Location-Centric View Selection in a Location-Based Feed-Following System

Kaiji Chen
Huawei Technologies

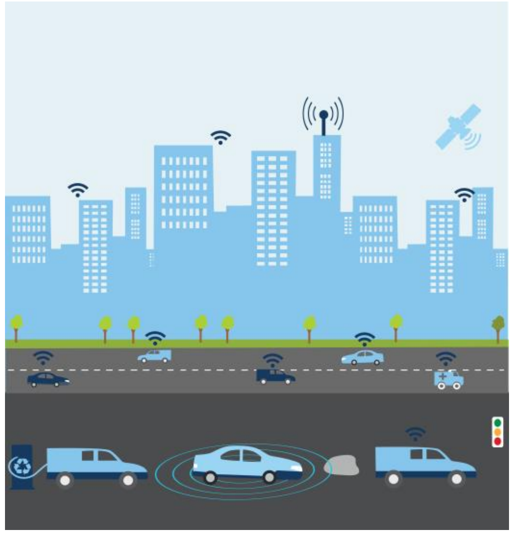
Yongluan Zhou
University of Copenhagen

UNIVERSITY OF COPENHAGEN

Location-Based Feed-Following Systems

- Location-based, augmented-reality mobile game

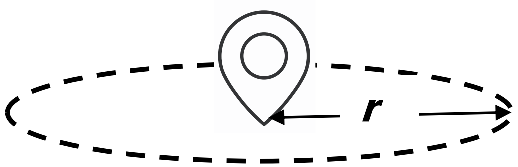
- Smart Vehicle and Vehicle-to-Everything (V2X) communication



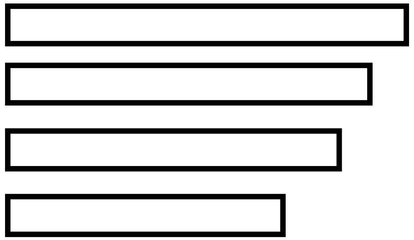
Preliminaries

- Both feeds and users have time-varying locations

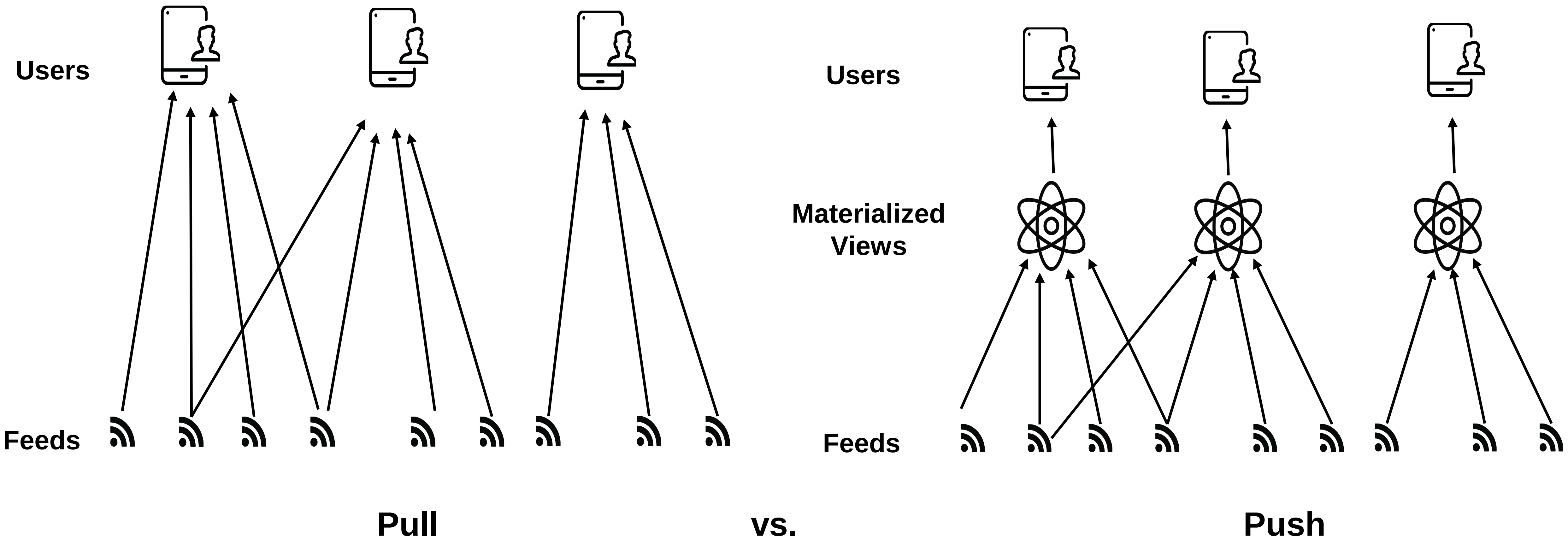
- Each user
  - subscribes to feeds located within a pre-defined proximity: *r*
  - receives updates when online or for every pre-defined time interval



- User query Q consists of
  - A ranking function that ranks the messages
  - Aggregate function across messages from multiple feeds: top-k and diversified top-k



Query Processing



Problem Statement

- Problem:

  Given

  1) a set of moving users $\mathcal{U}$, and
  2) a set of moving feeds $\mathcal{F}$,

  dynamically generate a plan $\mathbb{P}$, consisting of

  1) a set of materialized views $\mathcal{V}$, and
  2) a set of query plans $\mathcal{QP}$, one for each user.

