Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра суперкомпьютеров и квантовой информатики

Отчёт по практическому заданию по курсу "Параллельная обработка больших графов". Параллельная реализация алгоритма дельта-степпинг решения задачи SSSP в ориентированном графе с весами.

Ухин Сергей Алексеевич

523 группа

OpenSHMEM

Москва, 2021

1 Характеристика вычислительной системы и используемого ПО.

Работа выполнялась на кластере «Ангара-К1».

Кластер состоит из 36 узлов, соединённых сетью Ангара с адаптерами на базе СБИС ЕС8430.

Топология сети — 4D-тор размерности 3 × 3 × 2 × 2 (X × Y × Z × K). Пары узлов по направлению K соединены одним линком, по направлению Z — двумя линками.

Использовались вычислительные узлы A типа:

1. A-узлы (24 шт.):
    - сервер Supermicro X9DRW-3LN4F+/X9DRW-3TF+ (BIOS rev. 1.0);
    - 2 процессора Intel Xeon CPU E5-2630 @ 2.30GHz по 6 ядер (всего 12 ядер);
    - Операционная система: SLES 11 SP4 (kernel 3.0.101-63-default x86_64);
    - Память 64 ГБ ;

2 Описание программной реализации.

Алгоритм представляет из себя цикл, который можно разбить на несколько этапов:

1. Все процессы решают, какой номер бакета следует обрабатывать, тут используется функция `shmem_int_min_to_all`. Если этот номер равен INF_BUCKET, то цикл заканчивается, INF_BUCKET содержит недостижимые вершины.
2. Все процессы делают основной цикл алгоритма delta-stepping. Если вершина до которой надо пересчитать расстояние находится на этом же узле и новое расстояние меньше предыдущего, то текущая вершина запоминается. Если вершина находится не на этом процессе, то новое расстояние посылается процессу владельцу с помощью `shmem_double_put` и потом он решает, нужно ли обновлять расстояние или нет. Буферы для таких обменов выделяются заранее с помощью `shmalloc`.
3. Для всех собственных вершин, до которых изменилось расстояние, обновляется их бакет.
4. После этого идет проверка сообщений от других процессов, если нужно изменяется расстояние и бакет измененных вершин.

3 Проверка корректности.

```
ukhin@head:~/test/graph> ./gen_random -s 14
ukhin@head:~/test/graph> ./graphs_reference -in random-14
ukhin@head:~/test/graph> srun -N 1 --ntasks-per-node=4 --nodelist=A22 ./sssp_shmem -in random-14 -out shmem_
ukhin@head:~/test/graph> ./compare random-14.v shmem_14_1_4
norm = 0
Ok
ukhin@head:~/test/graph> srun -N 2 --ntasks-per-node=4 --nodelist=A22 ./sssp_shmem -in random-14 -out shmem_
ukhin@head:~/test/graph> ./compare random-14.v shmem_14_2_4
norm = 0
Ok
ukhin@head:~/test/graph> ./compare shmem_14_1_4 shmem_14_2_4
norm = 0
Ok
ukhin@head:~/test/graph> ./gen_RMAT -s 15
ukhin@head:~/test/graph> ./graphs_reference -in rmat-15.txt -out rmat-15
ukhin@head:~/test/graph> srun -N 1 --ntasks-per-node=4 --nodelist=A22 ./sssp_shmem -in rmat-15.txt -out rmat
ukhin@head:~/test/graph> ./compare rmat-15 rmat_shmem_14_1_4
norm = 0
Ok
ukhin@head:~/test/graph> srun -N 2 --ntasks-per-node=4 --nodelist=A22 ./sssp_shmem -in rmat-15.txt -out rmat
ukhin@head:~/test/graph> ./compare rmat-15 rmat_shmem_14_2_4
norm = 0
Ok
ukhin@head:~/test/graph> ./compare rmat_shmem_14_2_4 rmat_shmem_14_1_4
norm = 0
Ok
ukhin@head:~/test/graph>
```

4 Результаты запусков.

