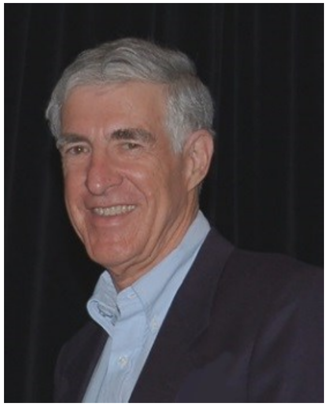
UNIVERSITY OF MARYLAND

SPDO: High-Throughput Road Distance Computations on Spark Using Distance Oracles





Shangfu Peng | **Jagan Sankaranarayanan** | **Hanan Samet**

shangfu@cs.umd.edu
University of Maryland
Spatial Tek LLC

jagan@nec-labs.com
NEC Labs America

hjs@cs.umd.edu
University of Maryland
Spatial Tek LLC

ICDE 2016

Motivation: Need to compute millions of network distances or trip times per second on a road network

Who needs such computations?

Large Package Delivery Companies

Small Food Delivery Companies

e-commerce companies

Trucking Companies

Delivery companies compute Origin-Destination (OD) matrices that can quickly require million distance computations

1,000 locations

1,000 locations

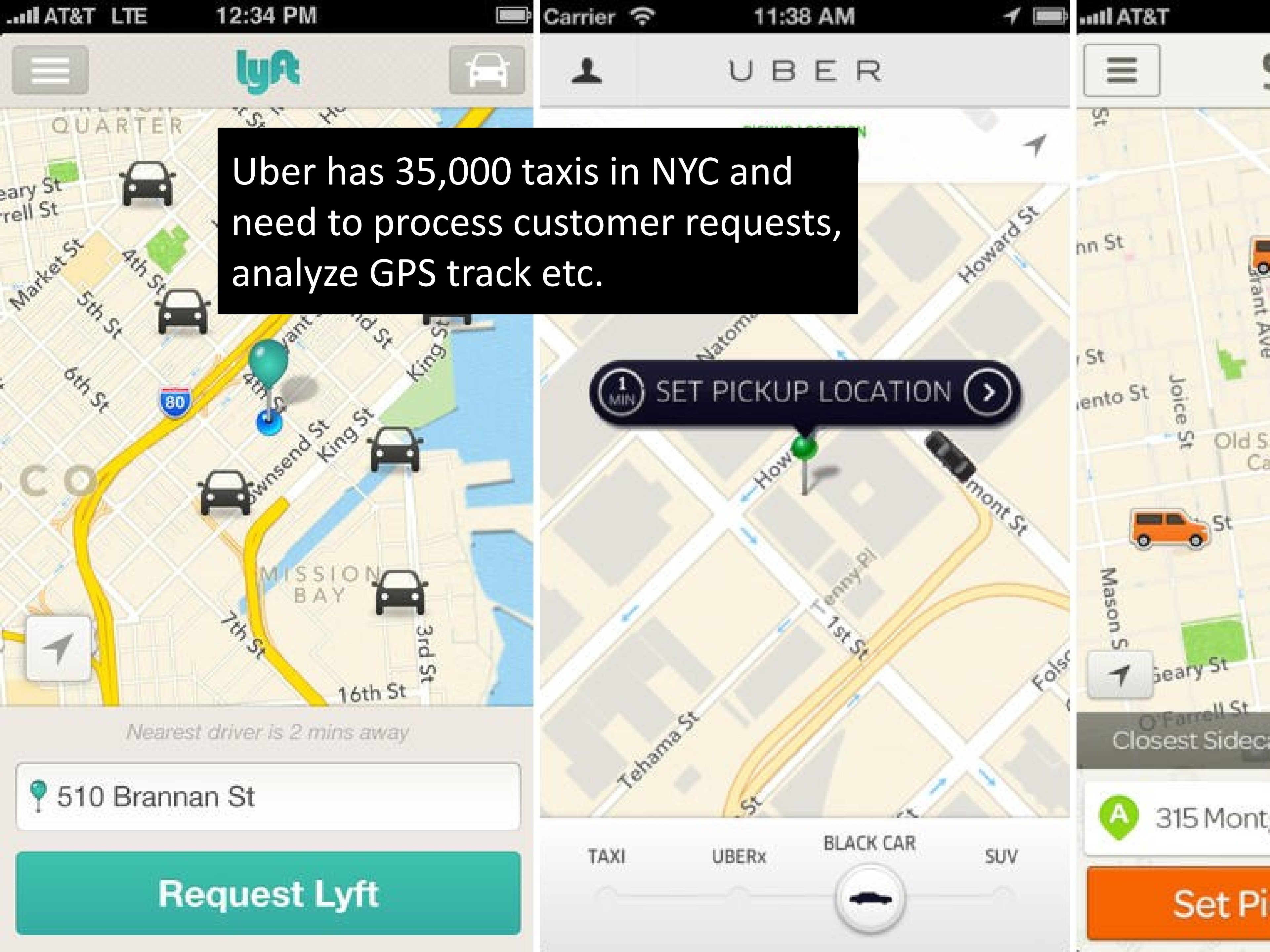
DISTANCES BETWEEN THE MAGGIE'S CENTRES IN MILES

Maggie's Centres	Highlands	Aberdeen	Dundee	Fife	Forth Valley	Edinburgh	Lanarkshire	Glasgow	Newcastle	at the Christie	Merseyside	Nottingham	Cheltenham	Wallace	Oxford	West London	Swansea
Highlands	0	102	132	148	150	151	162	169	277	374	385	432	489	511	519	565	585
Aberdeen	102	0	70	95	124	124	141	148	250	352	363	410	468	489	497	544	563
Dundee	132	70	0	33	56	57	73	80	182	284	296	343	400	422	430	476	496
Fife	148	95	33	0	34	25	50	57	151	259	270	317	374	396	404	450	470
Forth Valley	150	124	56	34	0	30	17	24	136	228	240	287	344	366	374	408	440
Edinburgh	151	124	57	25	30	0	36	48	120	227	238	278	343	357	372	419	439
Lanarkshire	162	141	73	50	17	36	0	14	146	212	224	271	328	350	358	404	424
Glasgow	169	148	80	57	24	48	14	0	155	221	232	279	337	358	366	412	432
Newcastle	277	250	182	151	136	120	146	155	0	159	195	161	265	240	270	283	361
at the Christie	374	352	284	259	228	227	212	221	159	0	43	99	128	181	158	204	224
Merseyside	385	363	296	270	240	238	224	232	195	43	0	117	146	199	176	222	162
Nottingham	432	410	343	317	287	278	271	279	161	99	117	0	108	96	117	132	203
Cheltenham	489	468	400	374	344	343	328	337	265	128	146	108	0	121	42	104	108
Wallace	511	489	422	396	366	357	350	358	240	181	199	96	121	0	96	60	238
Oxford	519	497	430	404	374	372	358	366	270	158	176	117	42	96	0	53	159
West London	565	544	476	450	408	419	404	412	283	204	222	132	104	60	53	0	189
Swansea	585	563	496	470	440	439	424	432	361	224	162	203	108	238	159	189	0

* Please note these distances are based on the most direct driving route and distances will vary depending on mode of travel and route chosen.
** If you're feeling adventurous you could also visit Maggie's Hong Kong!