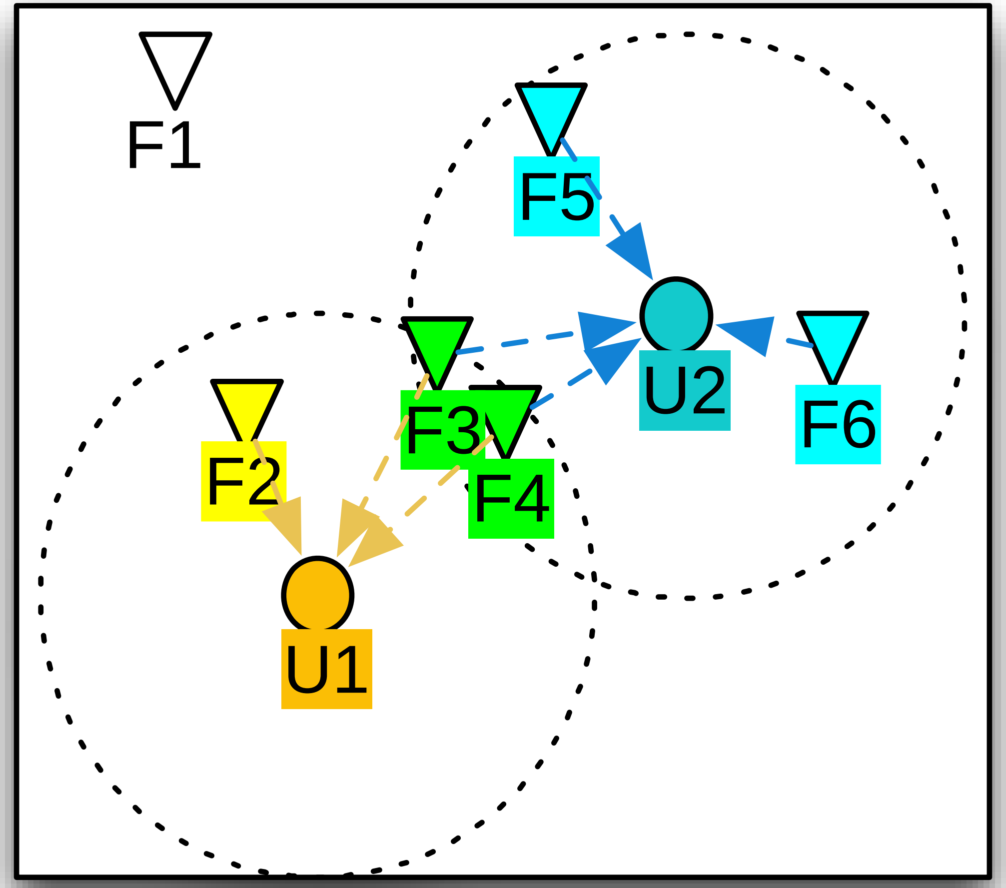
- Cost Model of the plan

$$Cost(\mathbb{P}) = \sum_{v_i \in \mathcal{V}} M(v_i) + \sum_{u_j \in \mathcal{U}} EV(u_j, Vu_j)$$

View Maintenance cost **Query evaluation cost**

Challenges of Moving Users and Feeds

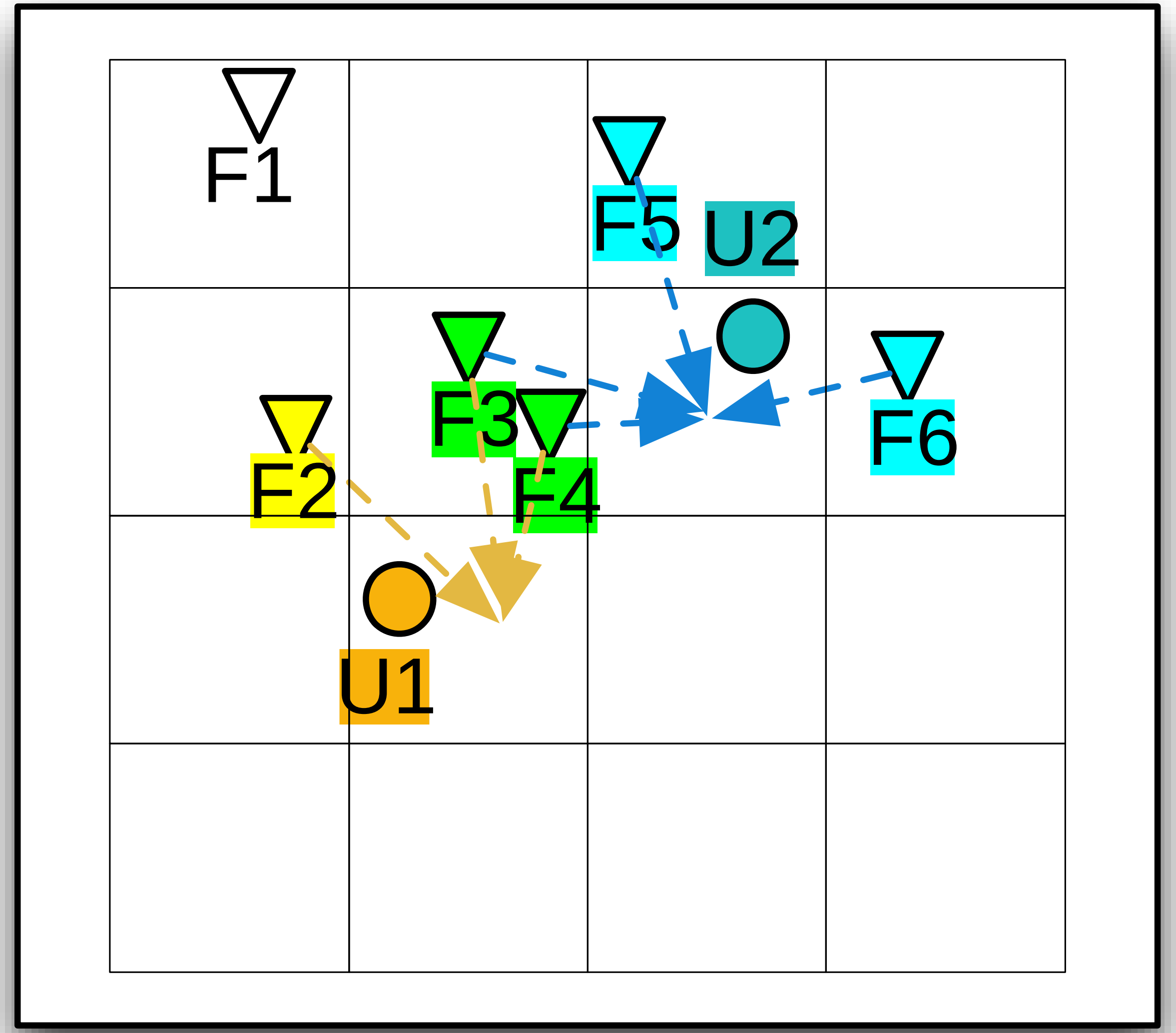
- User-centric paradigm
  - e.g. GeoFeed [1] and Feeding Frenzy [2]



