GLCS : TD1

Adrien Henrot
adrien.henrot@ens.uvsq.fr
M2 CHPS

December 8, 2024

Abstract

Le but de ce tp était de benchmarker un code en utilisant docker et sa fonctionnalité swarm pour simuler une grille de calcul. Le code était fournit en deux versions; une en Python, l'autre en C.

1 Introduction

Dans ce TP nous avons utilisé Docker pour simuler une grille de calcul. Le déroulement était le suivant: créer un swarm Docker pour permettre a différents containers de communiquer entre eux via SSH et préparer les clefs SSH requises. Une fois cela opérationnel, il suffit de rentrer dans un des containers pour l'utiliser comme frontale d'un cluster et lancer des programmes. Le code a benchmarker étant dépendant de MPI nous utilisons sa fonctionnalité ORTE de manière a faire en sorte que les processus MPI puissent envoyer leurs messages a travers le réseau. Pour se faire, on récupère l'adresse IP des différents containers que l'on passera ensuite a MPI. Enfin nous avions a lancer deux codes python et un code C faisant des test de bande passante bi-directionelle, c'est a dire de communication bi-directionelle entre deux processus MPI, puis récupérer leurs performances pour différentes tailles de buffers. La référence du benchmark : https://mvapich.cse.ohio-state.edu/benchmarks/

2 Résultats

Après avoir lancé les différentes versions du code en utilisant la commande suivante :

```
mpirun --mca orte_base_help_aggregate 0
       --mca btl_tcp_if_include [network mask]
       -n 2 -host [containers ip addresses] <program>
```

Nous obtenons les résultats suivants (rapide analyse):

Les résultats observés peuvent etre étonnants, notamment la différence entre les deux versions python. On passe du simple au triple en terme de bande passante sur de grandes tailles de buffer entre la version fournit sur github (Python v.2) contre la version du benchmark original (Python v.1), la version du benchmark original a en effet une bande passante bien plus haute. En revanche, étonnament la version C a une bande passante étant par endroit plus faible que son équivalent python (v.1) a partir de buffers de 1024 bytes. La seconde version python est bien moins performante que les deux autres codes.

Pour ses benchmarks, l'ordinateur était branché sur le courant, la fréquence du processeur stable et le minimum possible d'application activé en arrière plan.

Size (bytes)	Python v.1 (Mb/s)	Python v.2 (Mb/s)	C (Mb/s)
1	0.02	0.02	0.03
2	0.03	0.02	0.08
4	0.08	0.03	0.09
8	0.12	0.05	0.35
16	0.48	0.18	0.57
32	0.56	0.44	2.11
64	0.92	0.76	2.44
128	3.16	1.52	2.20
256	4.77	2.80	6.50
512	10.56	6.89	13.04
1024	16.82	9.09	13.77
2048	52.38	17.71	58.35
4096	93.93	47.89	83.83
8192	249.22	89.06	157.00
16384	330.16	155.24	395.94
32768	592.22	187.37	471.69
65536	337.91	171.45	379.29
131072	351.03	178.33	451.65
262144	430.14	189.40	514.64
524288	462.83	247.63	502.23
1048576	504.34	233.06	553.53
2097152	422.68	166.80	370.21
4194304	946.10	249.56	372.61

Table 1: MPI Bi-Directional Bandwidth benchmark

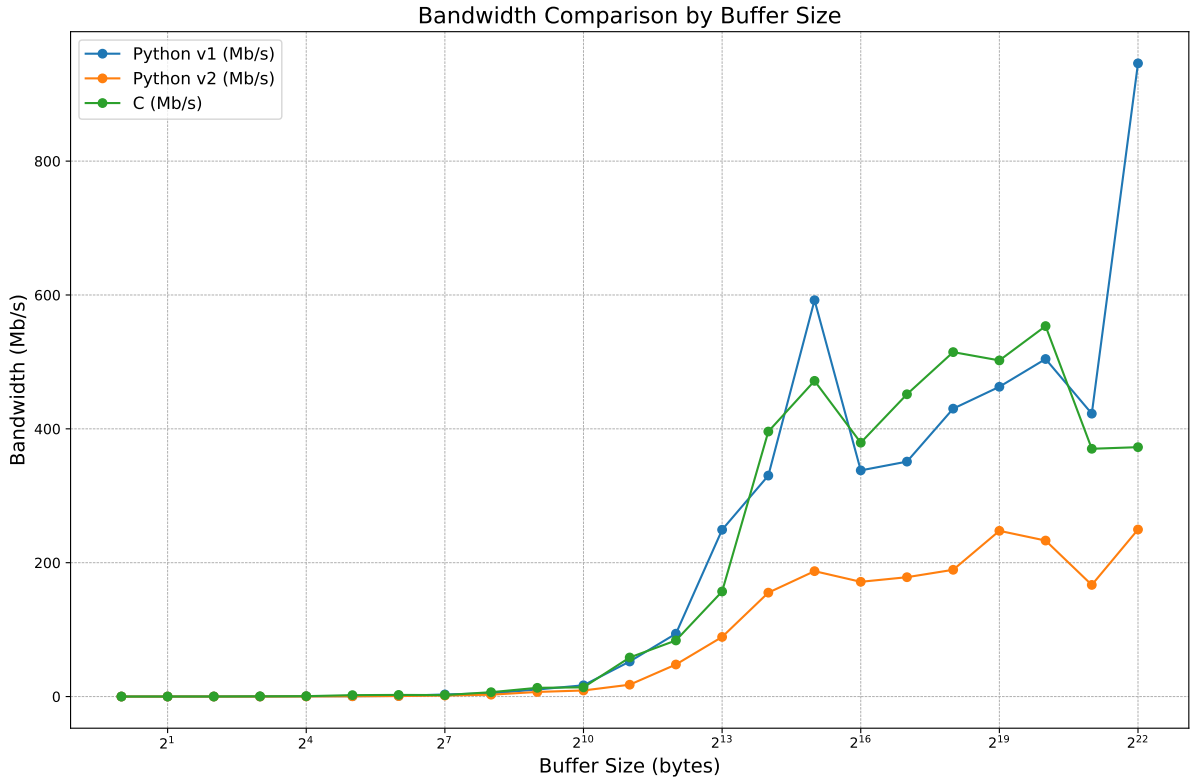

Figure 1: MPI Bi-Directional Bandwidth benchmark.