Drum: A Rhythmic Approach to Interactive Analytics on Large Data

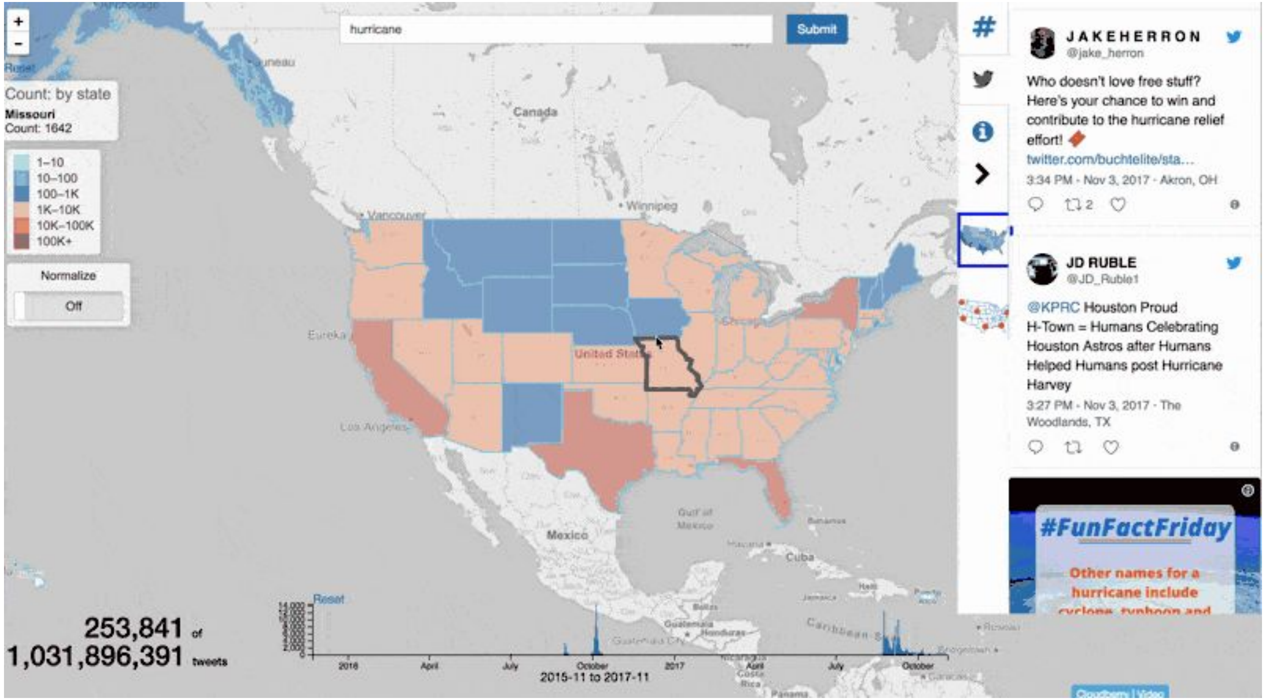
Jianfeng Jia, Chen Li, Michael Carey

UCIrvine

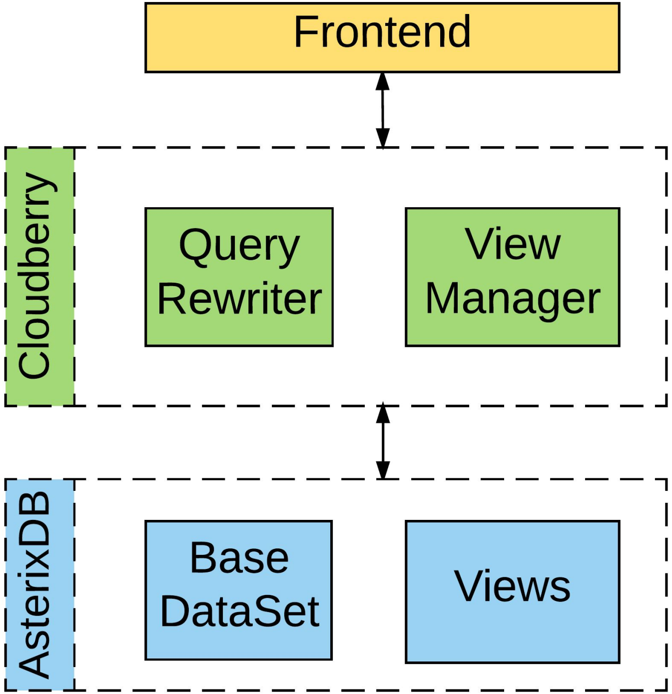
Cloudberry

Understand Big Data

Visualize One Billion Tweets Under $4,000



Architecture



Latency Issue

- Views are not always available
- 2TB data: large
- Aggregation queries: expensive