[1] Jie Bao, Mohamed F. Mokbel, Chi-Yin Chow: GeoFeed: A Location Aware News Feed System. ICDE 2012: 54-65
[2] Adam Silberstein, Jeff Terrace, Brian F. Cooper, Raghu Ramakrishnan: Feeding frenzy: selectively materializing users' event feeds. SIGMOD Conference 2010: 831-842

Location-Centric Query Plans

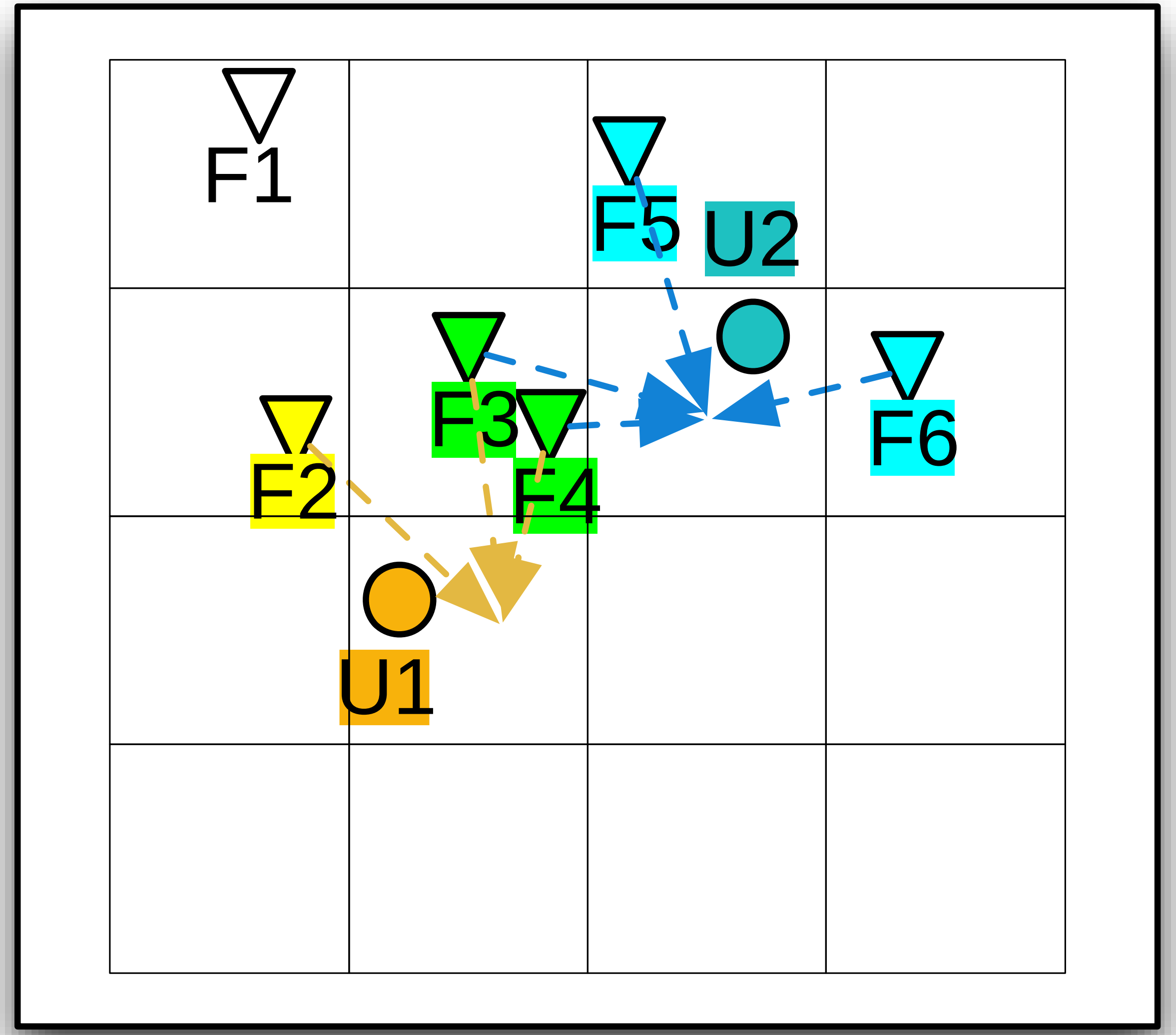
- Grid partitioning
- Generate location-centric plans for each cell
- #query plans = #cells (rather than #users)
- Next step: algorithms to generate and optimize location-centric views and query plans



