Отображение алгоритма поиска в ширину на больших графах на различные одноузловые параллельные архитектуры

Колганов А.С.
МГУ ВМК / ИМП РАН

Поиск в ширину (BFS)

- Breadth first search – один из важных и фундаментальных алгоритмов в обработке графов;

- Алгоритмические трудности BFS:
  - Очень мало вычислений;
  - Нерегулярный доступ к памяти.

Graph500 и GreenGraph500

- Использование алгоритмов BFS и SSSP для ранжирования суперкомпьютеров (**TEPS** – обработка количества ребер в секунду);
- Использование метрики MTEPS / WATT для ранжирования в рейтинге энергоэффективных машин;
- Альтенатива рейтингам TOP500/GREEN500
- Оба списка до сих пор не заполнены:
  - Graph 500 (занято 235 позиций);
  - GreenGraph 500 (занято ~70 позиций).

Graph500 – поиск в графе

- Генерация ребер;
- Построение графа по полученным ребрам *(замеряется время, входит в таблицу);*
- Генерация 64 произвольных вершин из которых выполняется BFS;
- Для каждой вершины:
  - Выполнение алгоритма BFS *(замеряется время, входит в рейтинг);*
  - Выполнение процедуры проверки корректности;
- Вывод результирующей информации.

Реализация BFS

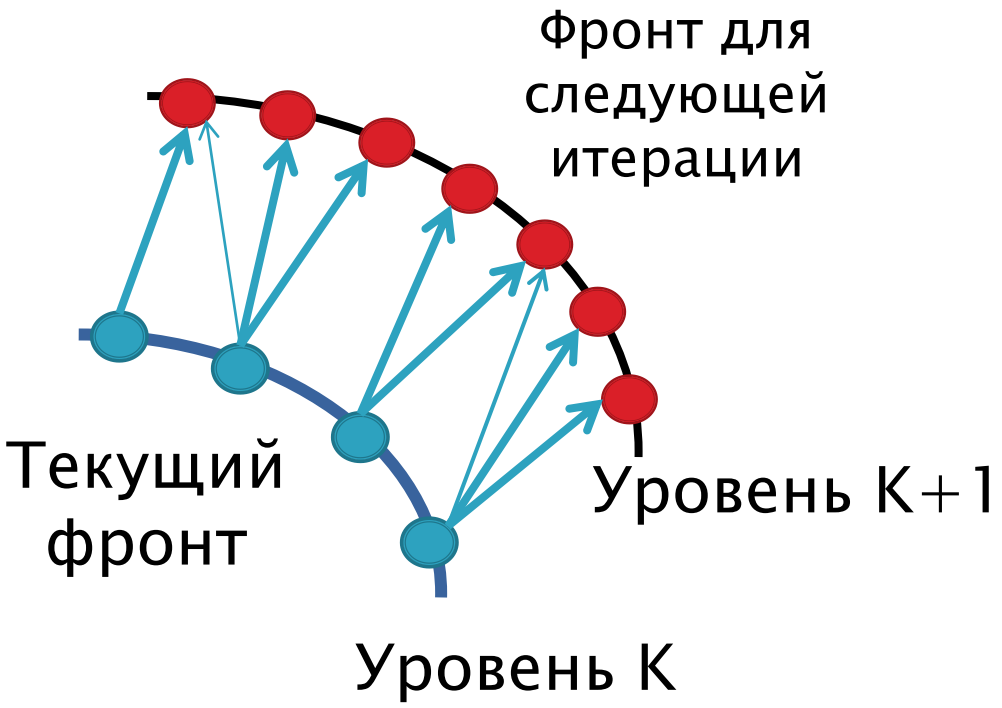
- Фаза 1 (преобразования):
  - преобразование полученного от генератора графа в CSR;
  - Преобразование графа CSR
- Фаза 2 (алгоритм):
  - основной цикл алгоритма по 64 ключам;
  - гибридный алгоритм Top Down + Bottom Up.
- Реализация основана на идеях GraphCREST: *«Fast and Energy-efficient Breadth-First Search on a Single NUMA System, 2014»*

Реализация BFS: преобразования

- Преобразование набора вершин в CSR (compressed sparse rows);
- Глобальная сортировка вершин по степени связности;
- Локальная сортировка соседей по степени связности;
- «Цепная» перенумерация всех вершин графа;
- Частичное выравнивание данных для GPU;
- Сжатие списка вершин соседей для CPU.