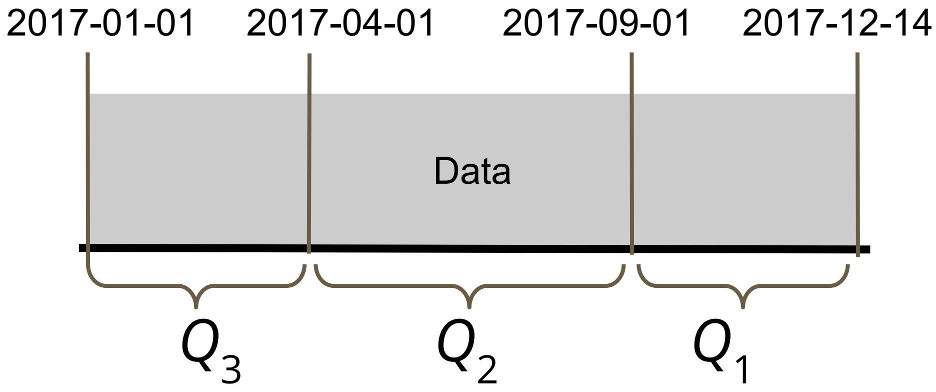
Progressive Computing

Query Slicing

- $Q \rightarrow$ ***mini-queries*** $Q_1, Q_2, \ldots Q_i$
- Each Q_i has an additional range condition on the slicing attribute (e.g. time)
- r_i : size of range condition

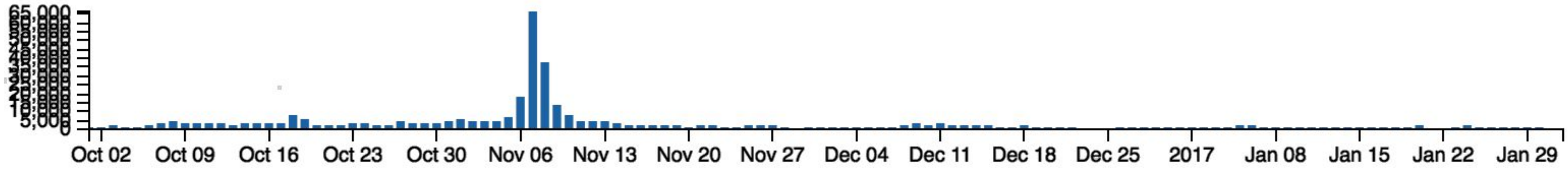
Q:

```
SELECT state, COUNT(*)
FROM twitter t
WHERE contains(t.text, "election")
GROUP BY state;
```



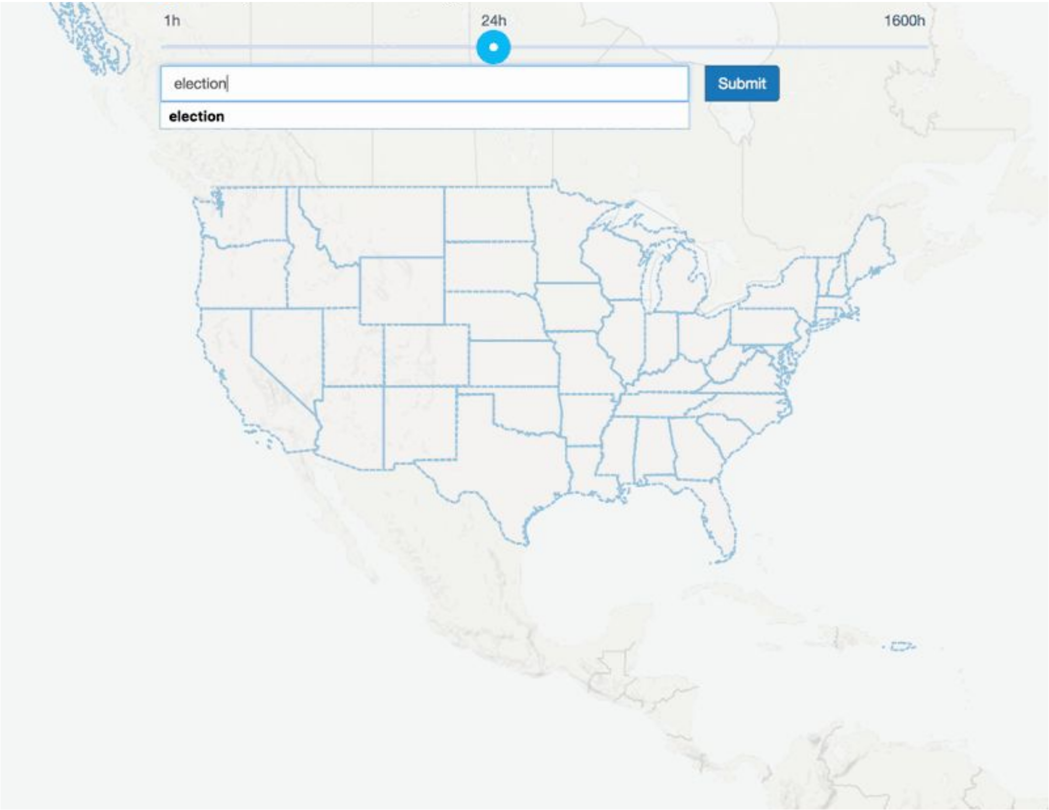
Fixed-length Slicing?