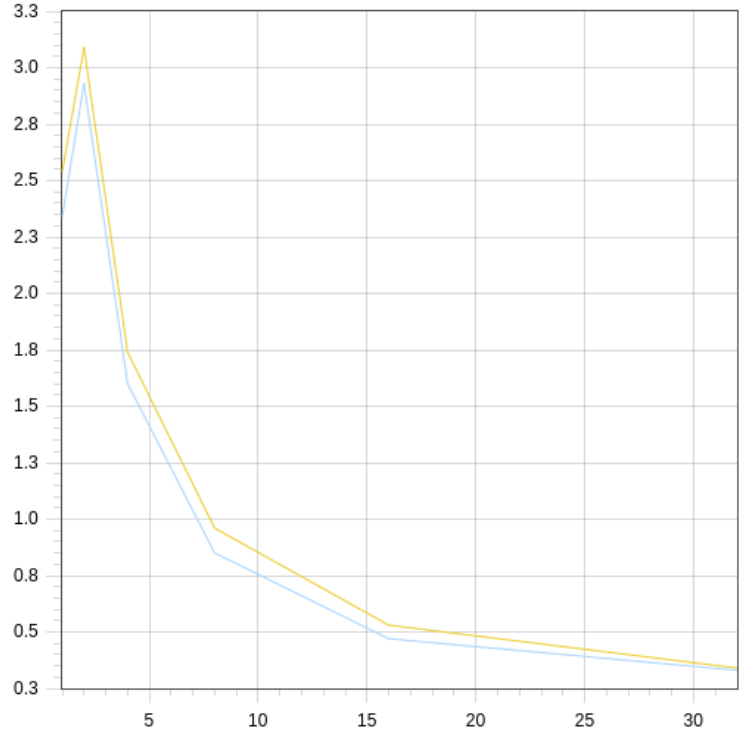

рис1. Результаты запусков для графов размером 2^18. Желтый - Rmat, Синий - uniform. Слева время работы алгоритма в секундах. Снизу количество запущенных процессов(1, 2, 4, 8, 16, 32).

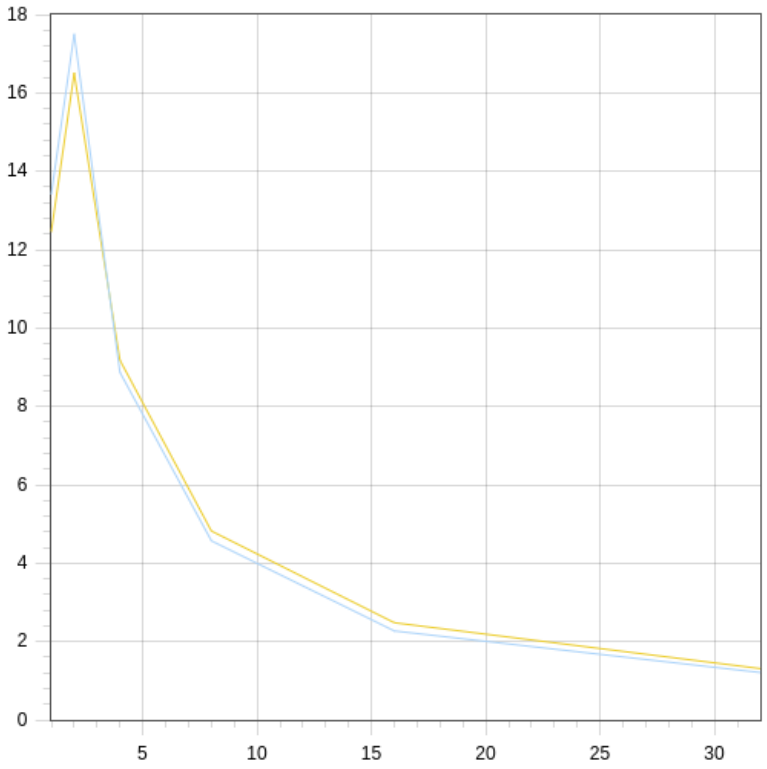

рис2. Результаты запусков для графов размером 2^20. Желтый - Rmat, Синий - uniform. Слева время работы алгоритма в секундах. Снизу количество запущенных процессов(1, 2, 4, 8, 16, 32).

	2^18		2^20	
Size	rmat	uniform	rmat	uniform
1	2.54	2.34	12.46	13.42
2	3.09	2.93	16.5	17.5
4	1.74	1.60	9.18	8.87
8	0.96	0.85	4.82	4.57
16	0.53	0.47	2.48	2.27
32	0.34	0.33	1.31	1.21

Увеличение времени выполнения при переходе с одного процесса на два скорее всего связано с тем, что при выполнении на одном процессе отсутствуют межузловые коммуникации. При дальнейшем увеличении количества процессов алгоритм показывает хорошую сильную масштабируемость: независимо от размера графа при увеличении количества используемых процессов его время работы падает.