Реализация BFS: алгоритм

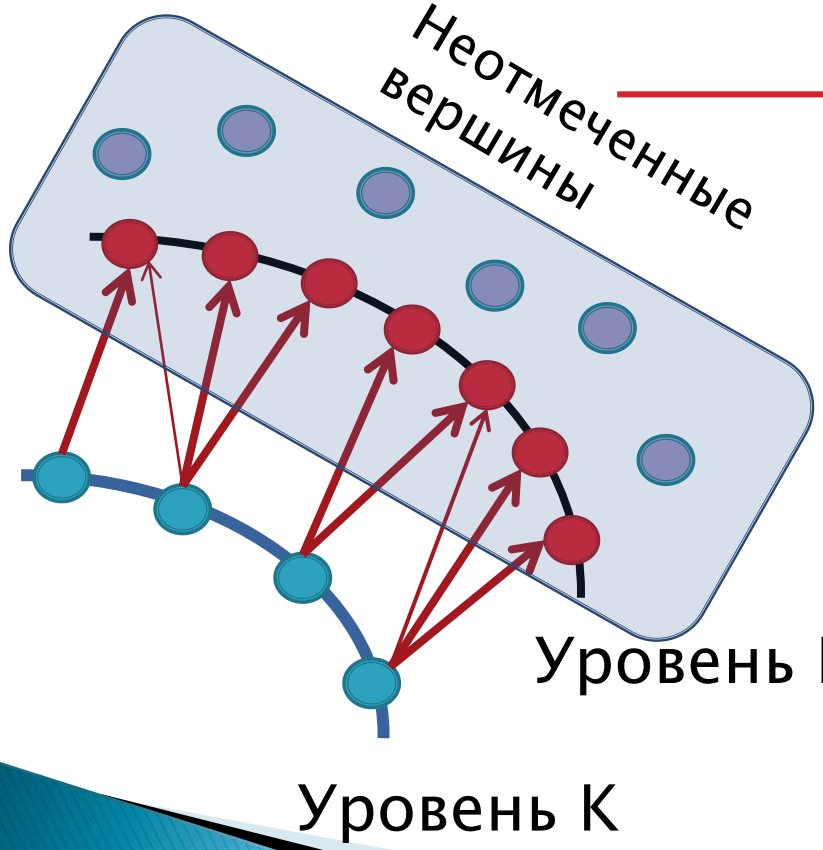
- Синхронный по уровням **Top-Down** обход




```
foreach (i = [0, N])
{
    foreach (k = [rInd[i], rInd[i+1])
    {
        unsigned v = endV[k];
        if (levels[v] == 0)
        {
            levels[v] = lvl;
            parents[v] = i;
        }
    }
}
```

Реализация BFS: алгоритм

- Синхронный по уровням **Bottom-Up** обход




```
foreach (i = [0, N])
{
    if (levels[i] == 0)
    {
        foreach (k=[rInd[i], rInd[i+1] ])
        {
            unsigned endk = endV[k];
            if (levels[endk] == lvl - 1)
            {
                parents[i] = endk;
                levels[i] = lvl;
                break;
            }
        }
    }
}
```

Реализация BFS: алгоритм

- Гибридный алгоритм **Top-Down** + **Bottom-Up** (оптимизация по направлениям)

Граф SCALE 26 |V| = 2^26 (67,108,864) |E| = 2^30 (1,073,741,824)