Grid-Based View Algorithm

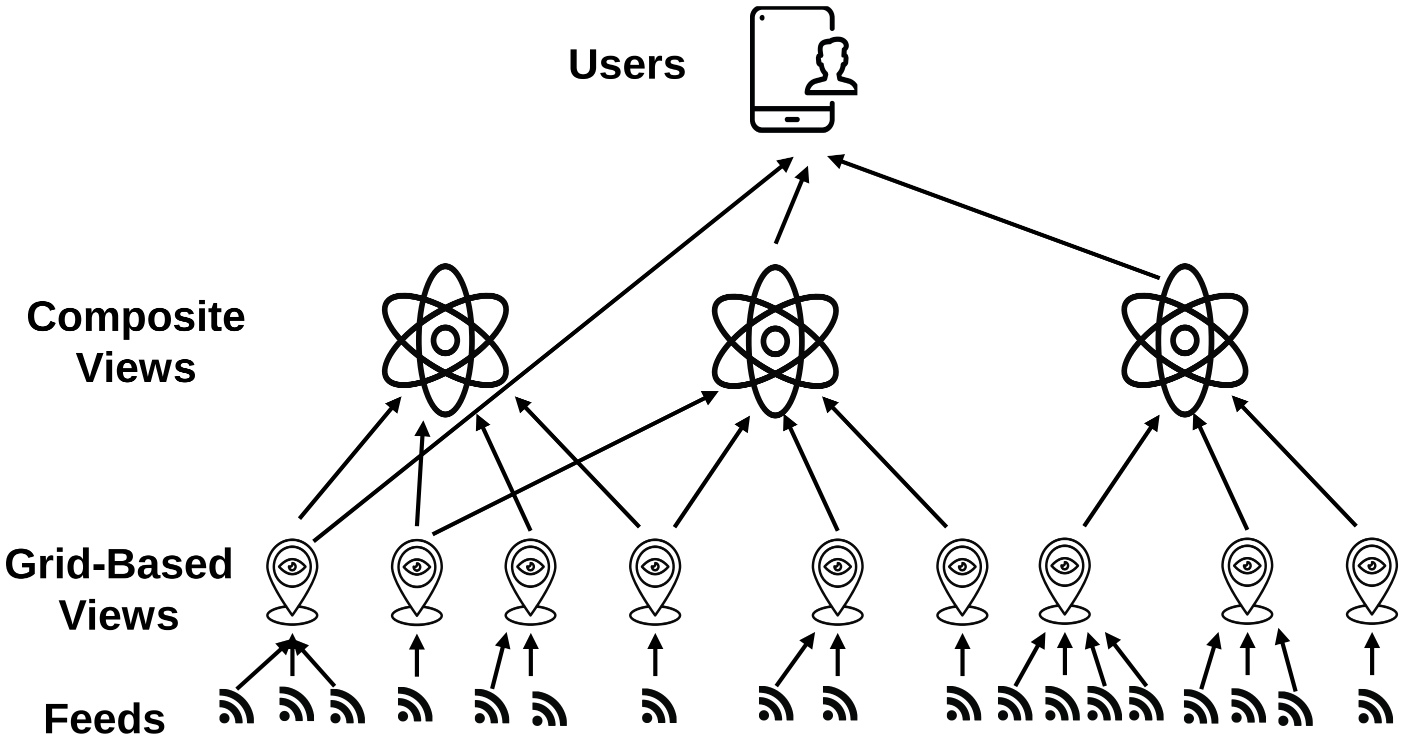
- Assume feeds are not moving at the moment
- Group users according to their query ranges, ranking functions and aggregate functions
- For each user group, do the following

1) For each cell, generate a view over all the feeds located in the cell
2) For each cell, generate a query plan for each user group



Composite-View Algorithm

- Extra maintenance cost, but
- Potentially lower query evaluation cost



Iterative Local Search

1) Start with an initial plan.
2) Iteratively combine two views to form a candidate composite view with the highest benefit.
3) Sort all the composite views in descending order of their benefits;
4) If the benefit is less than a threshold, discard it; otherwise add it to the list;
5) In any case, use minimum set cover algo to generate the query plans.

- The algorithm can be run to re-optimize existing plans.
- Worst-case complexity: $\mathcal{O}(m \cdot n_{max}^2 \cdot |\mathcal{V}| + |\mathcal{V}|^2)$

#cells

query range in #cells

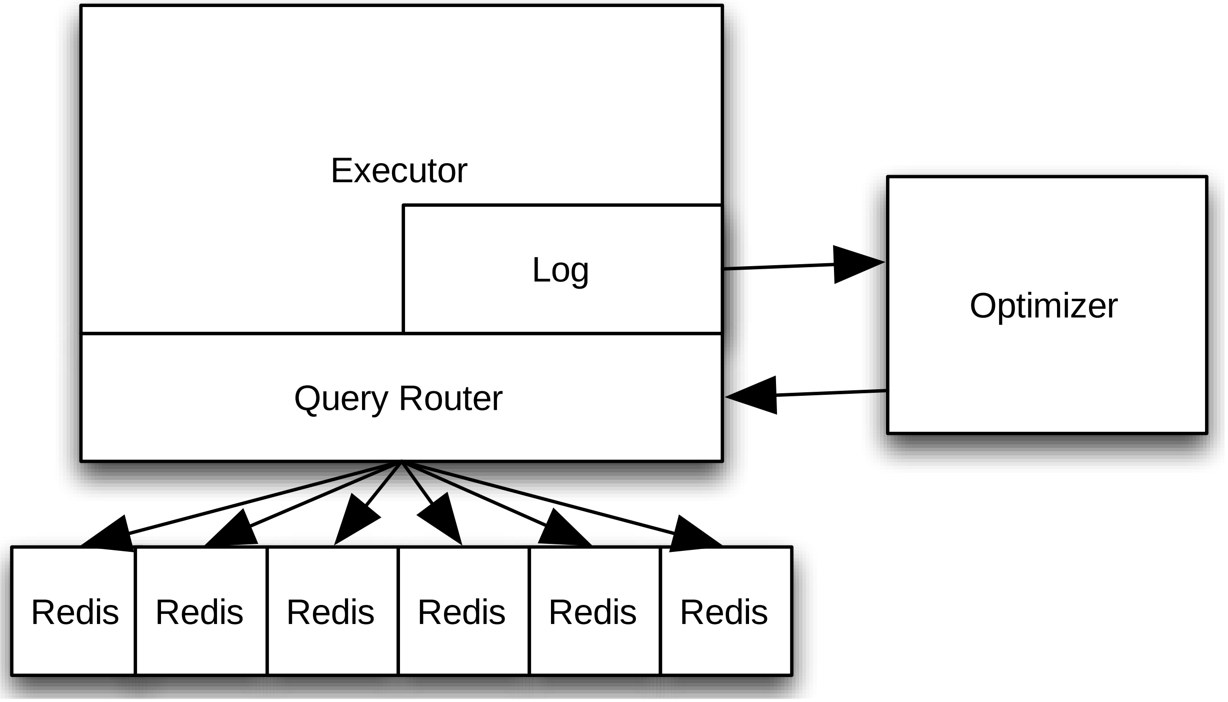
#views

Moving Feeds and Grid Granularity

- Virtual static feeds
  - One for each cell
  - Update messages are assigned to the virtual feeds according to their locations
- Grid granularity
  - trade-off between spatial accuracy and system workload
  - should be determined by the requirement of the applications