1 Million Network Distance Computations

Uber has 35,000 taxis in NYC and need to process customer requests, analyze GPS track etc.

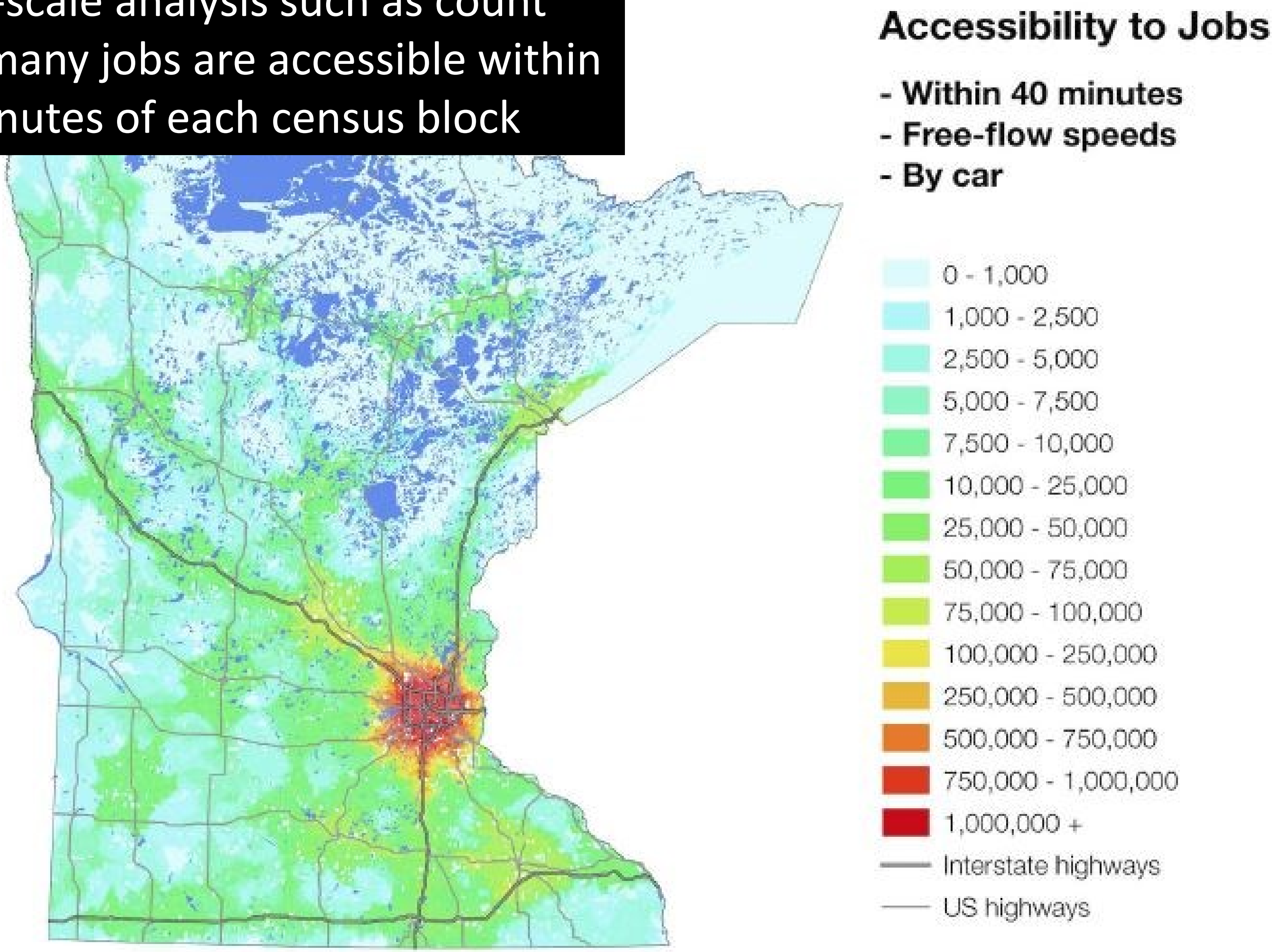


Local Advertisement companies need to serve ads to people that are both proximal and can reach the customer business, need to do that at say 10K impressions a second!

xAd

PlaceIQ

Large-scale analysis such as count how many jobs are accessible within 40 minutes of each census block

