High-Performance Online Spatial and Temporal Aggregations on Multi-core CPUs and Many-Core GPUs

Jianting Zhang[1,2] Simin You[2], **Le Gruenwald**[3]

1 Depart of Computer Science, CUNY City College (City College of New York)
2 Department of Computer Science, CUNY Graduate Center
3 School of Computer Science, the University of Oklahoma

Outline

- Introduction
- Background and Motivation
- Spatial, Temporal and Spatiotemporal Aggregations of Taxi Trips
- Implementation Details
- Experiments and Results
- Conclusion and Future Work

Introduction

- Spatial, temporal and spatiotemporal aggregations are commonly used OLAP operations ➔SOLAP, TOLAP, STOLAP
- Several existing OLAP systems are built on top of GIS and Spatial Databases and suffer from low performance when handling large-scale datasets on traditional hardware (disk-resident + serial CPU)
- This research aims at investigating the feasibility and efficiency on spatial, temporal and spatiotemporal aggregations on new hardware (large main-memory + massively data parallel GPUs) using a domain-specific case study (taxi trip records)

Background and Motivation

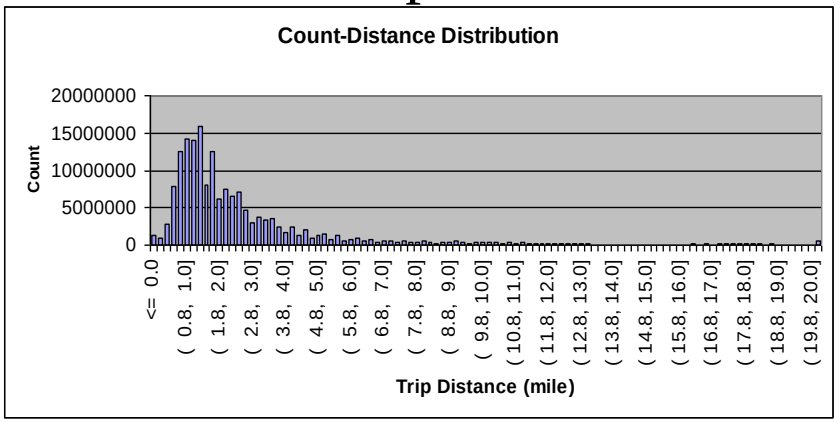
Taxi trip records

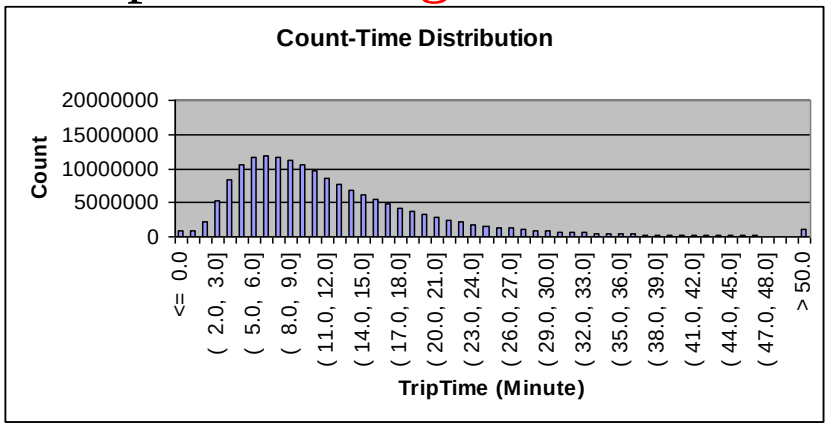
- ~300 million trips in about two years
- ~170 million trips (300 million passengers) in 2009
- 1/5 of that of subway riders and 1/3 of that of bus riders in NYC