Implementation

- Query evaluator and optimizer implemented using Python
- Redis is used to store the materialized views



Experiments

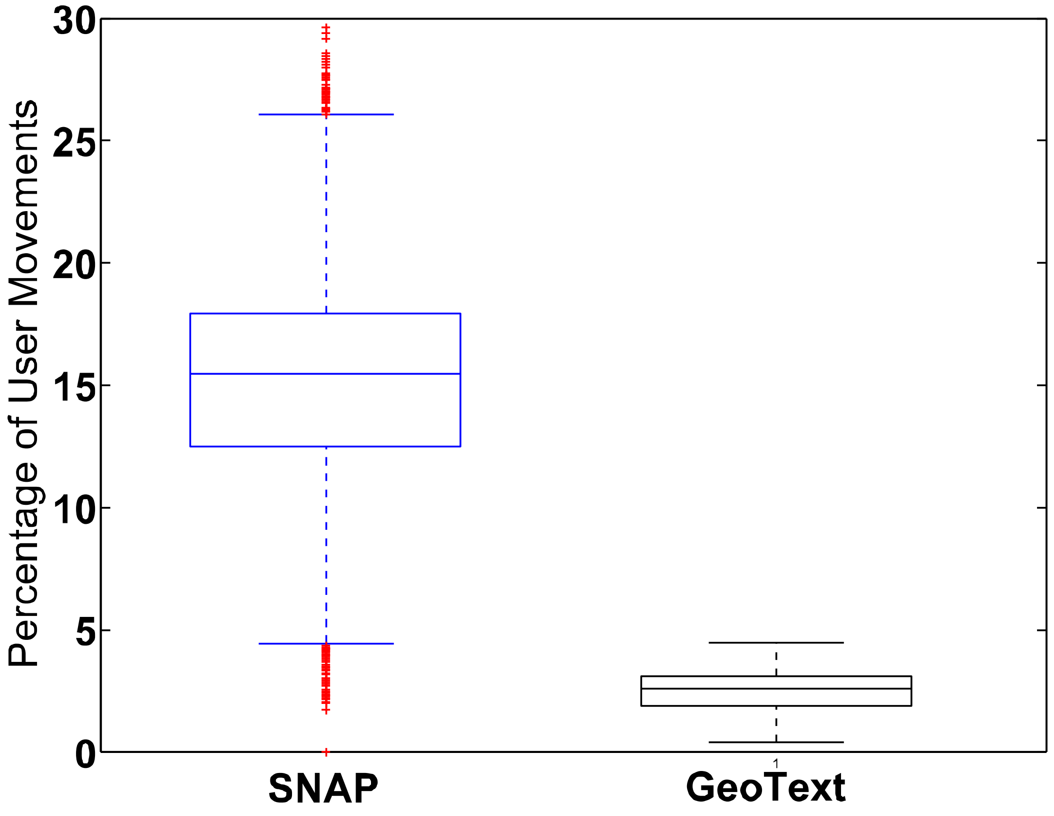
- A cluster with 7 servers
  - each has 2x 2.66 Ghz CPUs, and 48GB RAM
  - 6 Redis nodes + 1 query executor
  - Interconnected with 40GBps network
- Methods for comparison
  - GeoFeed (user-centric)
  - GridView (only use grid-based view)
  - CompView (use both grid-based and composite views)
- Metrics:
  - Resource consumption.
  - Total CPU usage (CPU is the bottleneck in our setup)

Datasets and Scenarios

- Datasets:
  - GeoText (tweeter dataset, light workload)
  - BrightKite from SNAP (location-centric social network, heavy workload)
- Static Scenario
  - Fixed the users and feeds at the initial locations, and ignore their movements
- Dynamic Scenario