- Difficult to fix the length
- DB performance can fluctuate
- Skewed distribution



Daily distribution of tweets mentioning “**election**”

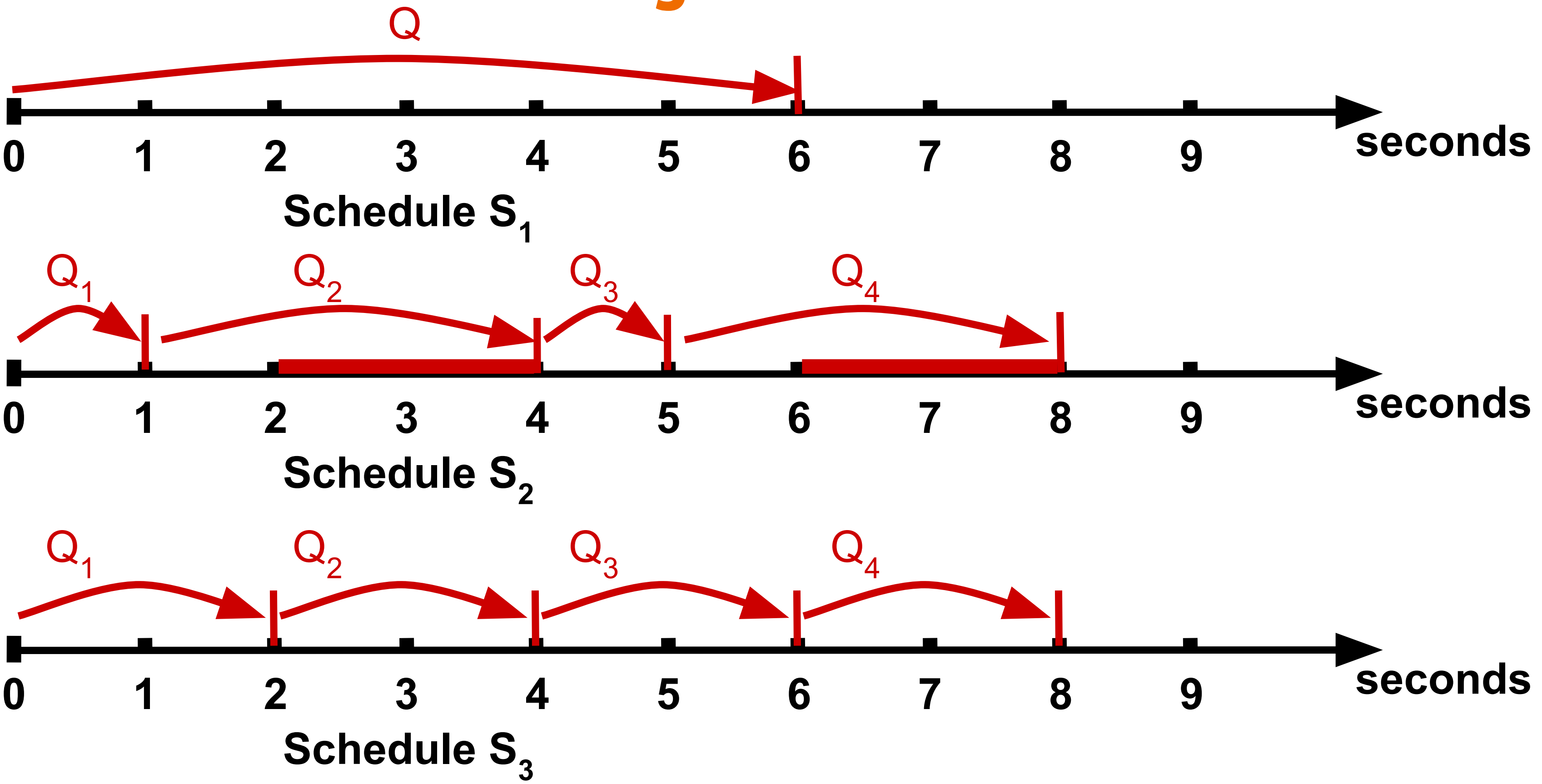
Fix-length Slicing User Experience