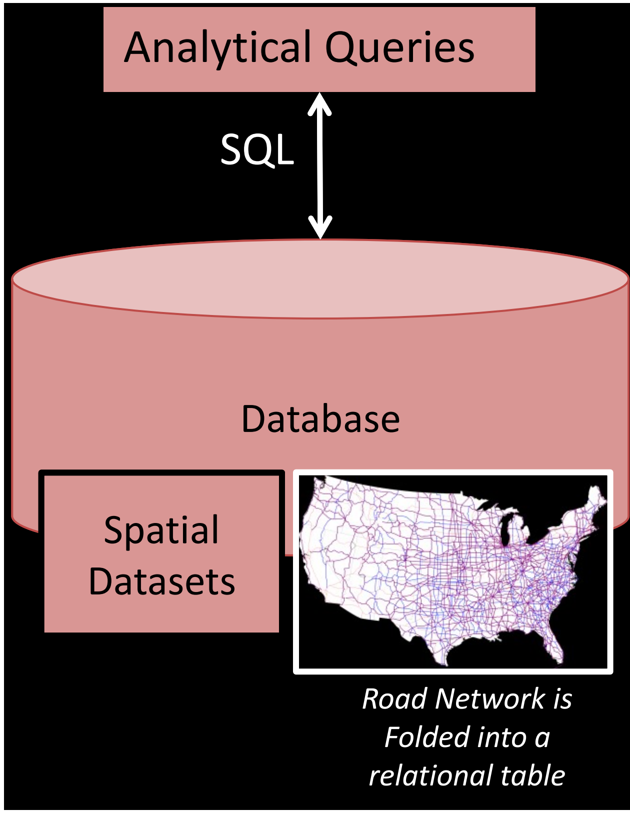


State Smart Transportation Initiative (SSTI) Example*

**source: http://www.ssti.us/Events/accessibility-towards-a-new-multimodal-system-performance-metric/*

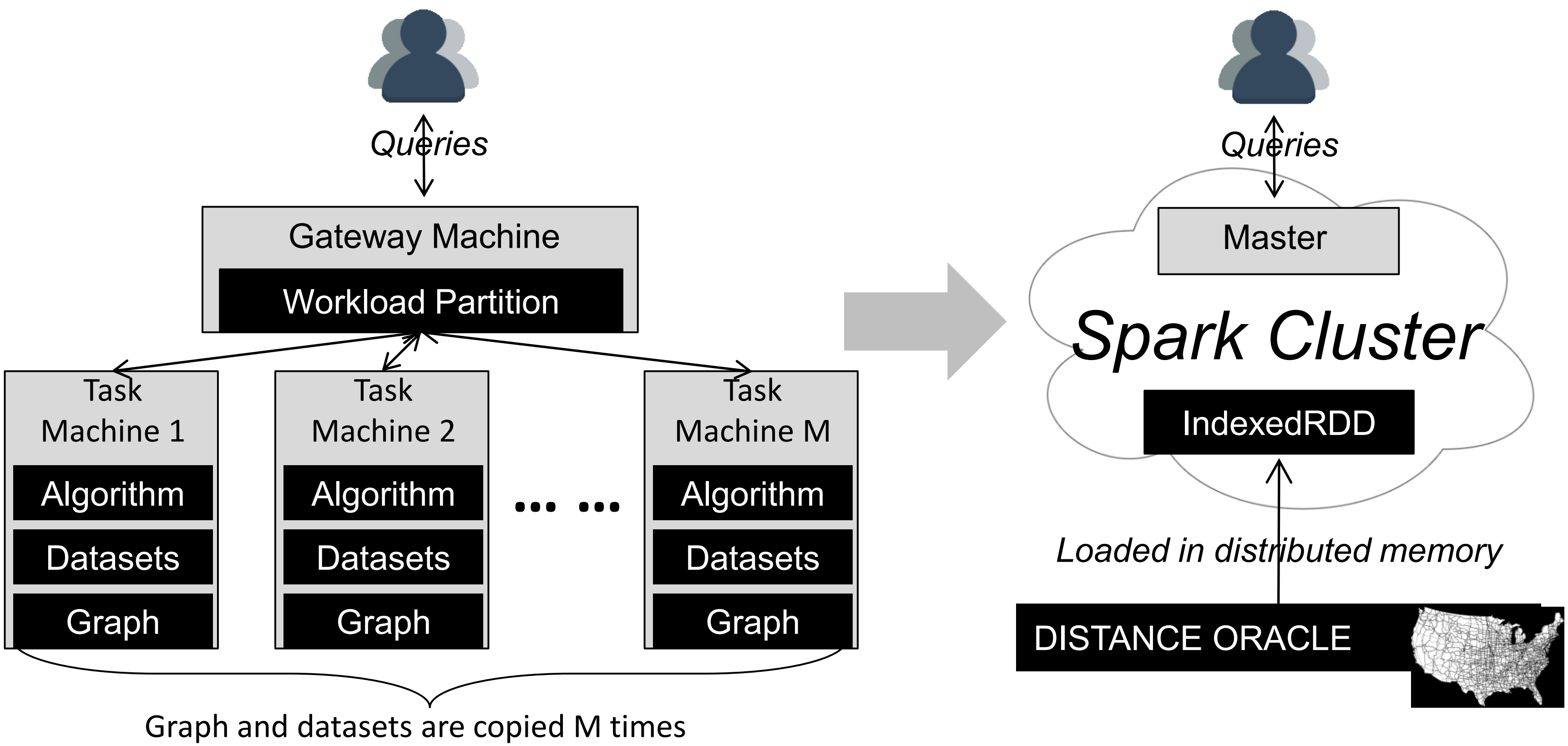
Motivation – lookup based

- Scan based methods is popular, but not good enough
  - Contraction Hierarchies(CH)
  - Transit Node Routing (TNR)
  - Customizable Route Planning (CRP)
- Lookup based methods, built for a relational database system
  - Naive way, precompute all pairs result
    - 24M vertices in USA road network
    - At least 6286TB for storage
  - Hub Labeling → HLDB (Microsoft)
  - **ϵ-Distance Oracle**
  - SILC
- Previous research has shown that lookup based methods achieve a much higher throughput performance



Motivation – distributed key-value store

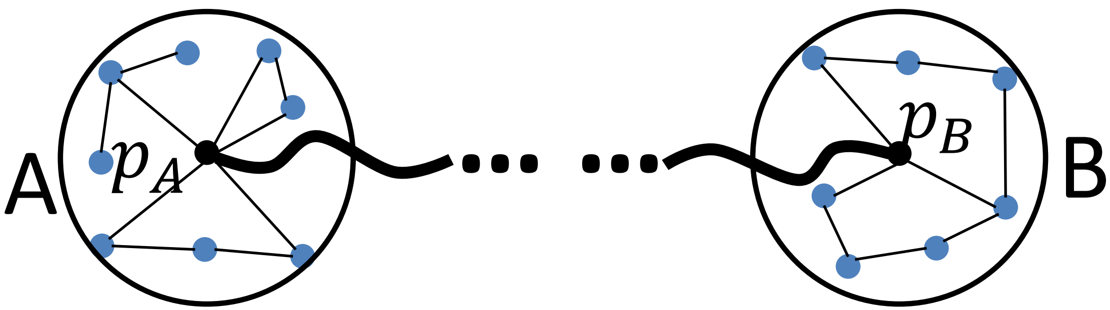
- ❑ High throughput → Distributed system
- ❑ Lookup based methods → Key-value data structure → hash access
- ❑ We need to adapt distance oracles for distributed architecture
  - Test on RDBMS shows that we cannot scale more than 60K queries/per core
  - Need to scale to even higher throughputs, using Apache Spark

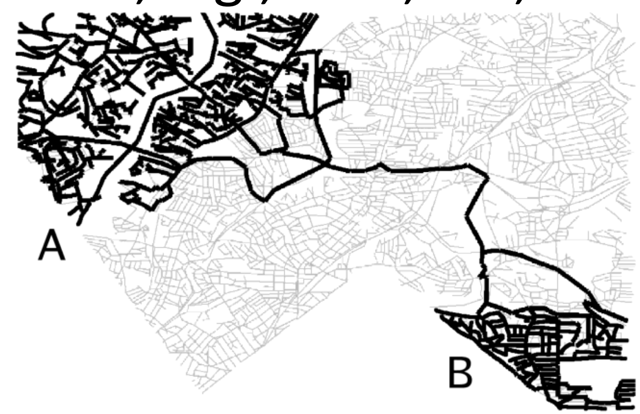


(a) Existing distributed solution **(b) SPDO**

Method - ϵ-Distance Oracle (ϵ-DO)