Уровень	Top-Down	Bottom-Up	Hybrid
0	2	2,103,840,895	2
1	66,206	1,766,587,029	66,206
2	346,918,235	52,677,691	52,677,691
3	1,727,195,615	12,820,854	12,820,854
4	29,557,400	103,184	103,184
5	82,357	21,467	21,467
6	221	21,240	221
Всего:	2,103,820,036	3,936,072,360	65,689,625
	100%	187%	3.12%

= 2x|E|

Существенное уменьшение количества просмотренных ребер

Реализация BFS: алгоритм

- Гибридный алгоритм **Top-Down** + **Bottom-Up** (оптимизация по направлениям)

Граф SCALE 26 |V| = 2^26 (67,108,864) |E| = 2^30 (1,073,741,824)

Уровень	Top-Down	Bottom-Up	Hybrid	Выбор обхода
0	2	2,103,840,895	2	Фаза роста
1	66,206	1,766,587,029	66,206	
2	346,918,235	52,677,691	52,677,691	
3	1,727,195,615	12,820,854	12,820,854	Фаза спада
4	29,557,400	103,184	103,184	
5	82,357	21,467	21,467	
6	221	21,240	221	
Всего:	2,103,820,036	3,936,072,360	65,689,625	
	100%	187%	3.12%	

= 2x|E|

Реализация BFS: оптимизации для GPU

- Использование CUDA Dynamic Parallelism для балансировки нагрузки в Top-Down;
- Использование «векторизации» внутри каждого потока (чтение 4x значений levels за 1 операцию);
- Использование частичного выравнивания части ребер для Bottom-Up (преобразование);
- Использование NVlink для доступа к большому графу с хоста (IBM + P100);
- Использование Bottom-Up с очередью и разделяемой памятью на более поздних итерациях.