UCIrvine

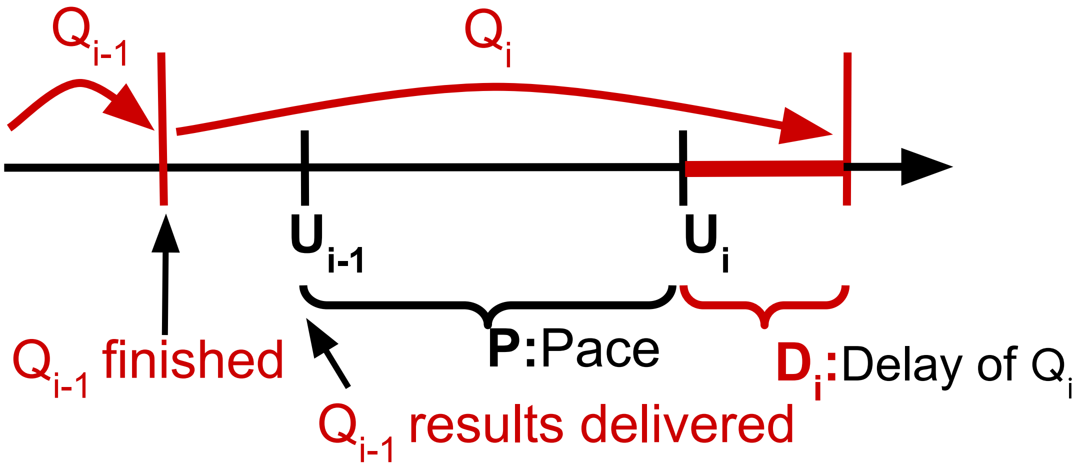
Jianfeng Jia, et al

What Is A Good Slicing Schedule?



Measuring Schedule Cost

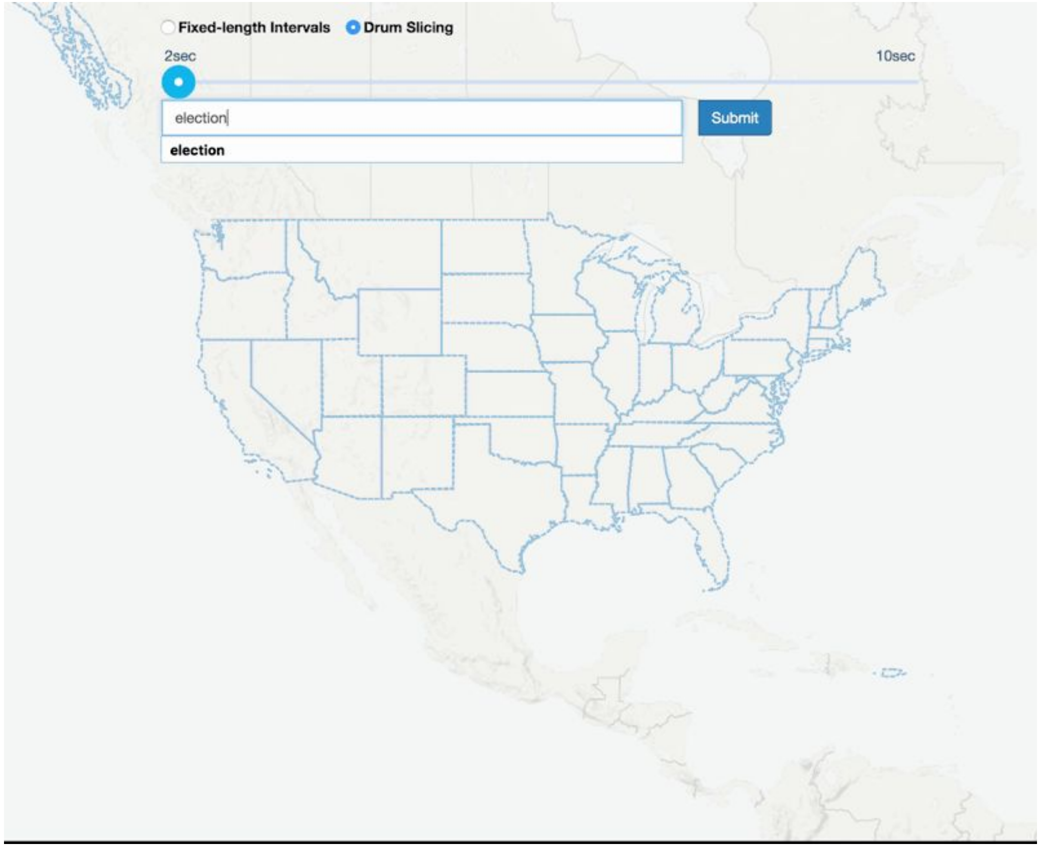
- Total running time
- Smoothness of result delivery