- A compression representation of road network distances
- Well-separated pair decomposition (WSPD) on a road network
- Can represent the distance between any two points in A and B by one value with an epsilon error tolerance, e.g., 0.25, 0.1, 0.05



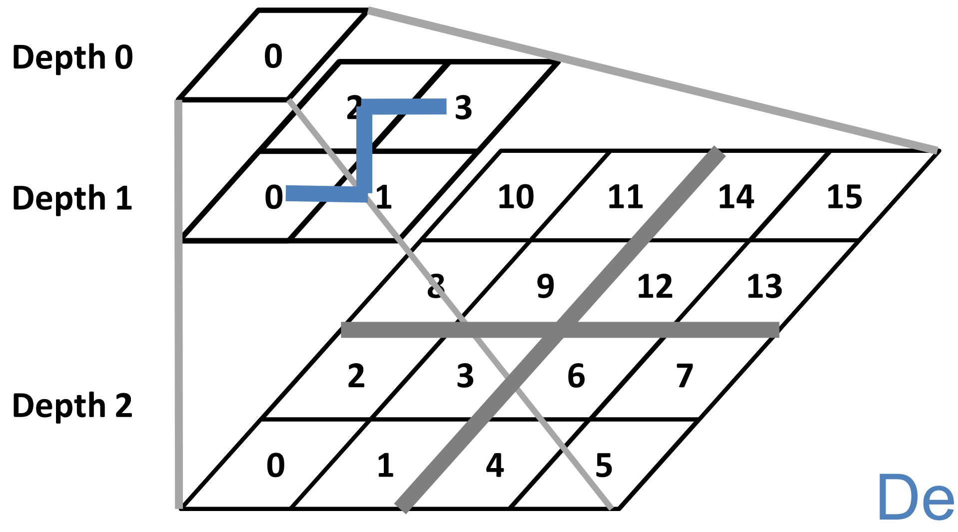


dumbbell structure

- $O\left(\frac{n}{\epsilon^2}\right)$ well-separated pairs, each well-separated pair can be represented as a key-value pair
  - Key is the pair of two vertex sets, A and B
  - Value is $d_G(p_A, p_B)$

Method - DO-Tree

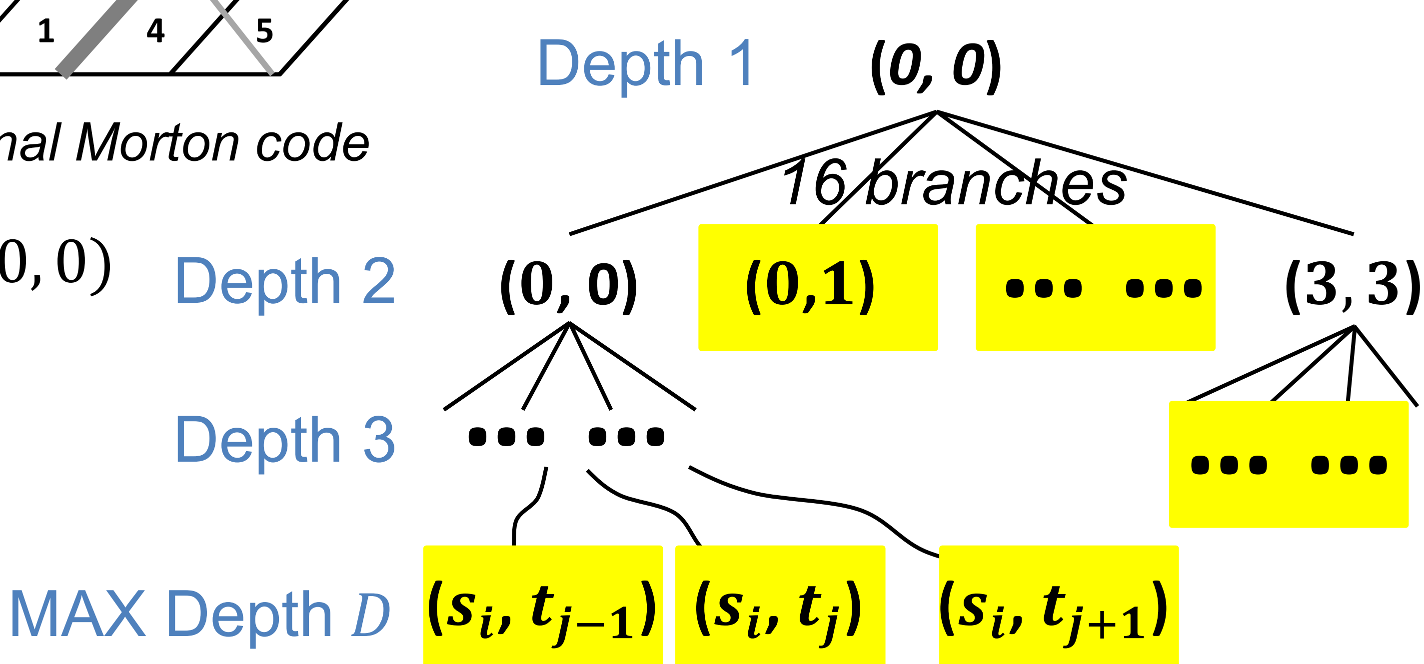
- ❑ Top-down decomposition on the whole space, (S, S)
- ❑ PR-Quadtree, Morton code representation



