Comprendre les informations de performances de LAMMPS

Antoine Rincent, 2019

Lien vers la documentation à ce sujet

Exemple d'output de performances

```
Loop time of 10294.3 on 12 procs for 10000 steps with 4000000 atoms


Performance: 0.084 ns/day, 285.953 hours/ns, 0.971 timesteps/s

1028.8% CPU use with 1 MPI tasks x 12 OpenMP threads


MPI task timing breakdown:

Section |  min time  |  avg time  |  max time  |%varavg| %total

---------------------------------------------------------------

Pair    | 8239.1     | 8239.1     | 8239.1     |   0.0 | 80.04

Neigh   | 0          | 0          | 0          |   0.0 |  0.00

Comm    | 123.48     | 123.48     | 123.48     |   0.0 |  1.20

Output  | 1.4146     | 1.4146     | 1.4146     |   0.0 |  0.01

Modify  | 1798.8     | 1798.8     | 1798.8     |   0.0 | 17.47

Other   |            | 131.4      |            |       |  1.28


Nlocal:    4e+06 ave 4e+06 max 4e+06 min

Histogram: 1 0 0 0 0 0 0 0 0 0

Nghost:    564665 ave 564665 max 564665 min

Histogram: 1 0 0 0 0 0 0 0 0 0

Neighs:    2.8e+08 ave 2.8e+08 max 2.8e+08 min

Histogram: 1 0 0 0 0 0 0 0 0 0


Total # of neighbors = 280000000

Ave neighs/atom = 70

Neighbor list builds = 0

Dangerous builds = 0
```

Breakdown de ce que chaque section de l'output désigne

```
Loop time of 10294.3 on 12 procs for 10000 steps with 4000000 atoms
```

Donne le nombre d'itérations et d'atomes sur le nombre de processeurs (procs, 12 dans ce cas), loop time donne le temps total de la simulation du clock de LAMMPS.

```
Performance: 0.084 ns/day, 285.953 hours/ns, 0.971 timesteps/s

1028.8% CPU use with 1 MPI tasks x 12 OpenMP threads
```

La ligne Performance permet d'estimer le temps nécessaire pour le déroulement d'une simulation désirée, le % devrait être 100% * n OpenMP threads ou inférieur, dans ce cas-ci, la valeur maximale serait 1200%, la différence entre cette valeur théorique et celle fournie est en raison des latences de parallélisation, par exemple avec le passage de l'information par la mémoire.

```
MPI task timing breakdown:

Section |  min time  |  avg time  |  max time  |%varavg| %total

---------------------------------------------------------------

Pair    | 8239.1     | 8239.1     | 8239.1     |   0.0 | 80.04

Neigh   | 0          | 0          | 0          |   0.0 |  0.00

Comm    | 123.48     | 123.48     | 123.48     |   0.0 |  1.20

Output  | 1.4146     | 1.4146     | 1.4146     |   0.0 |  0.01

Modify  | 1798.8     | 1798.8     | 1798.8     |   0.0 | 17.47

Other   |            | 131.4      |            |       |  1.28
```

Donne l'information sur le temps passé sur chaque type de calculs, toutes les sections possibles ne sont pas présentes dans l'exemple plus haut, l'intérêt majeur réside dans la dernière colonne, qui donne le % du temps de la simulation passé sur chaque type de tâche.

- Pair : calculs de force entre atomes sans liens
- Bond (pas affiché) : comme Pair, mais avec liens interatomiques
- Kspace (pas affiché) : les intéractions à longue distance
- Neigh : construction des listes de voisins
- Comm : communication entre les processeurs sur les atomes et leurs propriétés
- Output : impression des données thermodynamique et des fichiers dump
- Modify : les calculs imposés par des appels des fonctions Fix et Compute
- Other : toute autres tâches

Les colonnes min time, max time et avg time donnent respectivement le temps minimal, maximal et moyen passé sur chaque section de tous les processeurs. Par exemple, si pour une tâche donné le processeur 2 passe plus de temps que le processeur 4, alors min time donnera le temps écoulé dans le processeur 2, et max time donnera le temps écoulé dans le processeur 4. Ceci est représentatif d'un déséquilibre de la répartition des tâches

La colonne %varavg est utile pour déterminer un tel déséquilibre, un pourcentage près de 0 (comme dans l'exemple) représente une répartition parfaite des tâches entre chaque processeur, alors qu'un pourcentage plus élevé désigne un déséquilibre.

```
Nlocal:    4e+06 ave 4e+06 max 4e+06 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost:    564665 ave 564665 max 564665 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs:    2.8e+08 ave 2.8e+08 max 2.8e+08 min
Histogram: 1 0 0 0 0 0 0 0 0 0
```

Nlocal représente le nombre d'atomes présents dans la simulation, Nghost représente le nombre d'atomes « phantomes » de LAMMPS et Neighs représente le nombre de voisins, le tout par processeur. Les valeurs min et max permettent de déterminer la répartition des atomes totaux par processeur, on en voit la distribution sur l'histogramme de 10 chiffres binaires. La somme de ces chiffres est égale au nombre de processeurs. Ces résultats désignent que le total des 4 000 000 d'atomes de la simulation étaient sur un seul processeur, ce qui est logique puisque celle-ci a été lancée sur mon ordinateur personnel sur 12 threads (12 threads sur les 16 totaux d'1 seul processeur).

```
Total # of neighbors = 280000000
Ave neighs/atom = 70
Neighbor list builds = 0
Dangerous builds = 0
```

Cette dernière section donne des informations agrégées de la simulation. La valeur de Neighbor list builds représente le nombre de fois que la liste de voisins a du être rebuilt. Si la valeur de Dangerous builds est différente de 0 il faudrait considérer réduire la valeur dans la fonction neigh_modify, puisque cela veut dire que certains déplacements d'atomes n'ont pas été adéquatement pris en compte.

Utilisation de la mémoire

LAMMPS va imprimer dans le fichier .log la mémoire à utiliser pour le test en cours, par exemple :

```
run ${iterations}
run 10000
Last active /omp style is pair_style eam/alloy/omp
Per MPI rank memory allocation (min/avg/max) = 3511 | 3511 | 3511 Mbytes
```

Cette information (Per MPI rank memory allocation) donne un bon estimé de la quantité de mémoire à fournir lors du lancement du fichier d'input, pour chaque processeur (MEM-PER-CPU). Il faut savoir que ce chiffre peut grandement varier selon la simulation et le type de pair style utilisé (EAM vs LJ par exemple).