N Nearest Neighbour

Syed Hassan Jalil
Trisha Anand

Definition

- Algorithm to find your closest N Neighbours
- 2 Dimensional data (x,y)
- Euclidean Distance without taking square root

Algorithm

- Loop Over All Points (M total Points)
- For each point, Calculate distance to All other points (M^2)
- Sort Neighbors of each point in ascending Order (M^3)
- Write the top N neighbor of every point to output file

Struct

We have the following Structure to store our data

```
struct Nodes {
   int x;  // x-coordinate
   int y; // y-coordinate
   double *neighbourDistance; // array to store distance to neighbour
   int *neighbourID; // array to store sorted neighbours index
 };
```

Arithmetic Intensity - Distance Calculation

Computation : 6+1

Memory Access : (⅓ x (24) + 8 + 4) *

AI : 7/20 = 0.35

* Entire Struct is read into the cache line every time its variable is accessed

Arithmetic Intensity - Sorting

Computation : 4

Memory Access : 8+8+8+4+4 = 32

AI = 4/32 = 0.125

Cleary Memory intensive operation

Roofline Model - DAS4

DAS4 - Intel(R) Xeon(R) E5620
Sequential peak floating point performance = 2.4 GHz * 2 IPC = 4.8
Memory Bandwidth = 25.6 GB/s

Maximum Attainable Performance
For Distance Calculation - MIN (4.8, (0.35* 25.6)) = MIN (4.8,8.96) = 4.8 GFLOPS (CPU Bound)
For Sorting = MIN(4.8, (0.125 * 25.6)) = MIN (4.8, 3.2) = 3.2 GFLOPS (Memory Bound)

Roofline Model - Personal Computer

Personal Computer - Intel® Core™ i5-6360U
Sequential peak floating point performance = 2.0 GHz * 2 IPC = 4.0
Memory Bandwidth = 34.1 GB/s

Maximum Attainable Performance
For Distance Calculation - MIN (4.0, (0.35* 34.1)) = MIN (4.8,11.935) = 4.8 GFLOPS (Compute Bound)
For Sorting = MIN(4.0, (0.125 * 34.1)) = MIN (4.0,4.26) = 4.26 GFLOPS (Compute Bound)

Basic Model For Distance Calculation

M total Points

Time = t_compute + t_communicate

We have no communication in our code

t_compute = t_addition+ t_multiplication + t_sqrt

Total Additions = M+(6xM^2)

Total Multiplication = 2xM^2

Total Sqrt =M^2

T_compute = ((M+(6xM^2))*t_addition)+ ((2xM^2)*t_multiplication)+((M^2)*t_sqrt)

Basic Model - Sorting

M total Points

T= t_compute + t_communicate

T_communicate = 0

T_compute = t_addition

Total Addition = M + M^2 + (5xM^3)