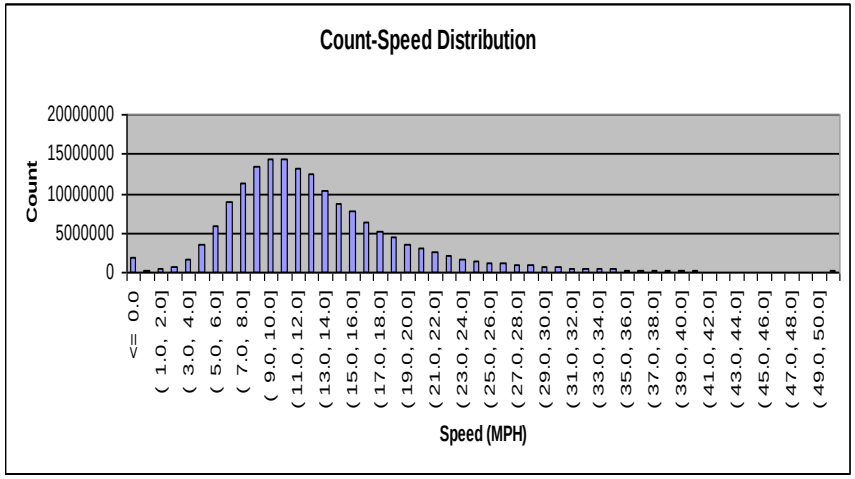
- 13,000 Medallion taxi cabs
- Only taxis with Medallion license are for hail (the rule could be changing outside Manhattan...)

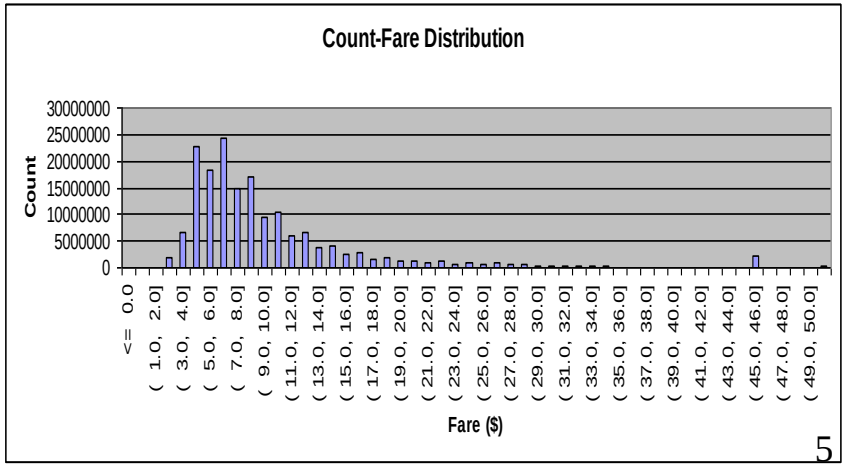
Background and Motivation

Overall distributions of trip distance, time, speed and fare: majority of taxi trips are within 3 miles and cost less than $10: affordable; but the median speed is about 10 miles per hour: significant traffic









Background and Motivation

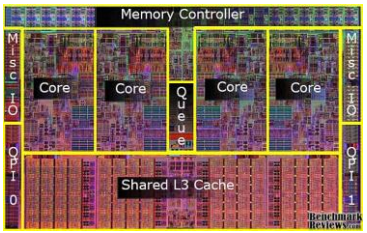
- How to manage taxi trip data?
  - Geographical Information System (GIS)
    - E.g. ESRI ArcGIS
  - Spatial Databases (SDB)
    - E.g., PostgreSQL/PostGIS
  - **Moving Object Databases (MOD)**
    - E.g. Secondo
- How good are they?
  - Pretty good for small amount of data ☺
  - But, rather poor for large-scale data ☹

