La reproductibilité des calculs coûteux

Konrad Hinsen [1]

La reproductibilité computationnelle est la possibilité de refaire un calcul en partant de ses ingrédients : code source et données d'entrée. Dans le contexte du calcul scientifique, l'intérêt de la reproductibilité est de pouvoir vérifier que des résultats d'une simulation ou d'une analyse de données, tels que publiés dans un article, proviennent réellement du calcul décrit dans ce même article, et que cette description est complète. La certitude sur la provenance des résultats permet par la suite d'explorer le code source et les données d'entrée afin de mieux comprendre ce qui a été fait.

La façon la plus évidente de vérifier la reproductibilité d'un calcul est de tenter de le refaire, et comparer à la fin les résultats avec ceux publiés par ses auteurs. Mais dans le cas d'un calcul coûteux, cette méthode n'est pas toujours applicable, parce que les moyens de calculs nécessaires sont soit indisponibles soit trop onéreux. Dans cet article, je vais décrire comment on peut se convaincre de la reproductibilité sans refaire le calcul, et aussi discuter ce que vaut une telle approche par rapport à la tentative explicite de reproduction.

C'est quoi, un calcul coûteux ?

Avant d'aller dans le vif du sujet, je vais rendre le terme abstrait du calcul coûteux plus concret. Évidement, l'étiquette « coûteux » est subjective. Quelles sont donc les situations réelles dans lesquelles un chercheur ne peut pas simplement refaire un calcul ?

1. CR CNRS au Centre de biophysique moléculaire, Orléans.

DOI:10.48556/SIF.1024.18.11 **1024** – Bulletin de la SiF, numéro 18, Novembre 2021, pp. 11–14

Le cas le plus extrême est le calcul dit « haute performance », effectué en utilisant les plus grands centres de calcul actuellement disponibles : par exemple, la simulation d'une grande protéine sur le calculateur Jean Zay [2], installé à l'IDRIS à Orsay. Une telle simulation peut occuper des centaines de processeurs pendant quelques semaines. Et comme les logiciels sont souvent spécifiquement optimisés pour de tels ordinateurs, on ne peut pas refaire les calculs à l'identique sur une machine ordinaire, même si l'on était prêt à attendre bien plus longtemps. Reste l'option de demander un accès à ce même ordinateur (tant qu'il existe), auprès du GENCI [3] qui gère les gros moyens informatique de la recherche française. Je suis sûr que les comités du GENCI n'accepteraient pas une telle demande, qui ne représente pas un bon usage de moyens de calcul rares et chers.

Dans le travail quotidien d'un chercheur, même un calcul beaucoup plus modeste peut être considéré comme coûteux. Dans le cadre d'une collaboration avec un collègue, si je souhaite régulièrement vérifier la reproductibilité de son travail afin de l'adapter par la suite à une question un peu différente, même quelques heures de calculs peuvent être un frein à la progression fluide du projet commun. En fin de compte, un calcul coûteux est un calcul qu'on ne refait pas à cause de l'effort que cela représente, même si on sait qu'on devrait le faire.

Une question de confiance

La clé pour voir comment on peut s'assurer de la reproductibilité d'un calcul sans le refaire est de reformuler l'objectif. En répétant le calcul, j'obtiens la réponse *oui* ou *non* à la question : « Moi, aujourd'hui, puis-je reproduire les résultats de ce calcul ? » Ce n'est pas vraiment la question la plus intéressante du point de vue de la science, qui est un processus collectif de longue durée. Une meilleure question serait « Sous quelles conditions, et pendant combien de temps, un chercheur compétent dans le domaine peut-il reproduire les résultats de ce calcul ? » Cette question est bien moins précise, et il est impossible d'y répondre avec une certitude absolue.

Je propose donc de remplacer cette question par une autre, plus pragmatique : « Quel événement augmenterait la confiance que l'on peut avoir dans la reproductibilité d'un calcul ? » Une répétition réussie figure sûrement sur la liste des bonnes réponses, mais il y en a d'autres, mieux adaptées aux calculs coûteux. Dans la suite de cet article, je vais me concentrer sur une réponse sur laquelle je travaille moi-même depuis une dizaine d'année : l'utilisation d'une chaîne d'outils qui garantit la reproductibilité des calculs dont on lui confie la gestion. J'appelle cette approche la *reproductibilité par construction*.

Bien évidemment, une telle chaîne d'outils doit aussi permettre une répétition explicite de tout calcul aux fins de vérification. Le but est de supprimer la *nécessité* d'une répétition, mais pas sa *possibilité*. Au contraire, la confiance dans la chaîne

2. `http://www.idris.fr/jean-zay/`.
3. `https://www.genci.fr/`.

d'outils est fondée sur la réussite sans faille de reproductions explicites pour les calculs peu coûteux.

