АСОБОИ

Лабораторная работа №1. Веб-графы

Авласов Владислав

Тест Graph500

Запускал на ноутбуке с процессором Intel Core i7 7700 HQ CPU @ 2.80GHz. Процессор 8-ядерный.

Оперативная память: DDR4, 16 GB, 2400 MHz.

Кэш процессора:

Запускал graph500 трижды в 4 потока, scale = 22. Ноутбук начал греться, поэтому скинул скорость ядер, как я понял. Результаты 2-го и 3-го запусков хуже 1-го. В итоге получил следующие данные:

	1-ый запуск	2-ой запуск	3-ий запуск
SCALE	22	22	22
edgefactor	16	16	16
NBFS	64	64	64
graph_generation	8.25283	13.6166	32.5385
num_mpi_processes	4	4	4
construction_time	4.17888	5.81218	10.9828
bfs min_time	0.838713	1.1466	2.67774
bfs firstquartile_time	2.73808	1.16178	2.74364

bfs median_time	2.84608	1.20074	2.78115
bfs thirdquartile_time	3.13257	1.23061	2.85971
bfs max_time	3.80319	1.33574	3.47544
bfs mean_time	2.75368	1.20462	2.84557
bfs stddev_time	0.708632	0.0474363	0.176963
min_nedge	67106224	67106224	67106224
firstquartile_nedge	67106224	67106224	67106224
median_nedge	67106224	67106224	67106224
thirdquartile_nedge	67106224	67106224	67106224
max_nedge	67106224	67106224	67106224
mean_nedge	67106224	67106224	67106224
stddev_nedge	0	0	0
bfs min_TEPS	1.76447e+07	5.0239e+07	1.93087e+07
bfs firstquartile_TEPS	2.14221e+07	5.4531e+07	2.34661e+07
bfs median_TEPS	2.35784e+07	5.58872e+07	2.41289e+07
bfs thirdquartile_TEPS	2.45085e+07	5.77618e+07	2.44588e+07
bfs max_TEPS	8.0011e+07	5.85264e+07	2.50607e+07
bfs harmonic_mean_TEPS	! 2.43696e+07	! 5.57073e+07	! 2.35827e+07
bfs harmonic_stddev_TEPS	790105	276377	184773
bfs min_validate	3.2749	4.44206	8.70795
bfs firstquartile_validate	8.95208	4.51188	8.94208
bfs median_validate	9.56683	4.53688	9.05013
bfs thirdquartile_validate	9.90752	4.55522	9.17364
bfs max_validate	11.7437	5.31594	11.2353
bfs mean_validate	9.04749	4.57838	9.18054
bfs stddev_validate	2.01146	0.165639	0.451986

Валидация идёт невероятно долго, возможно из-за неё такое падение производительности.

Облачные вычислительные мощности

Запускал на Гугл-облаке, 16 процессоров, каждый со следующими характеристиками:

Тип машины

n1-standard-16 (16 виртуальных ЦП, 60 ГБ памяти)

Платформа ЦП

Intel Haswell

```
^Cbarabasha999@instance-1:~/graph500-graph500-3.0.0/src$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                16
On-line CPU(s) list:   0-15
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU @ 2.30GHz
Stepping:              0
CPU MHz:               2300.000
BogoMIPS:              4600.00
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              46080K
```

Подробней о кэше каждого процессора:

```
barabasha999@instance-1:~/graph500-graph500-3.0.0/src$ getconf -a | grep CACHE
LEVEL1_ICACHE_SIZE                 32768
LEVEL1_ICACHE_ASSOC                8
LEVEL1_ICACHE_LINESIZE             64
LEVEL1_DCACHE_SIZE                 32768
LEVEL1_DCACHE_ASSOC                8
LEVEL1_DCACHE_LINESIZE             64
LEVEL2_CACHE_SIZE                  262144
LEVEL2_CACHE_ASSOC                 8
LEVEL2_CACHE_LINESIZE              64
LEVEL3_CACHE_SIZE                  47185920
LEVEL3_CACHE_ASSOC                 20
LEVEL3_CACHE_LINESIZE              64
LEVEL4_CACHE_SIZE                  0
LEVEL4_CACHE_ASSOC                 0
LEVEL4_CACHE_LINESIZE              0
barabasha999@instance-1:~/graph500-graph500-3.0.0/src$ mpiexec -np 16 graph500
```

Запускал со scale=25, команда:
mpiexec -np 16 graph500_reference_bfs_sssp 25

Получил следующий результат:

NBFS	64
graph_generation	30.3344
num_mpi_processes	16
construction_time	36.7665
bfs min_time	3.40979
bfs firstquartile_time	3.52353
bfs median_time	3.58881
bfs thirdquartile_time	3.63565
bfs max_time	4.16831
bfs mean_time	3.59363
bfs stddev_time	0.112706
sssp min_time	8.14627
sssp firstquartile_time	11.1741
sssp median_time	11.7889
sssp thirdquartile_time	12.6896
sssp max_time	18.2366
sssp mean_time	11.8976
sssp stddev_time	1.47596
min_nedge	536861780
firstquartile_nedge	536861780
median_nedge	536861780
thirdquartile_nedge	536861780
max_nedge	536861780
mean_nedge	536861780

stddev_nedge	0
bfs min_TEPS	1.28796e+08
bfs firstquartile_TEPS	1.47666e+08
bfs median_TEPS	1.49593e+08
bfs thirdquartile_TEPS	1.52365e+08
bfs max_TEPS	1.57447e+08
bfs harmonic_mean_TEPS	! 1.49392e+08
bfs harmonic_stddev_TEPS	590297
sssp min_TEPS	2.94388e+07
sssp firstquartile_TEPS	4.23072e+07
sssp median_TEPS	4.55396e+07
sssp thirdquartile_TEPS	4.80453e+07
sssp max_TEPS	6.59028e+07
sssp harmonic_mean_TEPS	! 4.51234e+07
sssp harmonic_stddev_TEPS	705257
bfs min_validate	15.9571
bfs firstquartile_validate	16.3365
bfs median_validate	16.4824
bfs thirdquartile_validate	16.6503
bfs max_validate	19.764
bfs mean_validate	16.5722
bfs stddev_validate	0.503936
sssp min_validate	10.4234
sssp firstquartile_validate	10.6143
sssp median_validate	10.6943
sssp thirdquartile_validate	10.8316
sssp max_validate	23.7676
sssp mean_validate	10.9696

Затем запускал без sssp части, просто bfs, т.е. команда:

mpiexec -np 16 graph500_reference_bfs 25

Запускал 4 раза, но последние 3 раза запускал в цикле, т.е.:
for i in {0..2}; do mpiexec -np 16 graph500_reference_bfs 25 > "f${i}.txt" | echo "${i} done" ; done

Наверное, поэтому время на создание так увеличилось (90 секунд против 30 секунд). Не ожидал такого поведения, возможно, это моя ошибка. Но как есть, вот результаты:

	1 (запуск вне цикла)	2	3	4
SCALE	25	25	25	25
edgefactor	16	16	16	16
NBFS	64	64	64	64
graph_generation	29.5354	94.8939	86.5287	90.6361
num_mpi_processes	16	16	16	16
construction_time	29.3744	90.7266	81.0723	94.929
bfs min_time	3.58295	8.05167	7.78328	7.77773
bfs firstquartile_time	3.64658	13.3681	13.2578	13.2915
bfs median_time	3.68737	13.773	13.6742	13.8351
bfs thirdquartile_time	3.72698	14.0797	14.0243	14.2319
bfs max_time	4.14471	16.0582	16.2254	15.7604
bfs mean_time	3.69817	13.6387	13.337	13.3758
bfs stddev_time	0.0855578	1.20534	1.48626	1.7884
min_nedge	536861780	536861780	536861780	536861780
firstquartile_nedge	536861780	536861780	536861780	536861780
median_nedge	536861780	536861780	536861780	536861780
thirdquartile_nedge	536861780	536861780	536861780	536861780
max_nedge	536861780	536861780	536861780	536861780
mean_nedge	536861780	536861780	536861780	536861780
stddev_nedge	0	0	0	0
bfs min_TEPS	1.29529e+08	3.34323e+07	3.30876e+07	3.4064e+07