Background and Motivation

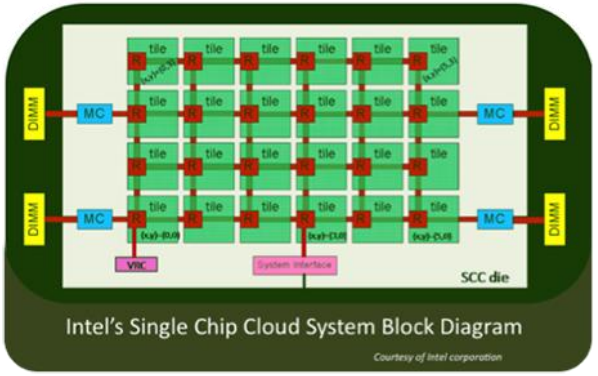
- Example 1:
  - Creating a geometry column from lat/long columns that is necessary for subsequent indexing and query processing in PostgreSQL/PostGIS
  - 170 million taxi pickup locations in 2009
  - UPDATE t SET PUGeo = ST_SetSRID(ST_Point("PULong","PULat"),4326);
  - 105.8 hours!
- Example 2:
  - Finding the nearest tax blocks for 170 million taxi pickup locations (to aggregate based on tax block types)
  - Using open source libspatiaindex+GDAL (to avoid database overhead)
  - 30.5 hours!

Can we get interactive responses?