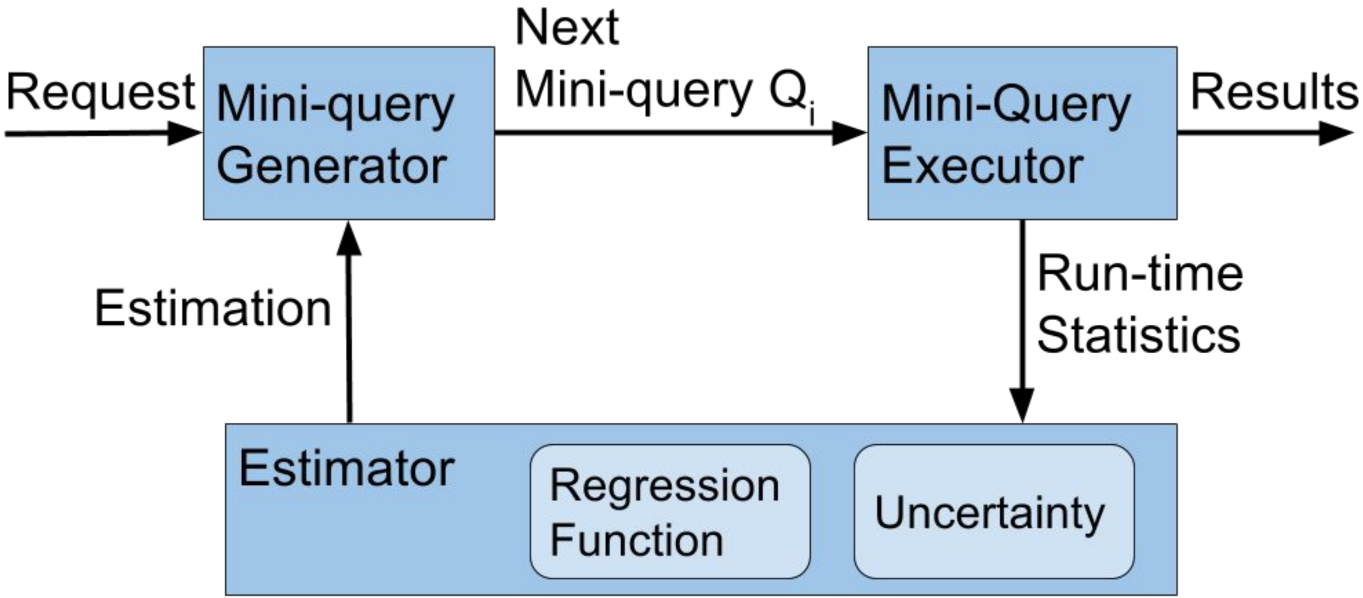
$$Cost(S) = TotalTime(S) + \alpha \sum D_i$$

α : Penalty weight

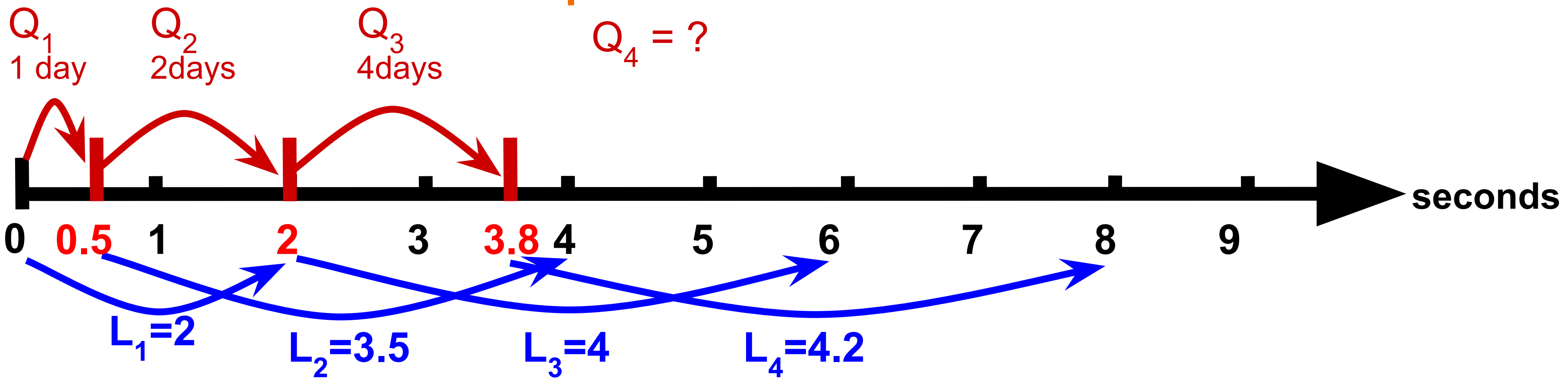
Query Slicing with A Rhythm

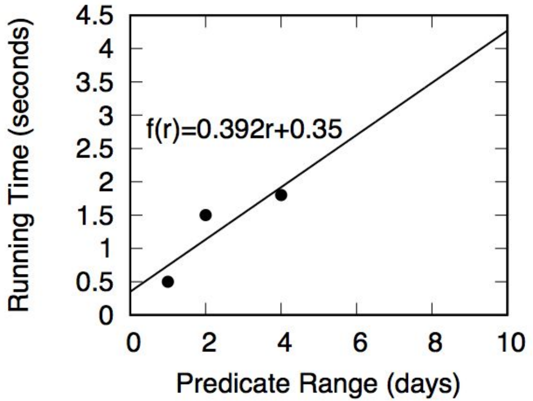


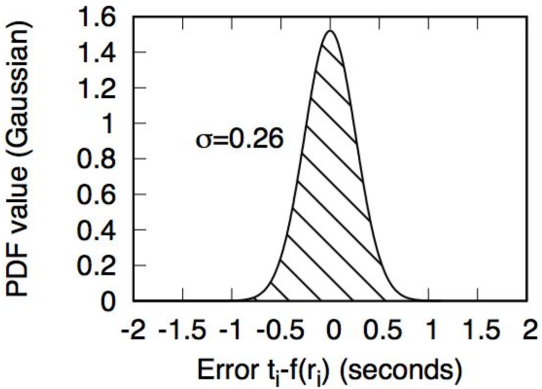
Drum: Adaptive Framework for Query Slicing