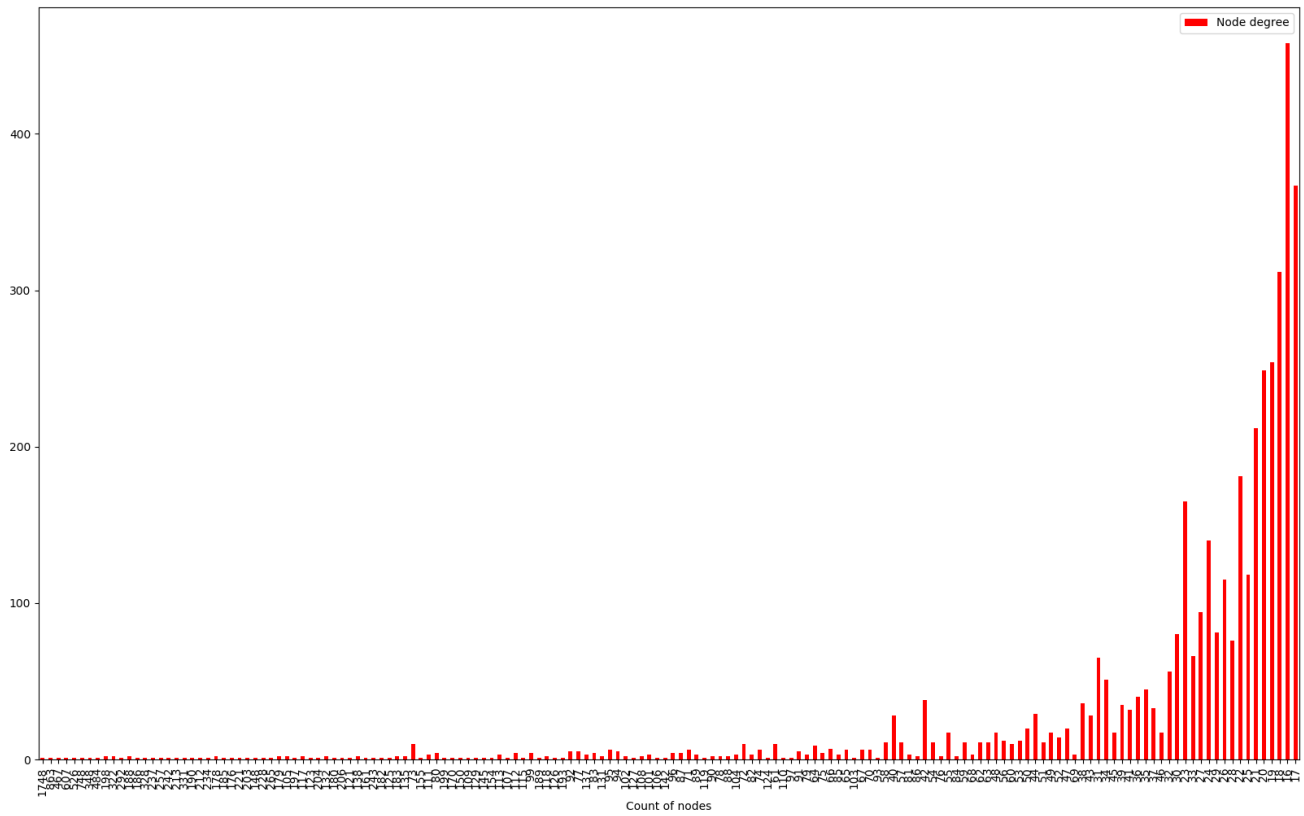
bfs firstquartile_TEPS	1.44048e+08	3.81303e+07	3.82808e+07	3.77224e+07
bfs median_TEPS	1.45595e+08	3.89792e+07	3.92608e+07	3.88043e+07
bfs thirdquartile_TEPS	1.47223e+08	4.01598e+07	4.04942e+07	4.03914e+07
bfs max_TEPS	1.49838e+08	6.66771e+07	6.89763e+07	6.90255e+07
bfs harmonic_mean_TEPS	! 1.45169e+08	! 3.93632e+07	! 4.02535e+07	! 4.01368e+07
bfs harmonic_stddev_TEPS	423133	438286	565158	676109
bfs min_validate	16.0983	32.588	34.306	32.936
bfs firstquartile_validate	16.3525	53.304	50.9212	51.2175
bfs median_validate	16.4572	54.2589	52.8807	54.1152
bfs thirdquartile_validate	16.6004	55.0236	54.2581	54.9468
bfs max_validate	23.0333	75.1139	70.3548	72.5168
bfs mean_validate	16.6208	53.883	51.7624	52.1986
bfs stddev_validate	0.864104	4.99214	5.5411	6.11727

Модель Боллобаша-Риордана

Реализовывал при помощи питоновской библиотеки networks. К сожалению, если генерировать и добавлять вершины по одной, а потом считать характеристики получившегося графа, то выходит невероятно медленно. Один граф с m=14 у меня считается порядка часа. Определённо, есть мой небольшой кусок кода, который можно оптимизировать, сейчас тоже добавляет пару минут к суммарному времени, но это не так много по сравнению с временем работы самой библиотеки.

Знаю, что лучше было написать свою генерацию матрицы смежности для большей скорости, но уж как получилось, скорость не в приоритете была.

Итак, гистограмма #(n,d) для одной из получившихся сгенерированных моделей:



В итоге из-за низкой скорости работы я сгенерировал лишь по одному графу для каждого значения m в интервале m={10..16}.

Все графы получились связными, в каждом по 4000 вершин.

Вот результаты:

m	10	11	12	13	14	15	16
Диаметр	4	4	4	4	4	4	4
Среднее кратчайшее расстояние	2.828286	2.748293	2.724987	2.680663	2.653335	2.614564	2.594322

Можно заметить, что с увеличением количества рёбер у каждой вершины среднее значение пути между вершинами уменьшается.

Так как такое время работы меня удивило, я добавил профилирование в свой код, поэтому прикреплю к этому отчёту сырой вывод утилит питона cProfile и pstats для графа с m=16, а также оригинальную гистограмму, так как не уверен, что полное качество сохранится при переносе в отчёт.

Ну и исходники генерации самого графа, конечно же.

Спасибо за внимание!