Background and Motivation

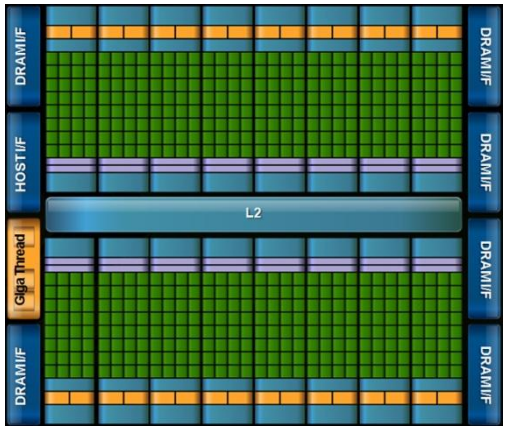


Multicore CPUs



Cloud computing+MapReduce+Hadoop

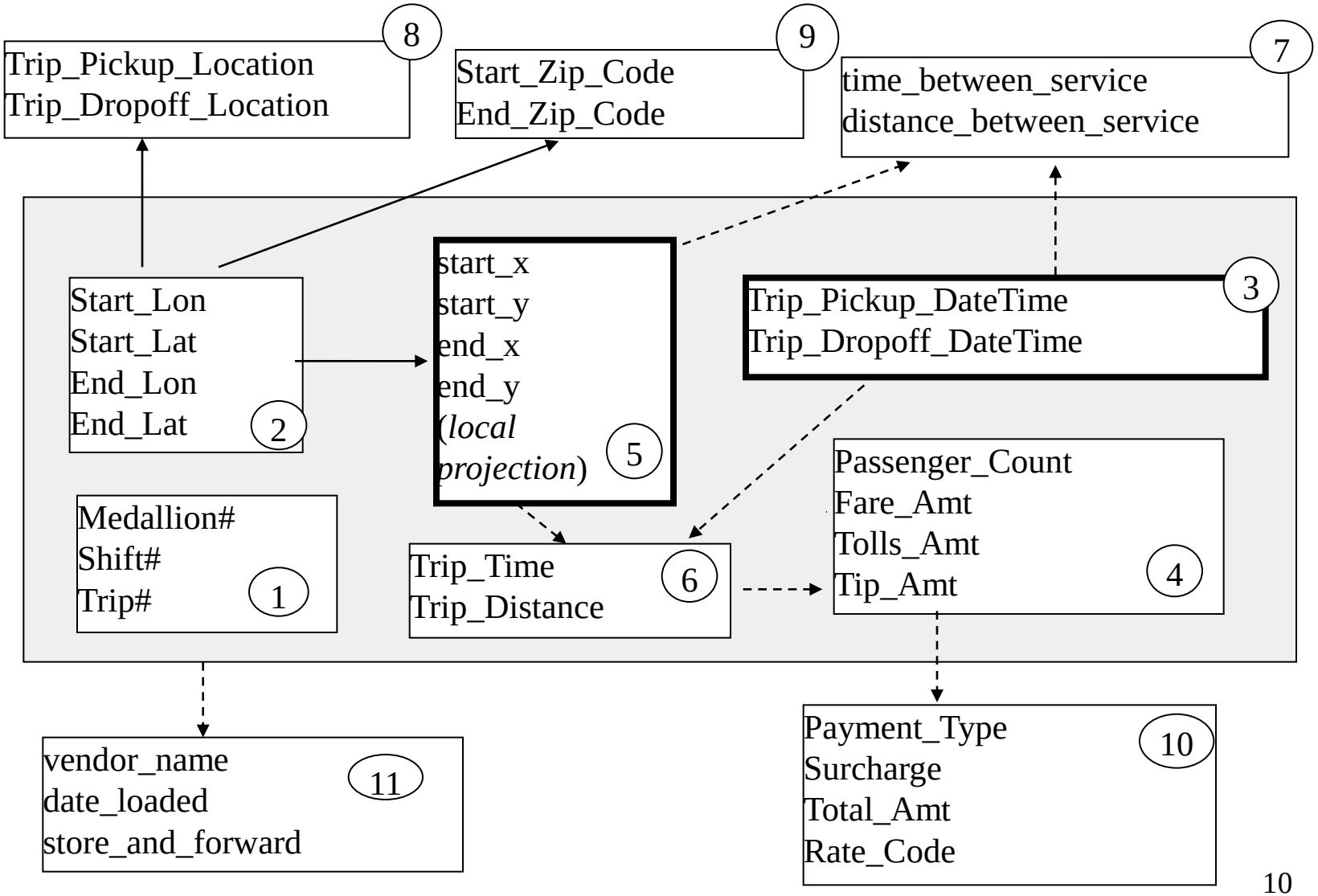
GPGPU Computing: From Fermi to Kepler



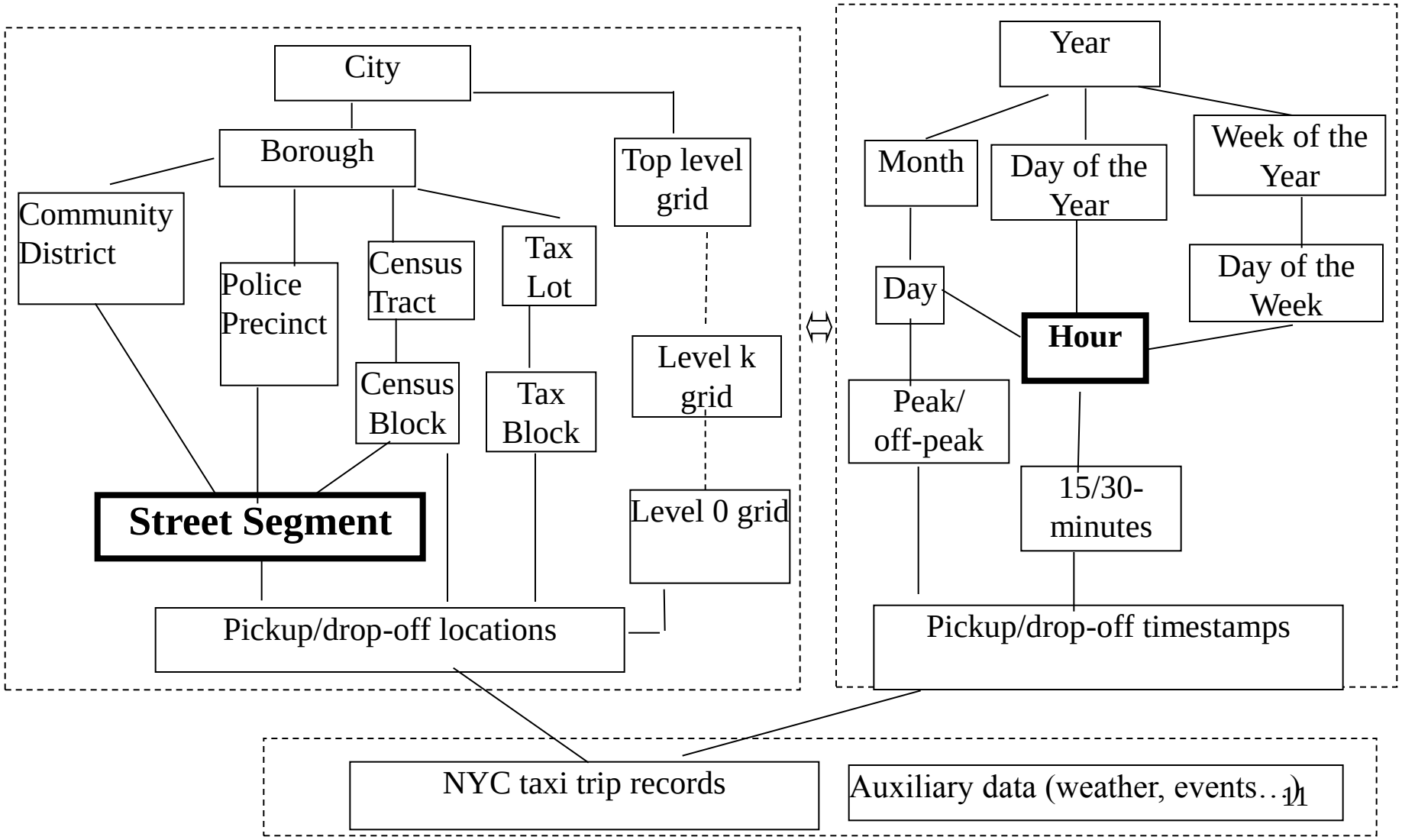
Background and Motivation

Feature	Intel Xeon E7-8870	Nvidia Tesla K10
Price	$4,61,6	$2,500
Processing Cores	10	3,072 (in 15 multiprocessors)
Hardware threads	10*2	15*2048
Frequency	2400 MHZ	745 MHZ
L1/L2/L3 cache	(32k+32K)/256K/30M per core	48K per SM
RAM	variable	8GB
Memory Bandwidth	25.6 GB/s	320 GB/s
Number of Transistors	2.6 Billion	7.0 Billion
Power Consumption	130 W	225 W