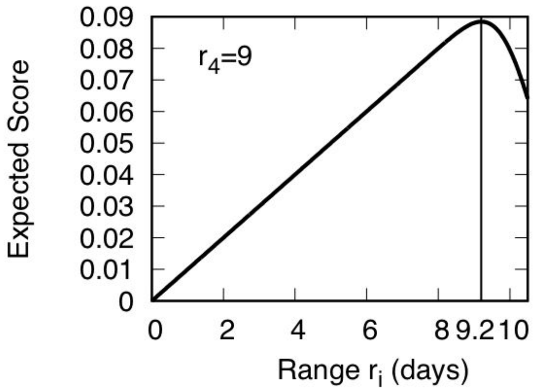


Example: choose Q_4

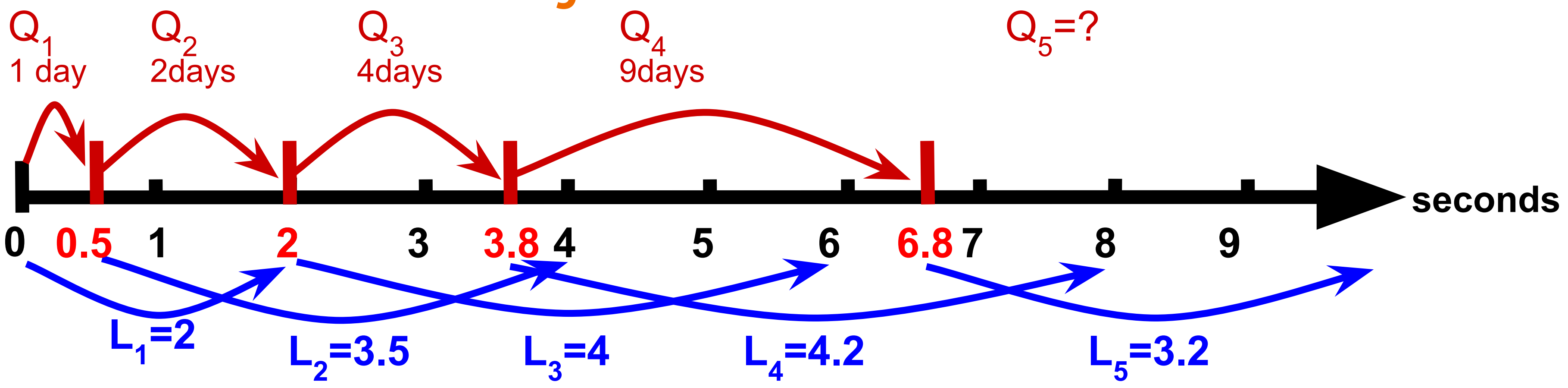


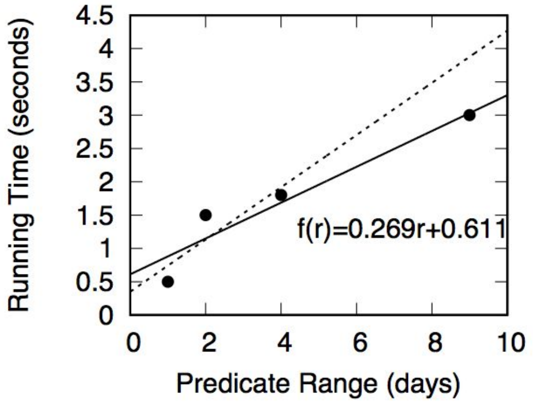


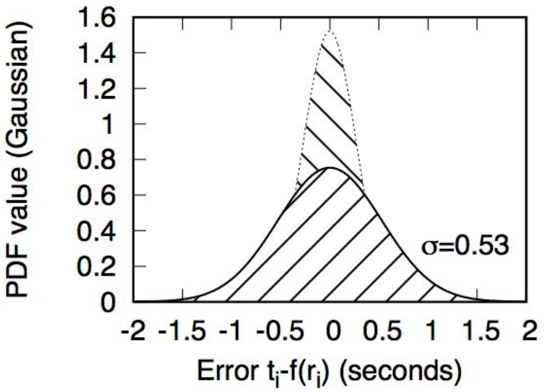


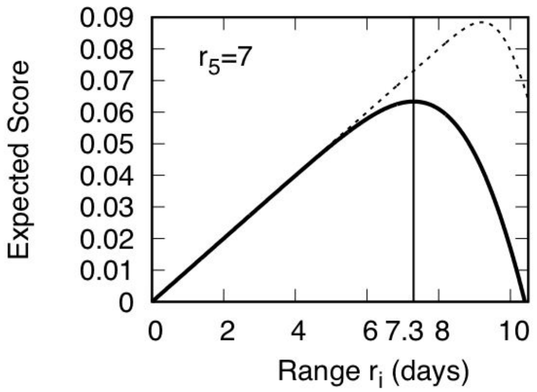


Example: choose Q_5









More results in the paper

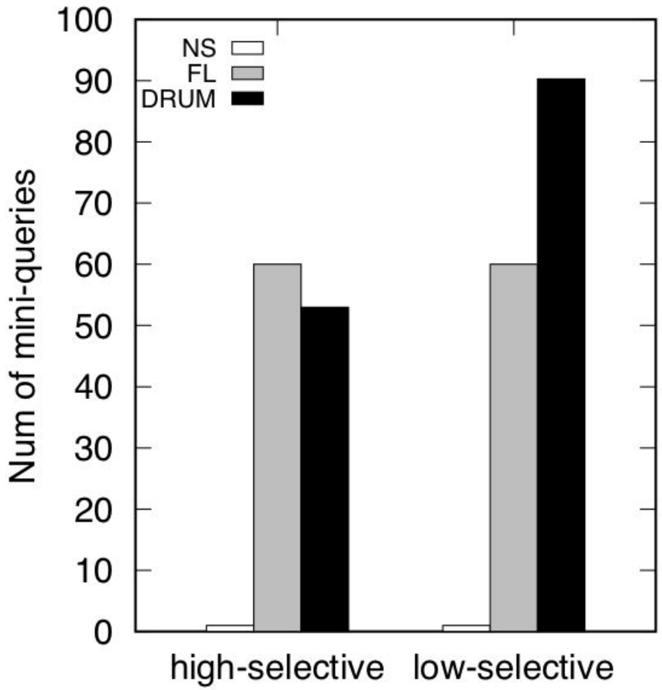
- Formulate an optimization problem
- Develop an adaptive framework