2-dimensional Morton code

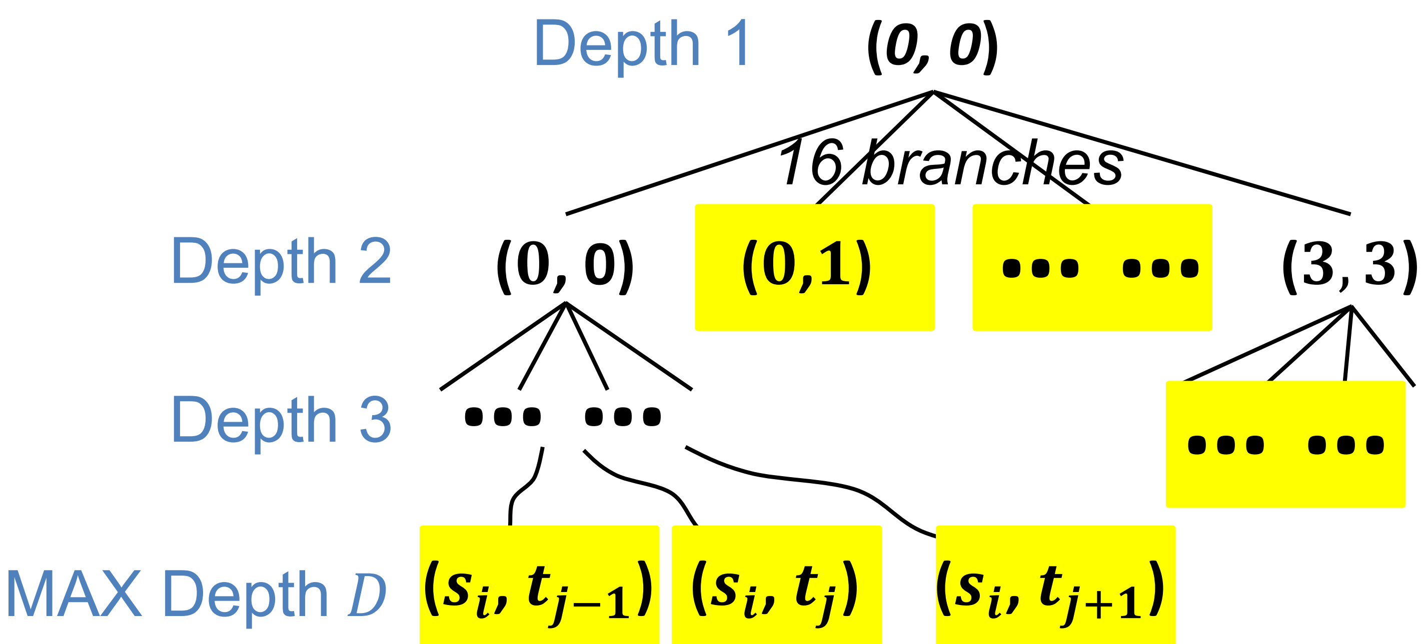
- ❑ (S, S) => $(0, 0)$

4-dimensional DO-tree



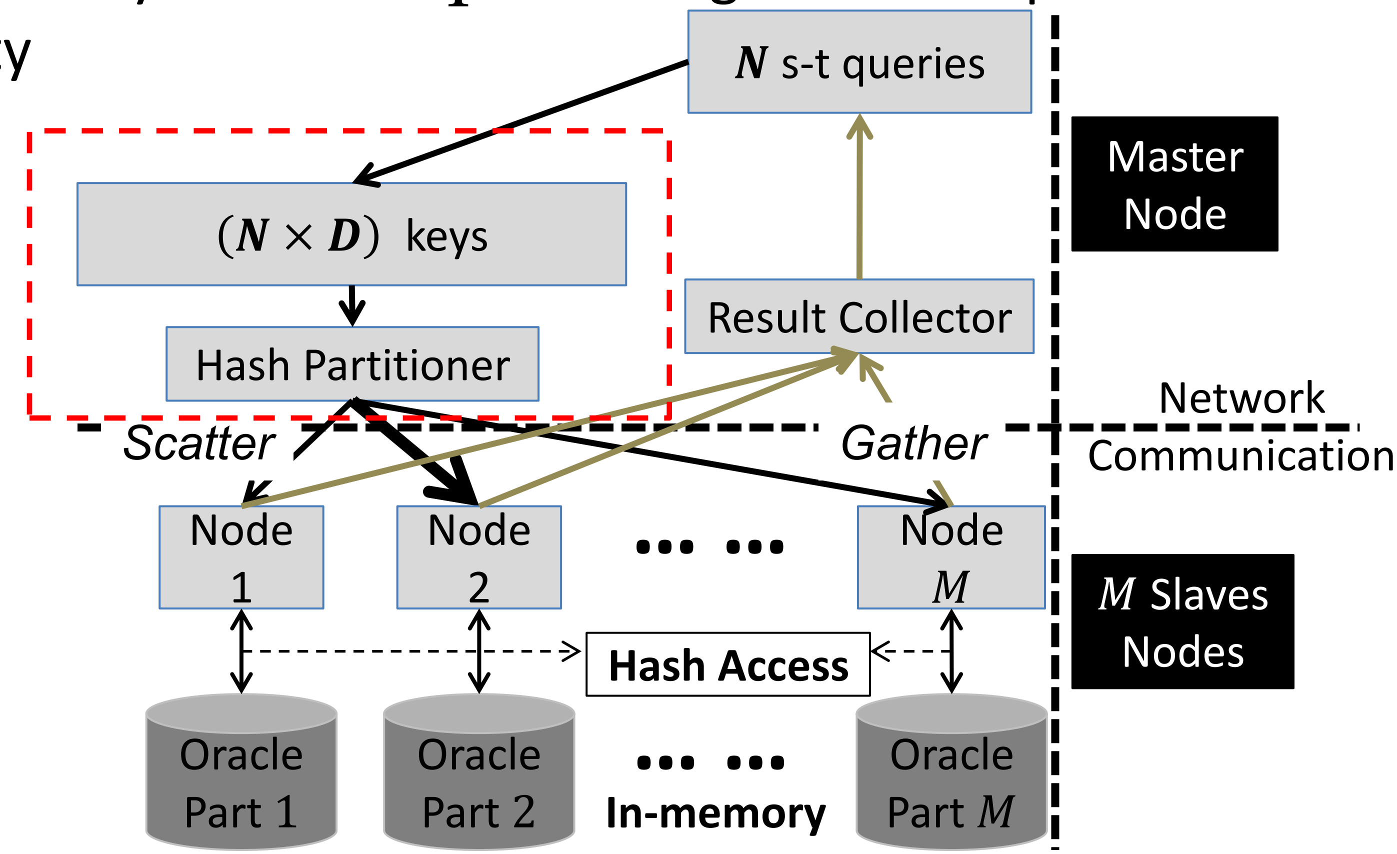
Method - DO-Tree

- DO-tree = the top-down WSPD
- Each leaf node of DO-tree is a well-separated pair
- **Uniqueness Property**: For any source-target query (s, t), there is exactly one leaf node of the DO-tree, i.e., WSP that contains both s and t.
- Task: for any query (s, t), finding the exact one WSP that contains (s, t) ***by hash access***, then returning the distance result