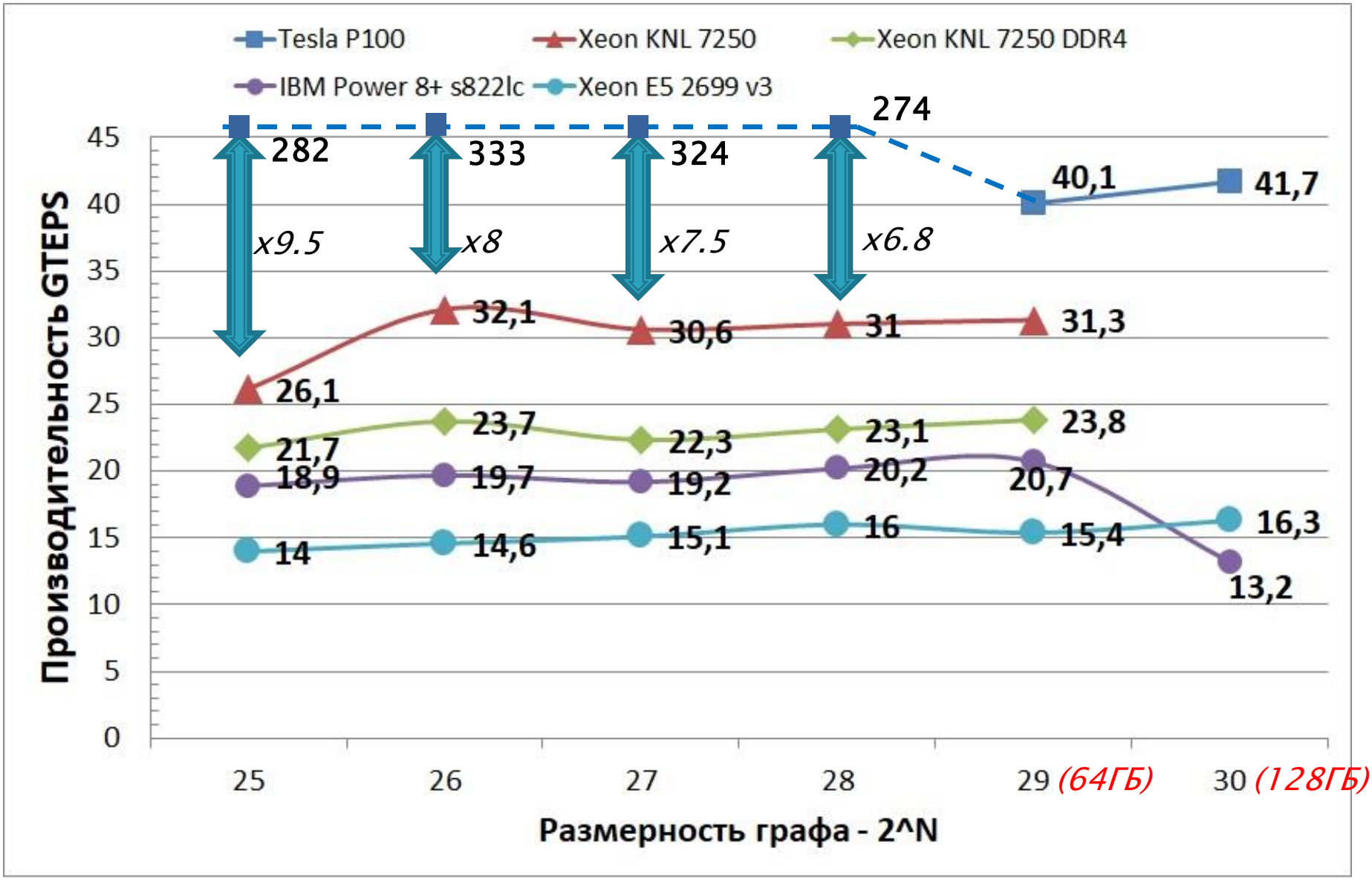
Реализация BFS: оптимизации для CPU

- Использование директив OpenMP, в том настраиваемой клаузы schedule;
- Использование локального сжатия списка вершин соседей (преобразование);
- Использование HBM памяти для хранения промежуточных данных (для KNL);
- Использование очередей для реализации алгоритма.

Тестирование: архитектуры

Вендор	Ядер/ потоков	Частота ГГц	RAM, GB/s	Макс. TDP, Вт	Транз., млрд
Xeon KNL 7250	68 / 272	1.4	115 / 400	215	~8
Xeon E5 2699 v3	18 / 36	2.3	68	145	~5.69
Power 8+ s822lc	10 / 80	3.5	205	270	~6
Tesla P100	56 / 3584	1.4	40 / 700	300	~15.3

- Компиляторы:
  для **Intel Xeon E5** — gcc 5.3, для **Intel KNL** — icc v17,
  для **IBM** — gcc 5.3 Power, для **NVidia** — CUDA 9.0


Tesla P100
Xeon KNL 7250
Xeon KNL 7250 DDR4
IBM Power 8+ s822lc
Xeon E5 2699 v3
Производительность GTEPS
Размерность графа - 2^N
0
5
10
15
20
25
30
35
40
45
25
26
27
28
29 (64ГБ)
30 (128ГБ)
282
333
324
274
x9.5
x8
x7.5
x6.8
26,1
32,1
30,6
31
31,3
21,7
23,7
22,3
23,1
23,8
18,9
19,7
19,2
20,2
20,7
13,2
14
14,6
15,1
16
15,4
16,3
40,1
41,7

Рейтинг Graph500 && GGraph500

- **IBM+P100** – лидирующая позиция для больших (41.7 GTEPS/ 0.17 GTEPS/w) и малых (220 GTEPS / 1.23 GTEPS/w) данных в GG500 и в G500 **среди всех односокетных систем**;

- **Intel KNL** – лучшая производительность на одной карте среди всех Intel Xeon Phi (31.3 GTEPS / 0.13 GTEPS/w);

- **Intel Xeon E5 v3** – лучшая производительность в одном сокете среди всех односокетных CPU систем.
  (16.3 GTEPS / 0.08 GTEPS/w);

Спасибо за внимание!

ссылка на статью: https://habrahabr.ru/post/344378/
(первая статья по ключевому слову BFS)

ссылка на рейтинг: http://graph500.org/

Александр Колганов, МГУ ВМК / ИПМ РАН,
alexander.k.s@mail.ru