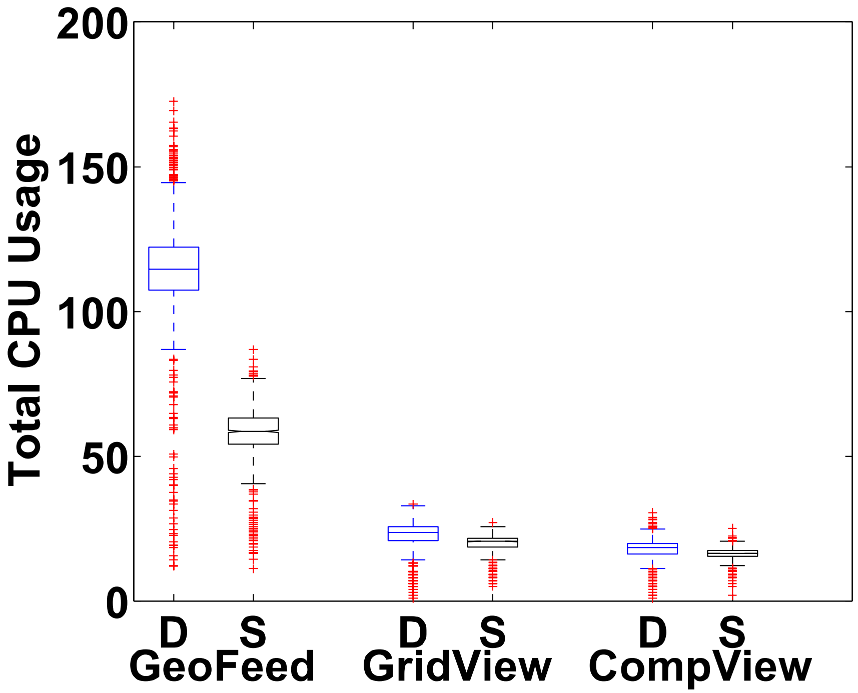
Dynamicity of the Datasets

- SNAP dataset has a higher dynamicity than GeoText

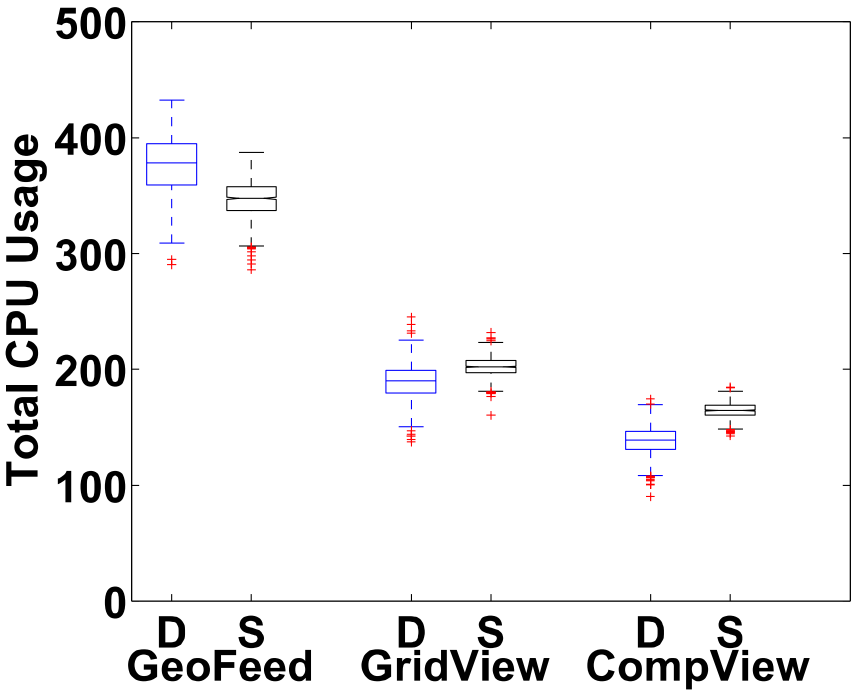


Varying Frequencies of Queries and Updates

- Location-centric (GridView and CompView) outperforms user-centric (GeoFeed), especially in dynamic cases