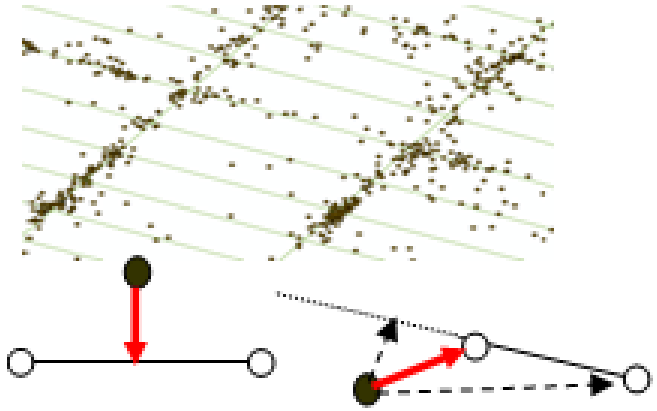
Aggregations on Taxi Trip Records



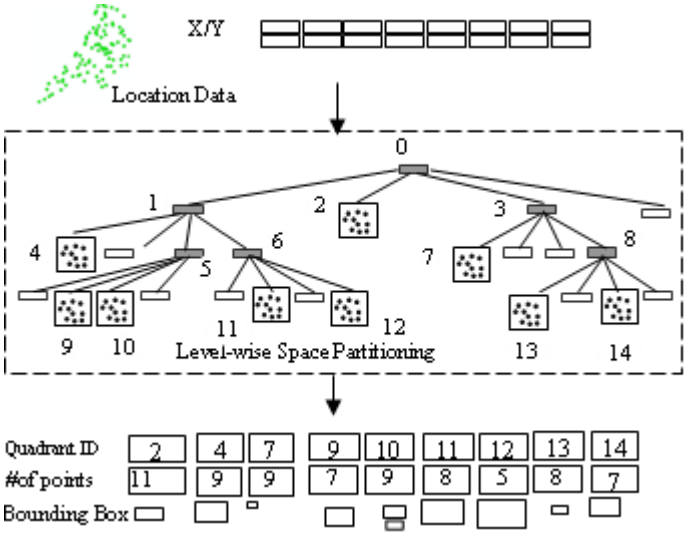
Aggregations on Taxi Trip Records



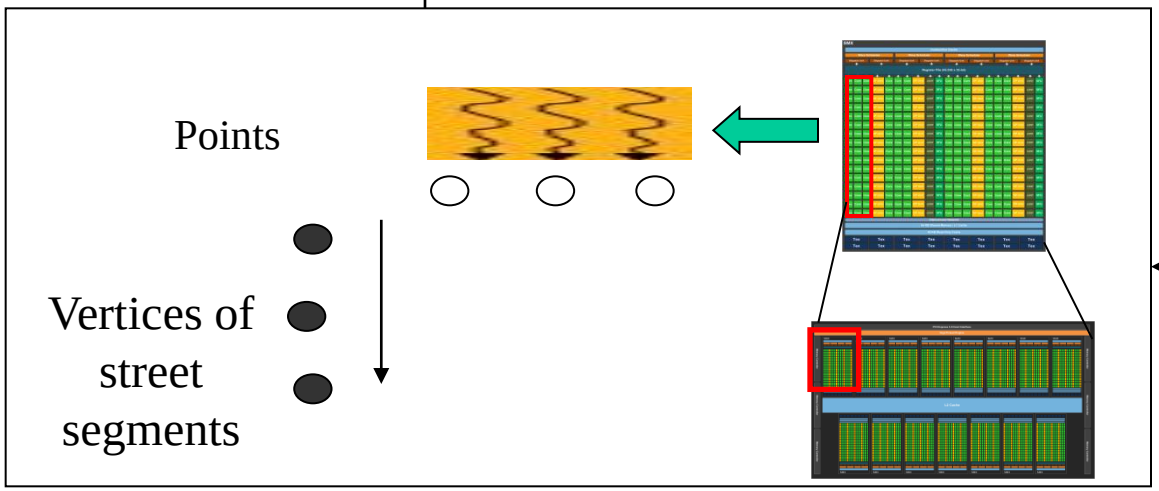
Implementation Details



Mapping a point to its nearest street segment



Grouping points into quadrants



Single-Level Grid-File based Spatial Filtering on GPUs

Implementation Details

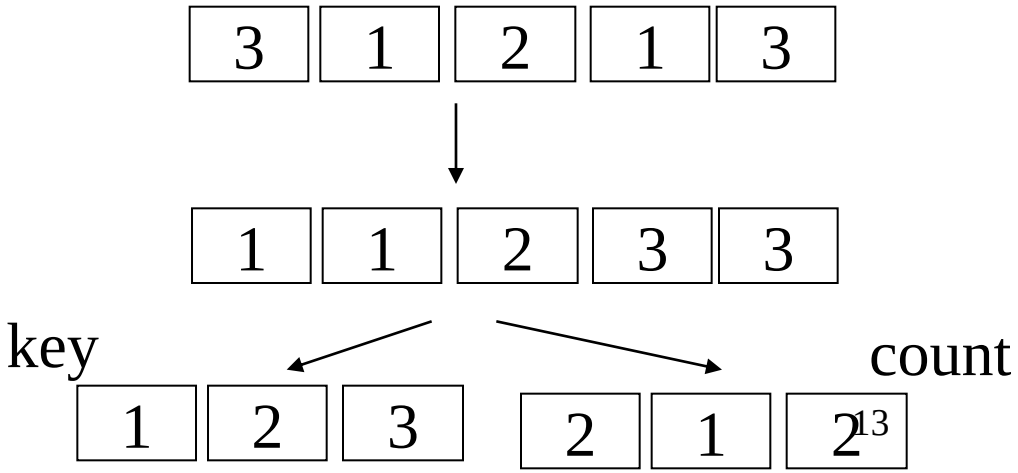
Parallel Counting on GPUs using parallel primitives

Transform to generate keys (spatial entity identifiers, temporal units or their combinations)

Sort

Reduce

```
struct make_key
{
    __host__ __device__
  uint operator()(thrust::tuple<uint, uint> v)
  {
      uint segid=(thrust::get(0)(v))  &0x07FFFFF
      uint hour  =(thrust::get(1)(v)>>12)&0x0000001F;
     return ((segid<<5)|hour);
  }
};
```