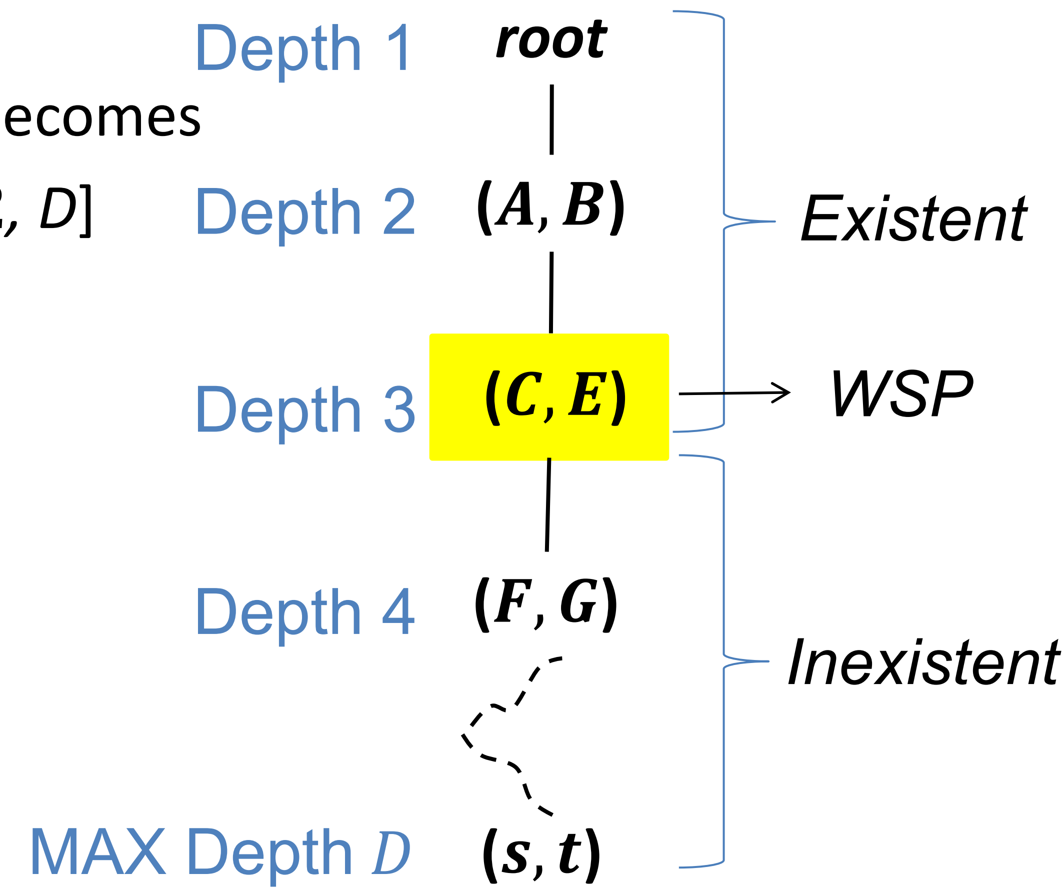
Method - Hash Access for ϵ-DO

- Basic idea (Basic)
  - Hash Table $\boldsymbol{H_1}$ = all leaf nodes (WSPs) of DO-tree
  - Each query (s, t) generates $O(D)$ keys, each key corresponds one ancestor node of (s, t)
  - Exact one key exists in $\boldsymbol{H_1}$ according to the uniqueness property



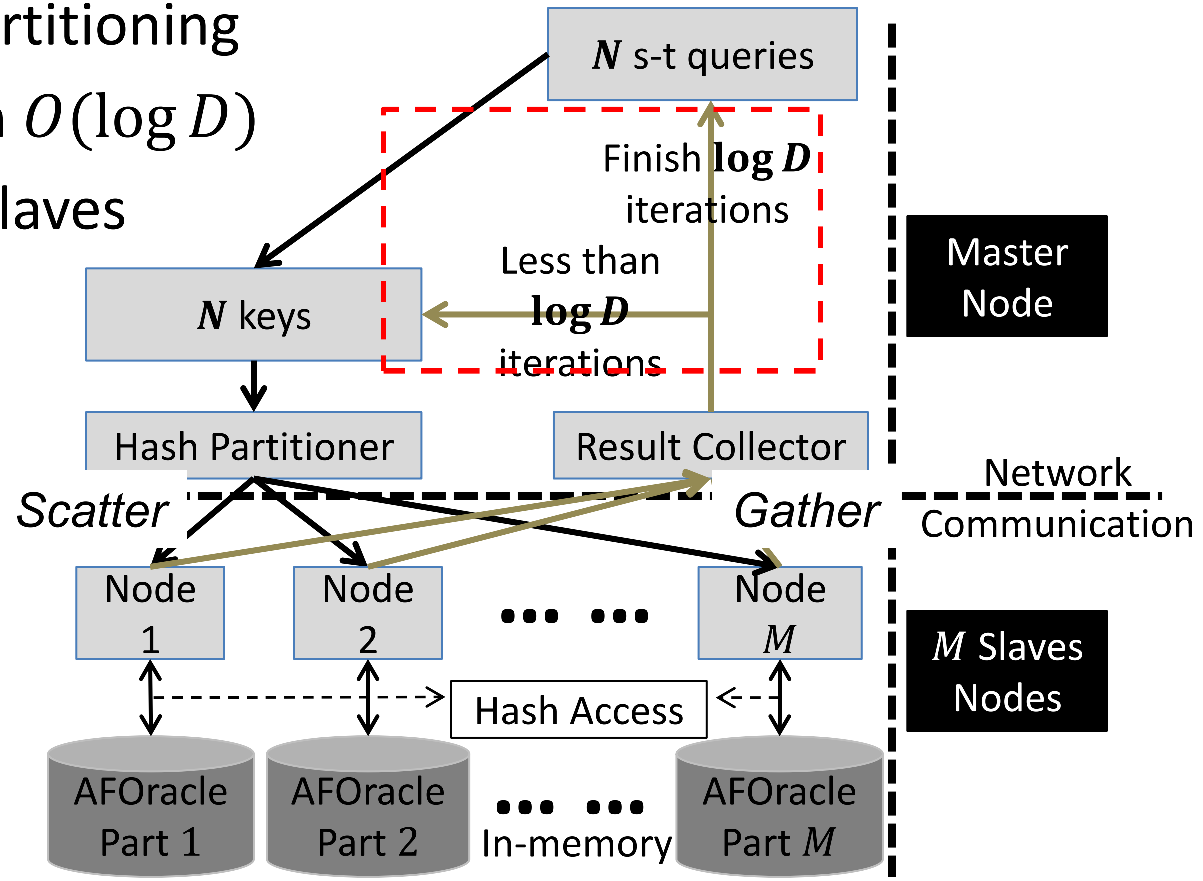
Method - Hash Access for ϵ-DO

- Binary search method (BS)
  - Hash Table $\boldsymbol{H_2}$ = all nodes of DO-tree, $\{root, (A, B), (C, E)\}$
  - For any query (s, t), initial possible depth range is [*1*, *D*]
  - Examining if the ancestor node of (s, t) at depth $D/2$ exists in $\boldsymbol{H_2}$ or not
  - Possible depth range becomes either $[0, D/2)$ or $[D/2, D]$