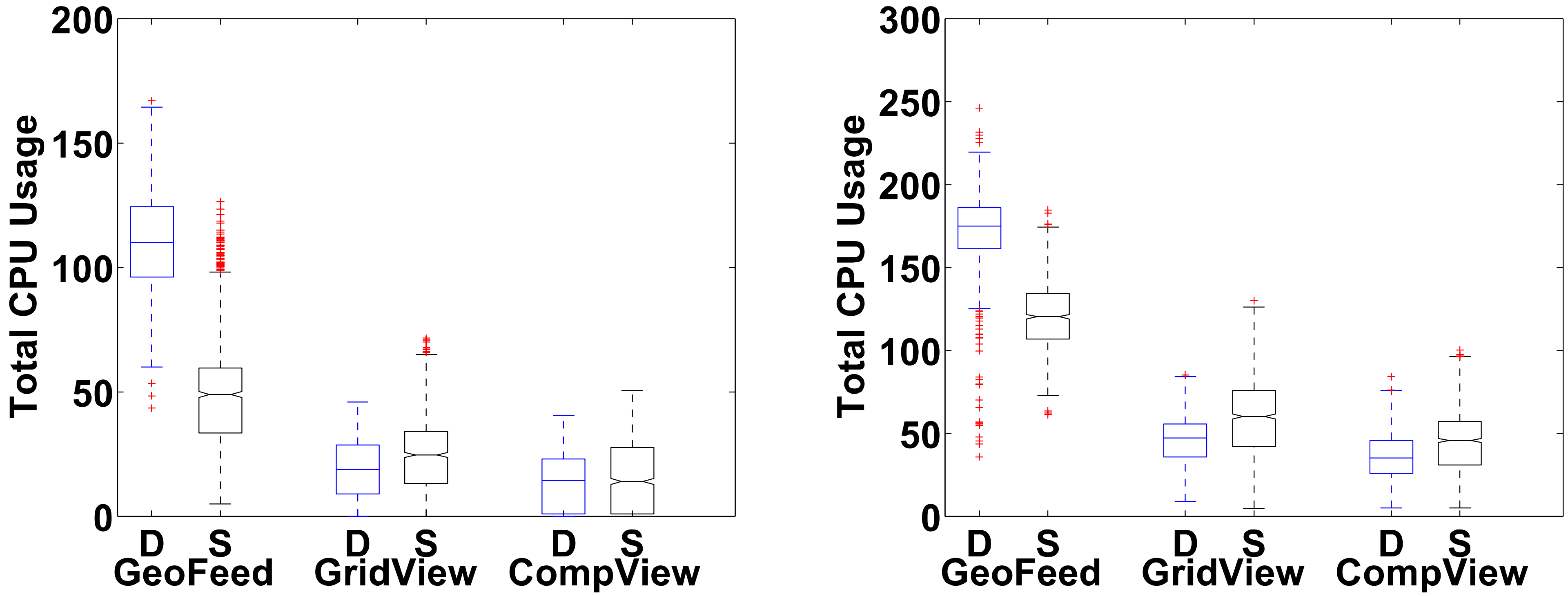
(a) Load Level 100, GeoText



(b) Load Level 1000, GeoText

Varying Frequencies of Queries and Updates

- Similar conclusion on the SNAP dataset

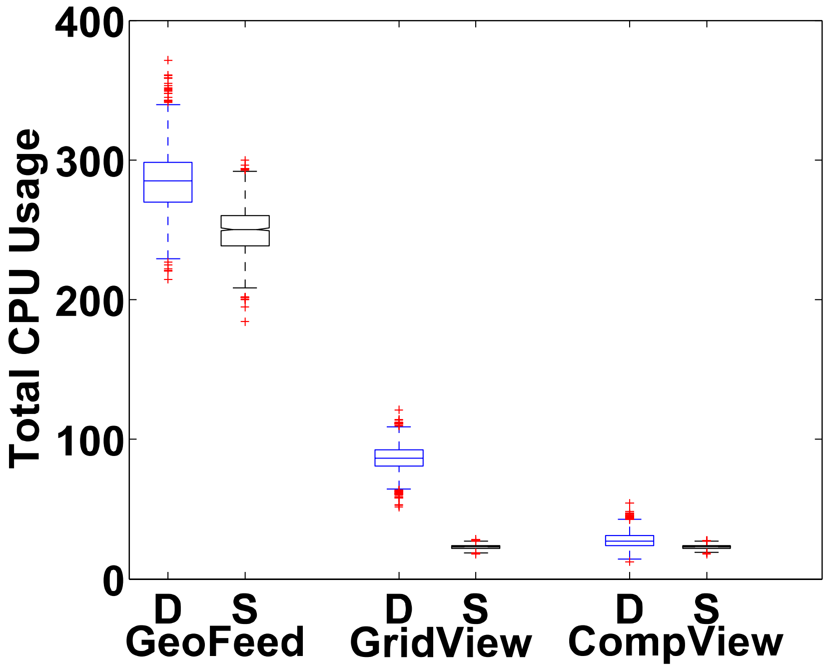


(c) Load Level 67, SNAP

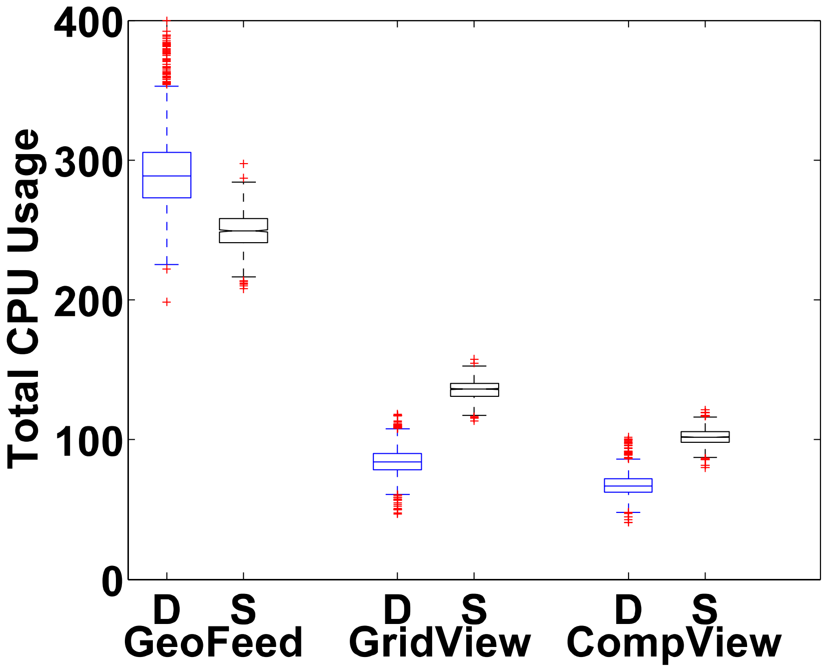
(d) Load Level 667, SNAP

Varying Grid Granularities

- With fine-grained grids, location-centric approaches perform even better under dynamic scenarios in comparing to static ones

