiter les coûts expérimentaux. Une analyse purement statique, première colonne, basée sur un temps fixe des calculs avec les paramètres à leur valeur maximale évaluée, ne donne pas de bons résultats, car la complexité des pipelines dépend directement des paramètres d'exécution. Les modèles quasi linéaires initiaux ne reflètent que grossièrement le comportement réel des algorithmes, car ils n'intègrent pas cette variabilité dynamique. En revanche, nos modèles polynomiaux permettent une analyse plus fine et une meilleure prédiction des temps d'exécution, avec une réduction de l'erreur d'au moins 9%. Cependant, nos modèles rencontrent des difficultés à modéliser précisément le comportement de G2G. En effet, cet algorithme a une nature dynamique, avec des valeurs de paramètres qui ne sont pas toujours connues en amont. Par conséquent, même si notre analyse statique est plus précise et mieux adaptée que les approches précédentes, elle reste limitée face à la complexité dynamique de G2G.

[†] https://www.grid5000.fr/w/Rennes:Hardware

[‡] https://github.com/Ophelie-Renaud/simsdp-generic-imaging-pipeline

5 Conclusion et perspectives

Cet article présente une approche combinant une analyse statique et des modèles polynomiaux multidimensionnels pour simuler le comportement des calculs des pipelines d'imagerie radio-interférométrique sur des systèmes HPC avec SimSDP. Cette modélisation couvre un large éventail de configurations et anticipe le comportement du pipeline lors du passage à l'échelle, établissant un lien entre analyse statique et évaluation dynamique des performances. Nous avons montré que SimSDP permet d'explorer le parallélisme spectral, de simuler le comportement des pipelines sur des architectures multicœurs CPU multi nœuds, et d'améliorer la précision des estimations avec une incertitude réduite à environ 10 %. Les perspectives incluent l'extension à des architectures mono-GPU et multi-GPU, ainsi que l'exploration de jeux de données plus variés.

Références

[1] H. CASANOVA et AL : Versatile, scalable, and accurate simulation of distributed applications and platforms. *JPDC*, 2014.

[2] K. DESNOS et AL. : Pimm : Parameterized and interfaced dataflow meta-model for mpsocs runtime reconfiguration. *SAMOS*, 2013.

[3] J. A. HOGBOM : Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines. *Astron. Astro.*, 1974.

[4] E.A. LEE et D.G. MESSERSCHMITT : Static scheduling of synchronous data flow programs for digital signal processing. *TC*, 1987.

[5] N. MONNIER et AL : Fast grid to grid interpolation for radio interferometric imaging. *Astron. Comput.*, 2023.

[6] M. PELCAT et AL. : Preesm : A dataflow-based rapid prototyping framework for simplifying multicore dsp programming. *EDERC*, 2014.

[7] Matthew ROCKLIN : Dask : Parallel computation with blocked algorithms and task scheduling, 2015.

[8] O. SMIRNOV : Revisiting the radio interferometer measurement equation. i. a full-sky jones formalism. *Astron. Astro.*, 2011.

[9] Sunrise WANG et AL : An initial framework for prototyping radio-interferometric imaging pipelines. *DASIP*, 2024.

[10] Chen WU et AL : Daliuge : A graph execution framework for harnessing the astronomical data deluge. *Astron. Comput.*, 2017.