Pourquoi la reproductibilité est si difficile ?

Pour construire une chaîne d'outils qui assure la reproductibilité, il faut d'abord comprendre quels sont les problèmes à résoudre. Le calcul étant déterministe, à l'exception de certains calculs parallèles, on s'attend à ce que la reproductibilité du calcul soit la règle plutôt que l'exception. Et si on relance le même logiciel avec les mêmes données d'entrée plusieurs fois sur le même ordinateur, on obtient en effet les mêmes résultats avec une haute fiabilité.

On peut en conclure que quand on n'arrive pas à reproduire un résultat de calcul scientifique, quelque chose a changé : le logiciel, les données d'entrée, ou l'ordinateur. Dans la vaste majorité de cas, c'est bien le logiciel qui est en cause. En fait, le « logiciel » est en réalité une pile complexe de composants, avec un système d'exploitation comme base sur laquelle on empile des couches d'infrastructure de plus en plus spécialisées, et seulement tout en haut ce que l'utilisateur considère être *son* logiciel. Au total, on a typiquement quelques centaines de composants. Il devient alors difficile d'en fournir une liste complète qui contient en plus la version précise de chaque composant. Et même avec une telle liste en mains, il n'est pas évident de reconstruire un assemblage identique plus tard et sur un autre ordinateur.

Une solution fréquemment proposée est le recours aux machines virtuelles ou aux conteneurs du genre Docker ou Singularity. Ces technologies permettent en effet de préserver une version exécutable de la totalité de la pile. Mais ils ne préservent pas le lien entre les exécutables et leur code source, qui est pourtant essentiel pour tirer un bénéfice scientifique de la reproductibilité : pouvoir inspecter le code pour le comprendre et pour l'adapter à d'autres situations.

Une autre complication s'ajoute à la complexité de la pile logicielle : tout calcul scientifique se déroule en plusieurs phases, dont chacune produit le *code* qui sera exécuté dans la phase suivante [1]. On oublie facilement les premières phases, qui mettent en oeuvre des compilateurs et d'autres outils de construction de logiciels, avant qu'on puisse lancer la dernière phase qui produit les résultats scientifiques. Mais les détails de la compilation ont souvent un impact sur ces résultats ; par exemple, via les options de compilation qui définissent les règles précises de l'arithmétique à virgule flottante. Pour rendre le calcul reproductible, il faut inclure toutes ces phases dans la reproductibilité par construction.

À la recherche de la reproductibilité par construction

Une chaîne d'outils pour assurer la reproductibilité du calcul doit donc se baser sur une fondation stable dans le temps, et permettre la construction d'une pile logicielle en plusieurs phases avec un traçage complet de chaque étape et de tout code source utilisé. Elle doit aussi être compatible avec les logiciels scientifiques en usage

aujourd'hui si elle veut avoir une utilité dans la recherche. Malheureusement, il n'est pas facile de satisfaire toutes ces exigences.

Ma première tentative [2] utilisait la *Java Virtual Machine* (JVM) comme fondation et le format HDF5 [4] pour la distribution, tant des logiciels que des données. Comme preuve de faisabilité, elle était plutôt réussie, mais peu de logiciels scientifiques sont écrits pour la JVM, et ces rares logiciels nécessitent une adaptation pour la distribution en format HDF5. Ma deuxième tentative [2] utilisait le langage Python comme base, pour pouvoir profiter de son vaste ensemble de bibliothèques scientifiques. Mais cet ensemble manque de stabilité, et les premières publications utilisant ce format en 2013 ne sont déjà plus reproductibles pour cette raison.

L'approche que j'explore actuellement est basée sur le gestionnaire de logiciels Guix [5]. Guix assure la reproductibilité d'une pile logicielle complète, mais ne prend pas en charge la dernière phase d'un calcul, celle qui produit les résultats d'intérêt scientifique. Ceci est l'objectif d'une extension en cours de développement, le *Guix Workflow Language* [6] (GWL). En théorie, cette approche devrait satisfaire toutes les exigences. Mais eu égard à la diversité des calculs et des modes d'interaction avec l'ordinateur, un seul outil pour la dernière phase ne pourra pas satisfaire les besoins de tout le monde. Il reste du travail à faire !

Références

[1] Konrad Hinsen, *Staged Computation : The Technique You Did Not Know You Were Using*, 2020, http://doi.org/10.1109/MCSE.2020.2985508.

[2] Konrad Hinsen, *ActivePapers : a platform for publishing and archiving computer-aided research*, 2015, http://doi.org/10.12688/f1000research.5773.3.

4. https://www.hdfgroup.org/solutions/hdf5/.
5. https://guix.gnu.org/.
6. https://www.guixwl.org/.