Method - Hash Access for ϵ-DO

- Binary search method (BS)
  - Hash Table $\boldsymbol{H_2}$ = all nodes of DO-tree, $\{root, (A, B), (C, E)\}$
  - Note that $O(\log D) = O(\log \log n)$
- Wise Partitioning method (WP)
  - Adding spatial partitioning of all nodes, push $O(\log D)$ hash lookups in Slaves



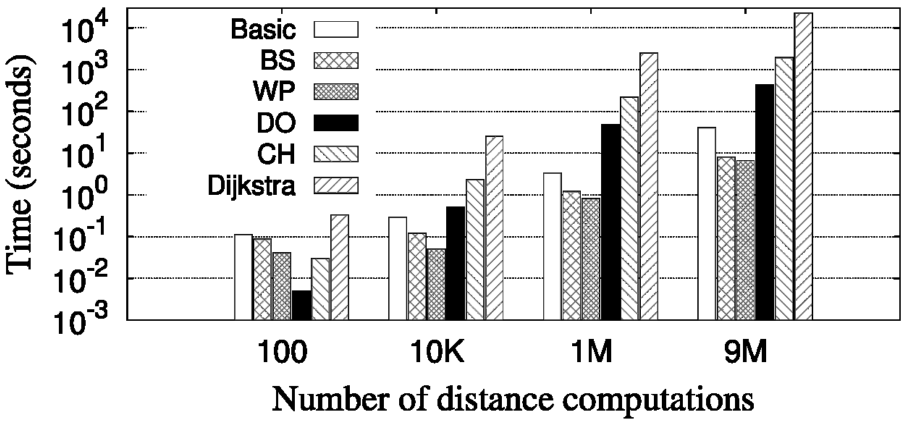
Evaluation – time complexity

❑ Time complexity analysis
- Master, Slave, and network communications
- N source-target queries, M slaves, D depths of DO-tree

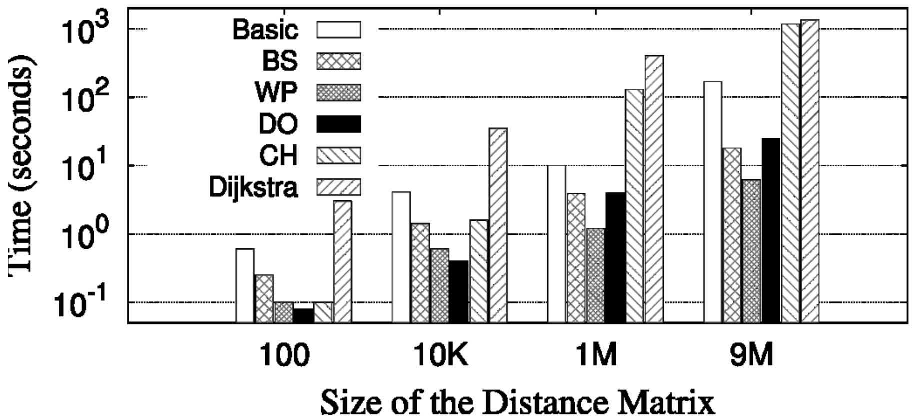
Design	Iteration	Master Time	Slave Time	Network Communication
Basic	1	$O(N \cdot D)$	$O\left(\frac{N \cdot D}{M}\right)$	$O(N \cdot D)$
BS	$\log D$	$O(N \cdot \log D)$	$O\left(\frac{N \cdot \log D}{M}\right)$	$O(N \cdot \log D)$
WP	1	$O(N)$	Random $O\left(\frac{N \cdot \log D}{M}\right)$ Worst $O(N \cdot \log D)$	$O(N)$

Evaluation - throughput

- Comparisons
  - Dijkstra's algorithm (Dijkstra), good for one-to-many pattern
  - Contraction hierarchies (CH)
  - Distance Oracle embedded in PostgreSQL (DO)



- NYC road network
- 264K vertices, 733K edges
- One local server, single-thread



- USA road network
- 24M vertices, 58M edges
- 20 machines cluster, single-thread

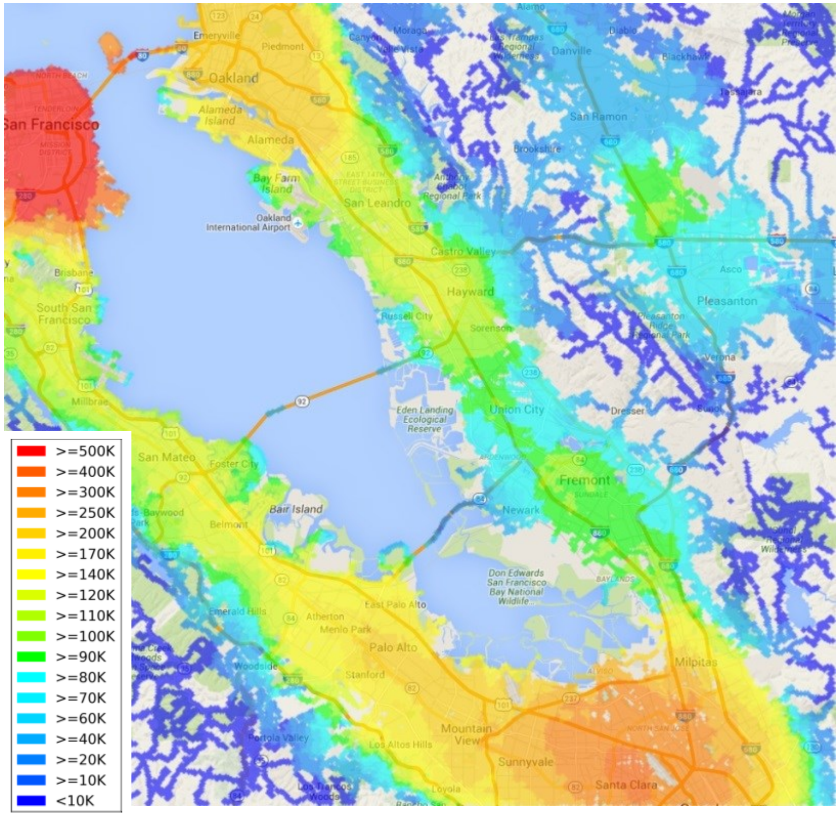
Throughput the USA road network

Method	Basic	BS	WP	DO	CH	Dijkstra
Dist/sec/machine	5.0K	25.0K	73.8K	18.8K	385	1.6