- Define a score function to maximize the gain for each slice

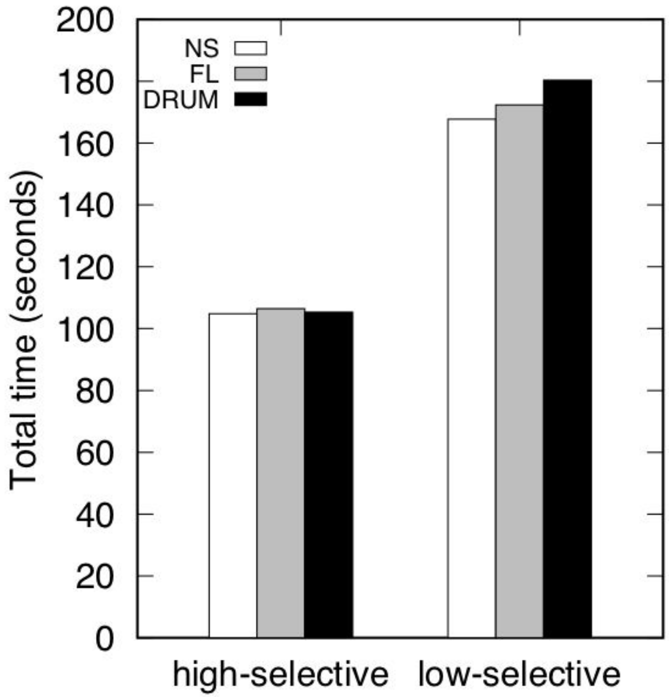
Experimental setting

- Data: 114 million tweets (90GB), Nov. 2016 to Jan. 2017
- DB: AsterixDB 0.9.2
  - Index on "***create_at***" field
  - Inverted index on "***text***" field
- Single machine, 4 cores, 16GB memory
  - with the middleware server
- Queries:

```
SELECT state, COUNT(*)
FROM twitter t
WHERE contains(t.text, $keyword)
GROUP BY state;
```