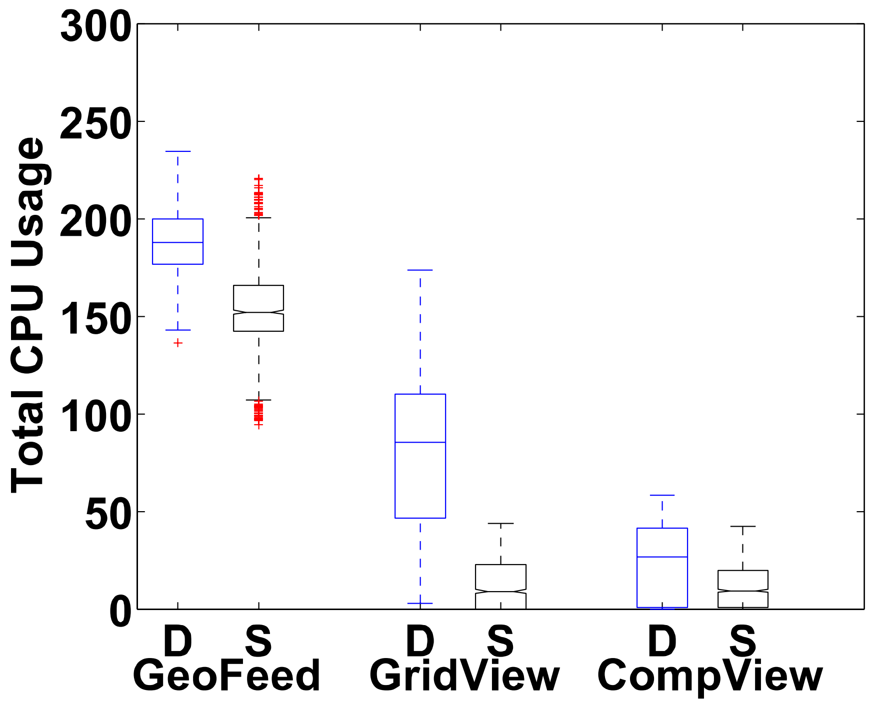


(a) Granularity 125, GeoText

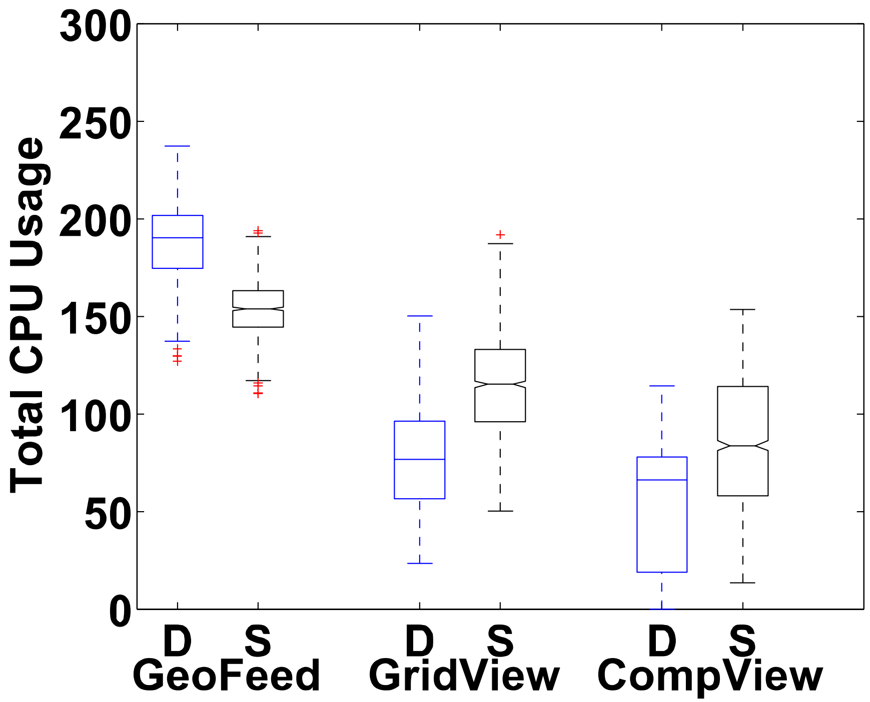


(b) Granularity 750, GeoText

Varying Grid Granularities



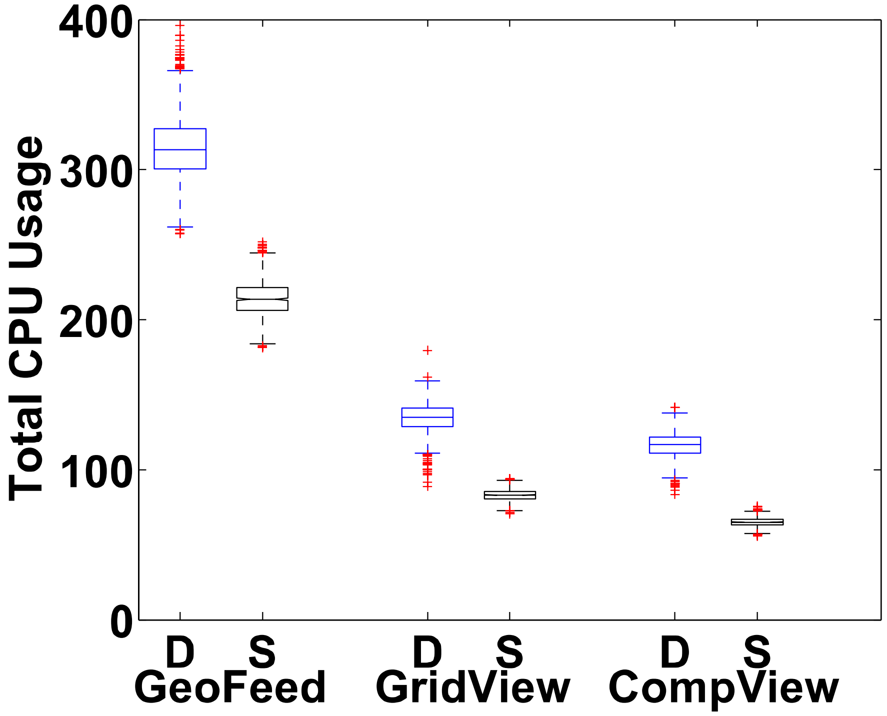
(c) Granularity 125, SNAP



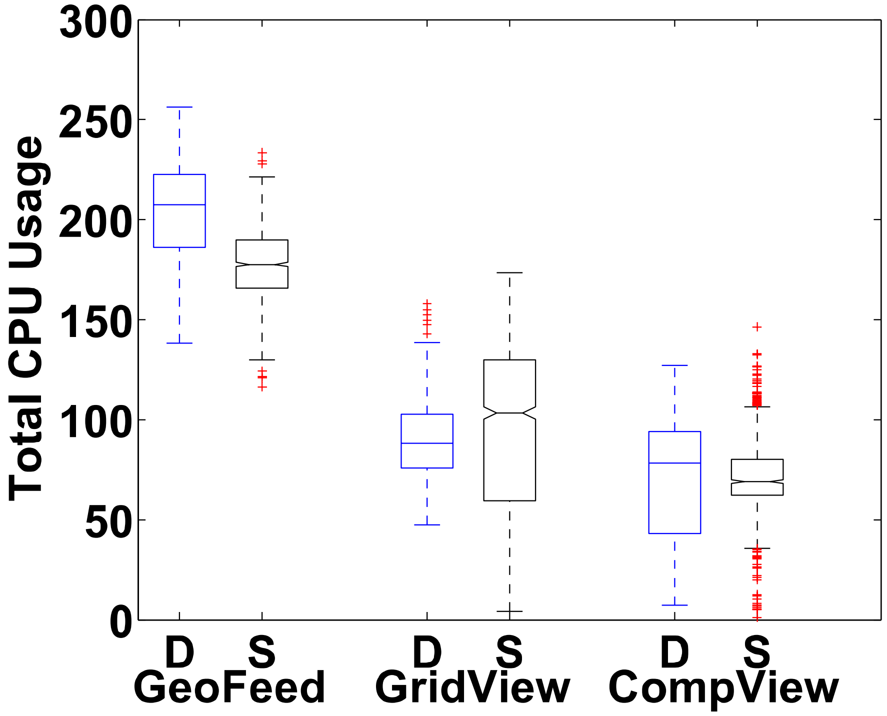
(d) Granularity 750, SNAP

Moving Feeds

- Movements of feeds make the cost slightly higher



(a) Moving Feed, GeoText



(b) Moving Feed, SNAP

Conclusion

- We formulated query optimization problem in location-based feed-following systems.
- In a dynamic setting, location-centric query plans outperform user-centric ones.
- The use of composite views can further reduce the query processing cost.
- Future work:
  - Distributed query executor
  - Filtering features

Thank You!