Evaluation - applications

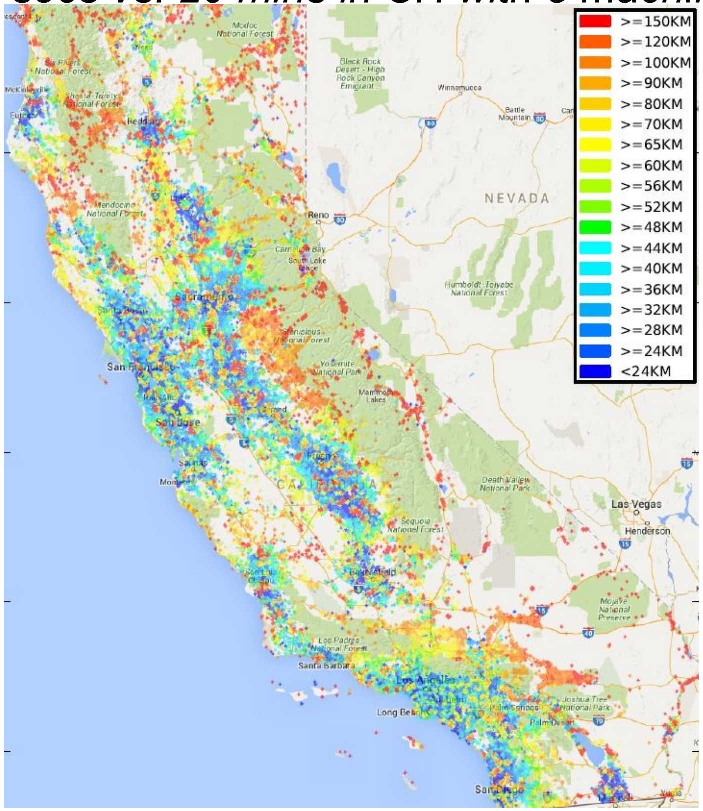
- ❑ Nearby job opportunities, one-to-many pattern
- ❑ Actual drive distance from residence to workplace

One-to-many pattern (2 mins with 1 machine)



Nearby job opportunities (e.g., within 10 kms) for each census block in the Bay Area, requiring 120 million distance computations

One-to-one pattern
(13 secs vs. 20 mins in CH with 5 machines)

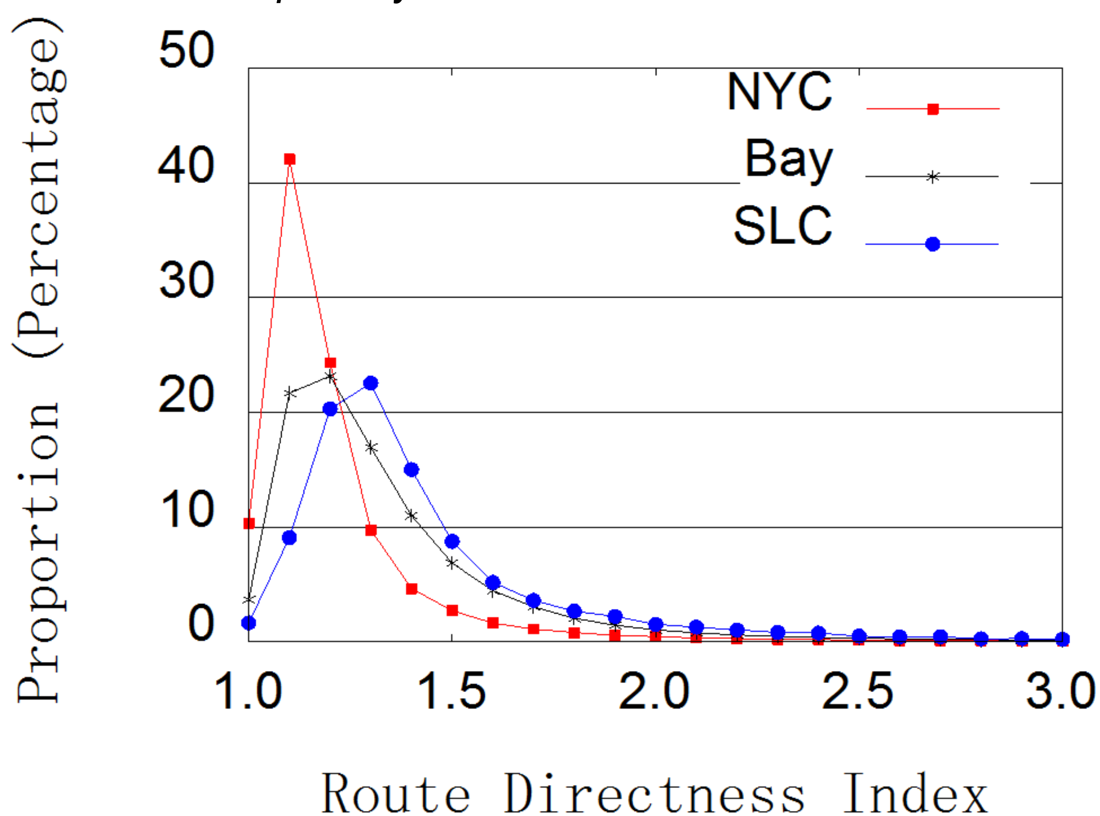


Average drive distance from residence to workplace for California residents, requiring 13.6 million distance computations

Evaluation - applications

- ❑ Route Directness Index (RDI) of two locations
  - ▪ Ratio of the network distance to the Euclidean distance
- ❑ Define Route Directness Spectrum (RDS) of a spatial region
  - ▪ The collection of RDIs between all pairs of vertices in a road network

1 mins per city vs. 4 hours in CH with 1 machine



Conclusions

- **SPDO** is a *high throughput* distributed solution for *shortest road network distance/time* computations using a *distributed key-value store* on Apache Spark.

- Proposed 3 methods on Apache Spark, Basic, BS, and WP, which are a scalable way of obtaining throughput performance that exceeds one million computations per second with just a few machines

- *Reduce running time* for GIS analysts/scientists, *save developing time* for GIS developers, and *simplify the system design* and *reduce hardware cost* for GIS architects

Thanks