High-Performance Analytics on Large-Scale GPS Taxi Trip Records in NYC

Jianting Zhang

Department of Computer Science
The City College of New York
Outline

• Background and Motivation

• Parallel Taxi data management on GPUs

• Efficient shortest path computation and applications
Background and Motivation

**Taxicabs**
- 13,000 Medallion taxi cabs
- License priced at $600,000 in 2007
- Car services and taxi services are separate

**Taxi trip records**
- ~170 million trips (300 million passengers) in 2009
- 1/5 of that of subway riders and 1/3 of that of bus riders in NYC
Background and Motivation

Over all distributions of trip distance, time, speed and fare (2009)
Background and Motivation

Other types of Origin-Destination (OD) Data

Social network activities

Cellular phone calls
Background and Motivation

• How to manage OD data?
  – Geographical Information System (GIS)
  – Spatial Databases (SDB)
  – Moving Object Databases (MOD)

• How good are they?
  – Pretty good for small amount of data 😊
  – But, rather poor for large-scale data 😞
Background and Motivation

• Example 1:
  – Loading 170 million taxi pickup locations into PostgreSQL
  – UPDATE t SET PUGeo = ST_SetSRID(ST_Point("PULong","PuLat"),4326);
  – 105.8 hours!

• Example 2:
  – Finding the nearest tax blocks for 170 million taxi pickup locations using open source libspatialindex+GDAL
  – 30.5 hours!

I do not have time to wait...

Can we do better?
Background and Motivation

Cloud computing + MapReduce + Hadoop

GPGPU Computing: From Fermi to Kepler

Multicore CPUs
Nvidia GTX Titan GPU sold at Amazon

4.5 Teraflops of single precision and 1.3 Teraflops of double precision
ASCI Red: 1997 First 1 Teraflops (sustained) system with 9298 Intel Pentium II Xeon processors (in 72 Cabinets) $$$?
Space?
Power?

• Feb. 2013
• 7.1 billion transistors (551mm²)
• 2,688 processors
• Max bandwidth 288.4 GB/s
• PCI-E peripheral device
• 250 W (17.98 GFLOPS/W -SP)
• Suggested retail price: $999

What can we do today using a device that is more powerful than ASC I read 16 years ago?
Background and Motivation

- The goal is to design a data management system to efficiently manage large-scale OD/trajectory data on massively data parallel GPUs
- With the help of new data models, data structures and algorithms
- To cut the runtimes **from hours to seconds** on a single commodity GPU device
- And support **interactive queries and visual explorations**
Outline

• Background and Motivation

• Parallel Taxi data management on GPUs

• Efficient shortest path computation and applications
O-D data management on GPUs

Physical Data Layout
- Day
- Month
- Year

Raw data

Compression, aggregation and indexing

Spatial Joins and Shortest Path Computation

(Zhang, Gong, Kamga and Gruenwald, 2012)
O-D data management on GPUs

(Zhang, Gong, Kamga and Gruenwald, 2012)
O-D data management on GPUs

(Zhang, You and Gruenwald, 2012)
O-D data management on GPUs

(Zhang, Gong, Kamga and Gruenwald, 2012)
O-D data management on GPUs

P2N-D

P2P-T

P2P-D

(Zhang, You and Gruenwald, 2012)
O-D data management on GPUs

Single-Level Grid-File based Spatial Filtering

Nested-Loop based Refinement

Points

Vertices
(polygon/polyline)

• Perfect coalesced memory accesses
• Utilizing GPU floating point power

(Zhang, You and Gruenwald, 2012)
O-D data management on GPUs

• Data
  – Taxi trip records: 300 million in two years (2008-2010), ~170 million in 2009 (~150 million in Manhattan)
  – NYC DCPLION street network data: 147,011 street segments
  – NYC Census 2000 blocks: 38,794
  – NYC MapPluto Tax blocks: 735,488 in four boroughs (excluding SI) and 43,252 in Manhattan

• Hardware
  – Dell T5400 Dual Quadcore CPUs with 16 GB memory
  – Nvidia Quadro 6000 with 448 cores and 6 GB memory
O-D data management on GPUs

Top: grid size = 256*256
resolution = 128 feet
Right: grid size = 8192*8192
resolution = 4 feet

Spatial Aggregation
9,424 / 326 = 30X (8192*8192)

Temporal Aggregation
1709 / 198 = 8.6X (minute)
1598 / 165 = 9.7X (hour)

(Zhang, You and Gruenwald, 2012)
O-D data management on GPUs

- 147,011 street segments
- 38,794 census blocks (470941 points)
- 735,488 tax blocks (4,698,986 points)

<table>
<thead>
<tr>
<th></th>
<th>P2N-D</th>
<th>P2P-T</th>
<th>P2P-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>-</td>
<td>15.2 h</td>
<td>30.5 h</td>
</tr>
<tr>
<td>GPU Time</td>
<td>10.9 s</td>
<td>11.2 s</td>
<td>33.1 s</td>
</tr>
<tr>
<td>Speedup</td>
<td>-</td>
<td>4,900X</td>
<td>3,200X</td>
</tr>
</tbody>
</table>

Algorithmic improvement: 3.7X
Using main-memory: 37.4X
Parallel Acceleration: 24.3X

(Zhang, You and Gruenwald, 2012), (Zhang and You 2012a) Zhang and You 2012b)
Outline