Experiment and Results

- Data
  - Taxi trip records: 300 million in two years (2008-2010), **~170 million in 2009**
  - NYC DCPLION street network data: 147,011 street segments
- Hardware
  - Dell T5400 Dual Quadcore CPUs with 16 GB memory
  - Nvidia Quadro 6000 with 448 cores and 6 GB memory

Experiment and Results

Table 1 Results on Spatial Associations on GPUs

# of Months	1	2	3	4	6	9	12
N1 (*10^6)	13.84	27.00	41.17	55.23	83.81	124.64	168.38
N2 (*10^6)	0.155	0.306	0.496	0.676	0.982	1.358	1.747
t1 (second)	0.955	1.876	2.908	3.915	5.986	9.001	12.233
t2 (second)	2.059	1.615	1.472	1.495	1.123	1.176	1.221
t3(second)	0.200	0.343	0.519	0.677	0.941	1.270	1.601
T=t1+t2+t3	**3.214**	**3.834**	**4.899**	**6.087**	**8.050**	**11.447**	**15.055**

N1- # of point locations; N2- # of point quadrants

t1: time to generate point quadrants
t2:time to filter bounding boxes (point quadrants/street segments)
t3: time to compute distances and assign identifiers

Experiment and Results

Table 1. Performance comparison on spatial association

	GPU-Time	CPU-Time	Speedup
t1 (s)	12.233	162.004	13X
t2 (s)	1.221	/	
t3(s)	1.601	35.338	
T=t1+t2+t3(s)	15.055	197.342	22X

t1: time to generate point quadrants
t2:time to filter bounding boxes (point quadrants/street segments)
t3: time to compute distances and assign identifiers

Experiment and Results

Table 2. Experiment Results for Different Aggregations on Multi-Core CPUs (in Seconds)

Aggregation	Serial	1T	2T	4T	8T	16T
1 Pickup Segment (spatial)	12.519	19.776	9.768	4.992	2.513	1.721
2 Pickup Hour (temporal)	7.043	6.089	4.347	2.121	1.186	0.907
3 Pickup Segment+Hour (Spatiotemporal)	17.128	24.238	12.522	6.707	3.803	3.781

Experiment and Results

Performance comparison on counting

Aggregation	CPU-Serial	CPU-Best	GPU	CPU-Serial /GPU	CPU-Best /GPU
Spatial	12.519	1.721	0.188	**66.6**	**9.2**
Temporal	7.043	0.907	0.257	**27.4**	**3.5**
Spatiotemporal	17.128	3.781	0.274	**62.5**	**13.8**

Conclusion and Future Work

- We report our designs, implementations and experiments on spatial, temporal and spatiotemporal aggregations of hundreds of millions of taxi trip records in an OLAP setting
- By utilizing the massively data parallel GPU processing power, we were able to spatially associate nearly 170 million taxi pickup location points with their nearest street segments among 147,011 candidates in about 15 seconds and achieved 13X speedup over optimized serial CPU implementation.
- Spatial, temporal and spatiotemporal aggregations can be processed in the order of a fraction of a second on GPUs.
- The experiment results support the feasibility of building a high-performance OLAP system for processing large-scale taxi trip data for real-time, interactive data explorations on GPUs.

Conclusion and Future Work

- To scale up, we would like to further reduce the processing times for both spatial association and counting.
- To investigate the appropriate spatial and temporal scales for interactive OLAP processing
- To scale-out, we plan to explore cluster computing technologies to process larger scale data, e.g. multi-year and multi-city.

Implementation Details

- Example query:

Count the number of taxi pickup locations at each of the street segments at each of the 24 hours

=> Need spatiotemporal aggregation

Procedure:

Step 1: perform spatial association to find out the corresponding street segment ids of each taxi pickup location => the vector PU_seg: each entry in the vector is the street segment id for a taxi pickup location (the number of vector entries is equal to the number of taxi pickup records)

Step 2: input vector PU_seg and vector PU_t that stores the taxi pickup hours for the corresponding taxi pickup location to the counting module and run the counting module to find the answer for the query