T= (M + M^2 + (5xM^3))xt_addition + 0

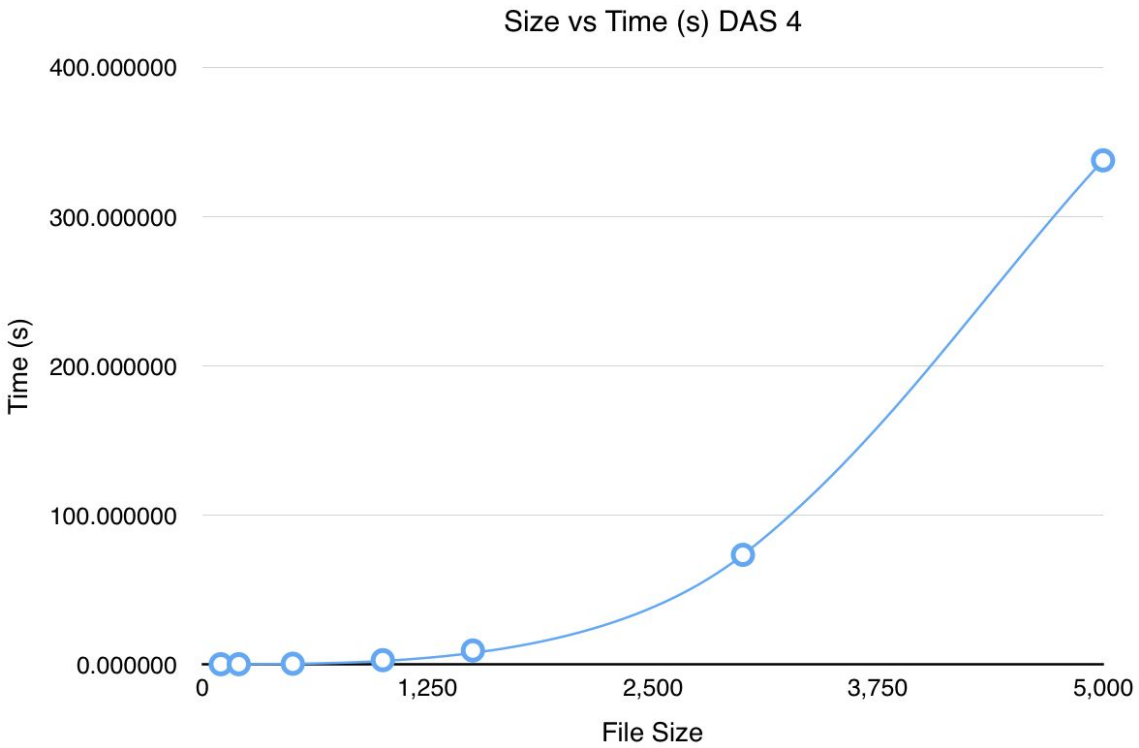
Evaluation - DAS4

- Hardware :
  - DAS4
  - Intel(R) Xeon(R) E5620 (2.4 Ghz 12MB Cache)
  - 24 GB RAM
- Software
  - CentOS v 6.6
- Method
  - Using input files of different sizes (number of nodes)
  - Files generated randomly using python script (provided with code)
  - Time calculated only for Kernel (Distance calculation + Sorting) and not for reading and writing file
  - Ran for top 10 Neighbours

Time for Execution

Number of nodes	Time in Seconds
100	0.002379
200	0.020641
500	0.338837
1000	2.719252
1500	9.211656
3000	73.285944
5000	337.580736

Visualization



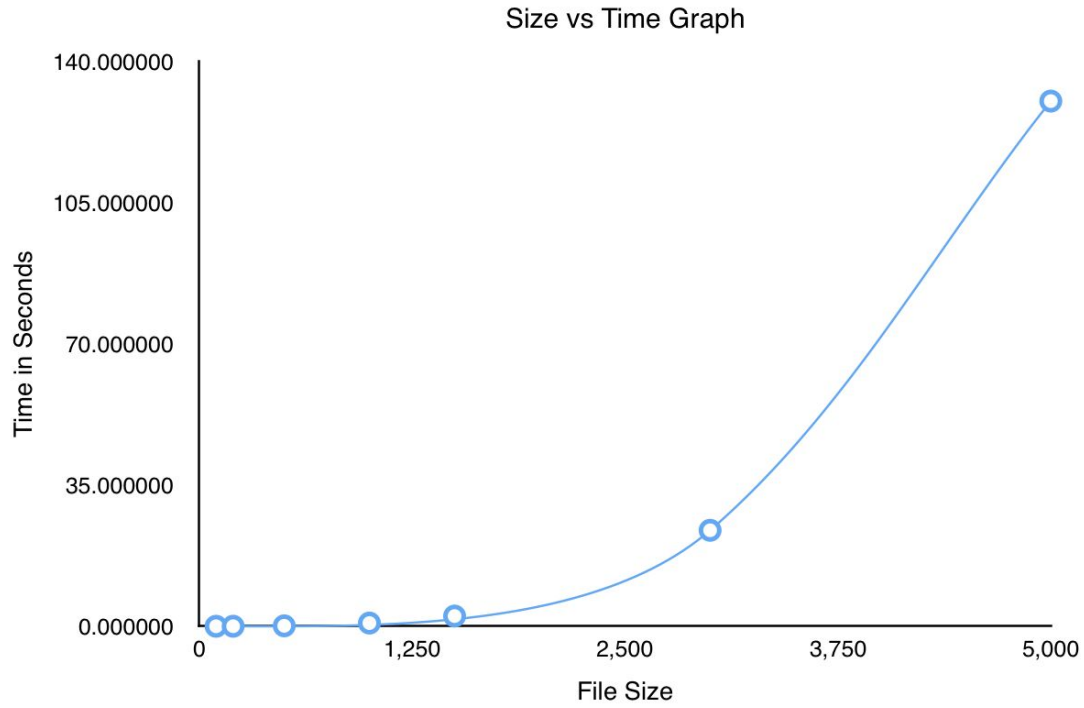
Evaluation - Personal Computer

- Hardware :
  - Macbook Pro 2016 (w/o touchbar)
  - Intel® Core™ i5-6360U (2.0 Ghz 4MB Cache)
  - 8 GB RAM
- Software
  - macOS Sierra v 10.12.4
- Method
  - Using input files of different sizes (number of nodes)
  - Files generated randomly using python script (provided with code)
  - Time calculated only for Kernel (Distance calculation + Sorting) and not for reading and writing file
  - Ran for top 10 Neighbours

Time for Execution

Number of nodes	Time in Seconds
100	0.001542
200	0.010451
500	0.144065
1000	0.904714
1500	2.868879
3000	24.664856
5000	148.987456

Visualization



Profiling

Profiled on DAS4 to measure Cache Miss.

Very High rate of Cache Miss

```
[pmms1701@node021 Nearest Neighbor]$ perf stat -B -e cache-references,cache-misses ./main 1000 1000.txt 10
2699962.000000

 Performance counter stats for './main 1000 1000.txt 10':

          335,712 cache-references
          200,568 cache-misses              #   59.744 % of all cache refs

       2.705961076 seconds time elapsed

[pmms1701@node021 Nearest Neighbor]$
```

Future Work

- Improve Data Locality
- Code Motion
- Parallelizing Distance Calculation till its memory intensive (OMP? GPU ? SPARK !!!!!)
- Optimizing Sort (Use Max Heap)
- Using LSH (Depends on Time and ability to quantify)