• Background and Motivation

• Parallel Taxi data management on GPUs

• Efficient shortest path computation and applications
  • Mapping Betweenness Centralities
  • Outlier Detection
SP computation and applications

- overview

• Shortest path computation
  – Dijkstra and A*
  – New generation algorithms
  – Contraction Hierarchy (CH) based

• Open source implementations of CH:
  • MoNav
  • OSRM

• Much faster than ArcGIS NA module
SP computation and applications

• Network Centrality (Brandes, 2008)

  Node based
  \[ C_B(v) = \sum_{s \neq v \neq t \notin V} \frac{\sigma_{st}(v)}{\sigma_{st}} \]

  Edge based
  \[ C_B(e) = \sum_{s \neq v \neq t \notin V} \frac{\sigma_{st}(e)}{\sigma_{st}} \]

• Can be easily derived after shortest paths are computed

• Mapping node/edge between centrality can reveal the connection strengths among different parts of cities
SP computation and applications

Mapping Betweenness Centralities

- 166 million trips, 25 million unique
- Shortest path computation completes in less than 2 hours (5,952 seconds) on a single CPU core (2.0 GHZ)
- 4200 pairs computation per second
- 3 orders of magnitude faster than ArcGIS NA

Mapping of Computed Shortest Paths Overlaid with NYC Community Districts Map
SP computation and applications

Mapping Betweenness Centralities (All hours)
SP computation and applications

Mapping Betweenness Centralities (bi-hourly)

Legend:
- 0 - 10000
- 10001 - 100000
- 100001 - 1000000
SP computation and applications
The data is not as clean as we had thought…outlier detection

Meshed up on purpose due to privacy concerns
SP computation and applications
Outlier Detection

• Existing approaches for outlier detection for urban computing
  • Thresholding: e.g. 200m < dist < 30km
  • Locating in unusual ranges of distributions
  • Spatial analysis: within a region or a land use type
  • Matching trajectory with road segments – treat unmatched ones as outliers

• Some techniques require complete GPS traces while we only have O-D locations

• Large-scale shortest path computing has not been used for outlier detection
SP computation and applications
The data is not as clean as we had thought…outlier detection

• In addition:
  – Some of the data fields are empty
  – Pickup and drop-off locations can be in Hudson River
  – The recorded trip distance/duration can be unreasonable
  – ...

Outlier detections for data cleaning are needed
SP computation and applications

- Raw Taxi trip data
- Match pickup/drop-off point locations to street segments within Distance $D_0$

Successful?

Assign pickup/drop-off nodes by picking closer ones

- Compute shortest path

$CD > D_1$ AND $CD > W \cdot RD$?

Update centrality measurements

Type I outlier (spatial analysis)

Aggregate unique (sid,tid) pairs

Type II outlier (network analysis)

$CD$: Compute shortest distance
$RD$: Recorded trip distance
SP computation and applications

- Discussion on outlier detection

- The approach is approximate in nature
  - Taxi drivers do not always follow shortest path
  - Especially for short trips and heavily congested areas
  - But we only care about aggregated centrality measurements and the errors have a chance to be cancelled out by each other

- Increasing $D_0$ will reduce # of type I outliers, but the locations might be mismatched with segments

- Reducing $D_1$ and/or $W$ will increase # of type II outliers but may generate false positives.
SP computation and applications

-Results of outlier detection

Examples of Detected Type II Outliers

<table>
<thead>
<tr>
<th></th>
<th>calc_dist numeric</th>
<th>record_dist numeric</th>
<th>record_time numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32.01</td>
<td>3.2</td>
<td>12.05</td>
</tr>
<tr>
<td>2</td>
<td>31.06</td>
<td>3.4</td>
<td>12.46</td>
</tr>
<tr>
<td>3</td>
<td>32.11</td>
<td>3.55</td>
<td>22.0</td>
</tr>
<tr>
<td>4</td>
<td>43.07</td>
<td>4.8</td>
<td>70.18</td>
</tr>
<tr>
<td>5</td>
<td>28.44</td>
<td>3.3</td>
<td>18.0</td>
</tr>
<tr>
<td>6</td>
<td>26.54</td>
<td>3.1</td>
<td>14.06</td>
</tr>
<tr>
<td>7</td>
<td>27.8</td>
<td>3.3</td>
<td>19.75</td>
</tr>
<tr>
<td>8</td>
<td>26.57</td>
<td>3.2</td>
<td>14.56</td>
</tr>
<tr>
<td>9</td>
<td>25.03</td>
<td>3.09</td>
<td>62.0</td>
</tr>
</tbody>
</table>

- $D_0=200$ feet, $D_1=3$ miles, $W=2$
- $\sim 2.5$ millions (1.5%) type I outliers
- $\sim 18,000$ type II outliers
Ongoing Research @ GeoTECI

- **CudaGIS** - a general purposed GIS on GPUs
  - More GIS modules in addition to indexing/query processing

- **Trajectory** data management on GPUs
  - Online moving point location updating
  - Segmentation/simplification/compression
  - Matching with road networks
  - Aggregation and warehousing
  - Indexing and query processing \(\rightarrow\) *similarity join*
  - Data mining (moving cluster, convoy, swarm...)
Q&A

http://www.cs.ccny.cuny.edu/~jzhang/
jzhang@cs.ccny.cuny.edu