Mode d'emploi rapide du test d'extensibilité (sur Antares)

Avant de pouvoir obtenir les temps d'exécution selon les configurations envisagées, plusieurs étapes sont à réaliser :

- compiler le code à l'aide de la commande "make",
- une fois les fichiers de sortie générés, utiliser la commande "./launch_runs.sh -g" ou "sh launch_runs.sh -g", ce qui permet de créer les dossiers correspondants à chaque cas,
- sans changer de dossier, taper la commande "./launch_runs.sh -s" ou "sh launch_runs.sh -s", ce qui permet de soumettre les jobs (pour accepter de soumettre un job, taper "y").

Une fois le calcul effectué, il faut aller dans le sous-dossier correspondant et visualiser le fichier "log" qui contient les informations sur le calcul. Le temps d'exécution y est inscrit.

1 Taille optimale des blocs

Fichier launch_runs_BO.sh

Le test pour déterminer la taille optimale des blocs s'apparente à celui réalisé lors du Strong Scaling (la taille totale du maillage reste fixe).

La taille optimale des blocs se détermine à l'aide de l'efficacité réduite :

$$E = t.\frac{n}{N_{noeuds}N_{ite}} \tag{1}$$

Où t est le temps d'exécution pour N_{ite} itérations (défini dans le fichier test3D.template) et un nombre total de noeuds N_{noeuds}.
Elle est tracée en fonction du nombre total de points par cœur. Cette courbe présente normalement un plateau, correspondant à la taille optimale des blocs. Si ce n'est pas le cas, le test est à refaire avec des tailles de blocs plus petites. Une fois la taille de bloc optimal trouvée, elle sera utilisée pour les tests suivants.

2 Test d'extensibilité

Le test d'extensibilité doit se faire dans des conditions proches de celles pour un calcul réel (par exemple, condition périodique, chimie...). Il faut définir un nombre de cœurs maximal n_{max}, correspondant à la plus grosse configuration envisagée. A partir de celui-ci, il est possible de définir un nombre de cœurs n_{ref} pour l'exécution de référence, tel que :

$$n_{ref} \leq \frac{n_{max}}{4} \tag{2}$$

Il est de plus conseillé d'utiliser une à trois valeurs intermédiaires.

Ceci nous permet ensuite de tracer l'accélération et l'efficacité parallèle en fonction du nombre de cœurs, et ainsi de visualiser le comportement du code. On note T_{ref} le temps d'exécution sur n_{ref} cœurs, et T_n le temps d'exécution du code sur n cœurs, sachant que $n > n_{ref}$.

2.1 Strong Scaling

Fichier launch_runs_SS.sh

Accélération :

$$acc(n) = \frac{T_{ref}}{T_n} \tag{3}$$

Efficacité parallèle :

$$eff(n) = \frac{acc(n)}{\frac{n}{n_{ref}}} \tag{4}$$

Le Strong Scaling permet d'évaluer le comportement du code lorsque la taille totale du maillage est fixée, et que l'on fait varier le nombre de cœurs sur lequel il calcule. La charge par cœur diminue donc lorsque leur nombre augmente.
Dans notre cas, la taille totale du maillage a arbitrairement été fixée à 200x200x200, mais il peut être nécessaire de la modifier.

2.2 Weak Scaling

Fichier launch_runs_WS.sh

Accélération :

$$acc(n) = \frac{n}{n_{ref}} \frac{T_{ref}}{T_n} \tag{5}$$

Efficacité parallèle :

$$eff(n) = \frac{T_{ref}}{T_n} \tag{6}$$

Le Weak Scaling permet d'évaluer le comportement du code lorsque la taille de bloc est fixée, et que l'on fait varier le nombre de cœurs sur lequel il calcule. La charge par cœur reste donc constante, et c'est la taille totale du maillage qui augmente lorsque le nombre de cœurs augmente.

Dans le fichier présenté, la taille de blocs optimale est supposée être entre 20x20x20 et 40x40x40. Afin d'être plus précis, la méthode du Weak Scaling est utilisée : en comparant les évolutions de l'accélération et de l'efficacité selon le nombre de cœurs, la configuration présentant le moins d'écart avec la théorie sera privilégiée.
Le nombre optimal de cœurs est ensuite déterminé pour la configuration en utilisant l'évolution de l'accélération : lorsque la courbe réelle ne suit plus la courbe théorique, le nombre optimal a été dépassé.