of mini-queries and total running time



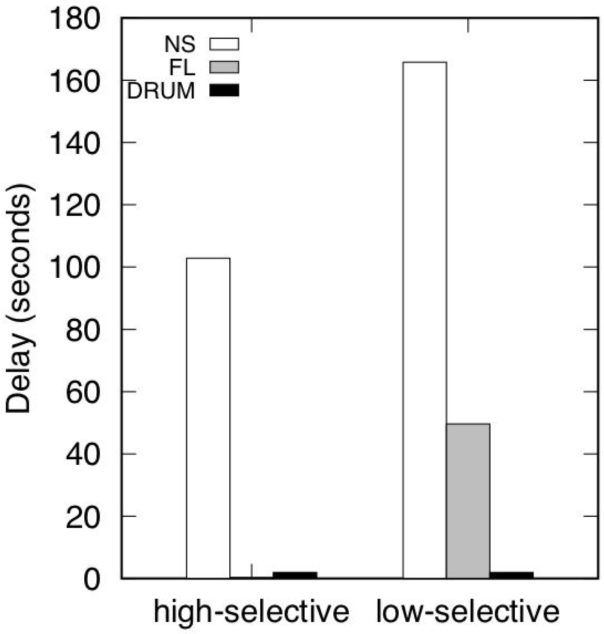


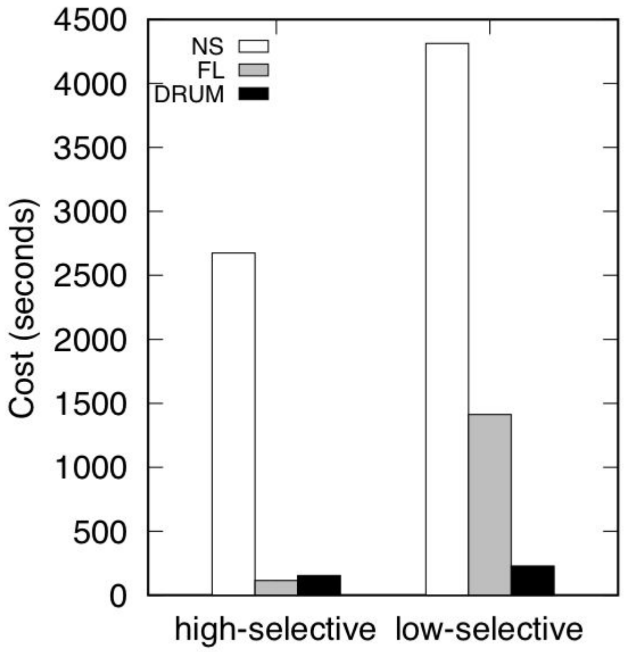
little overhead

UCIrvine

Jianfeng Jia, et al

Total delay and overall cost



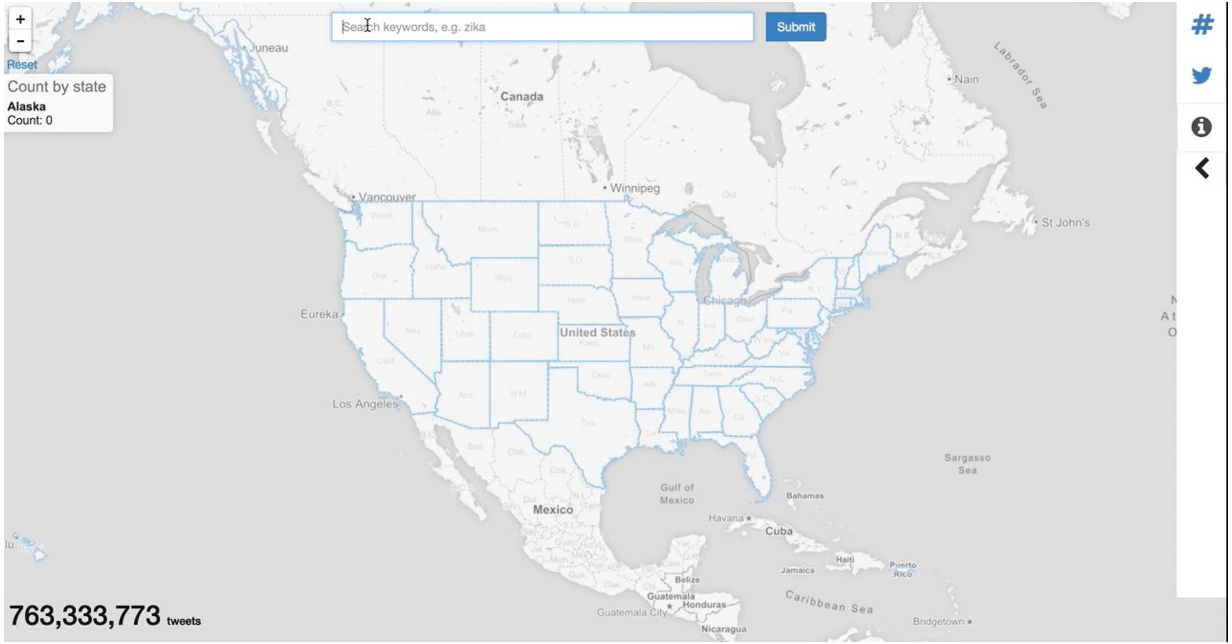


α=25

$$Cost(S) = TotalTime(S) + \alpha \sum D_i$$

Conclusions

- Problem: query slicing to answer big queries responsively
- Solution: Drum, slicing on a dimension adaptively



UCIRVINE

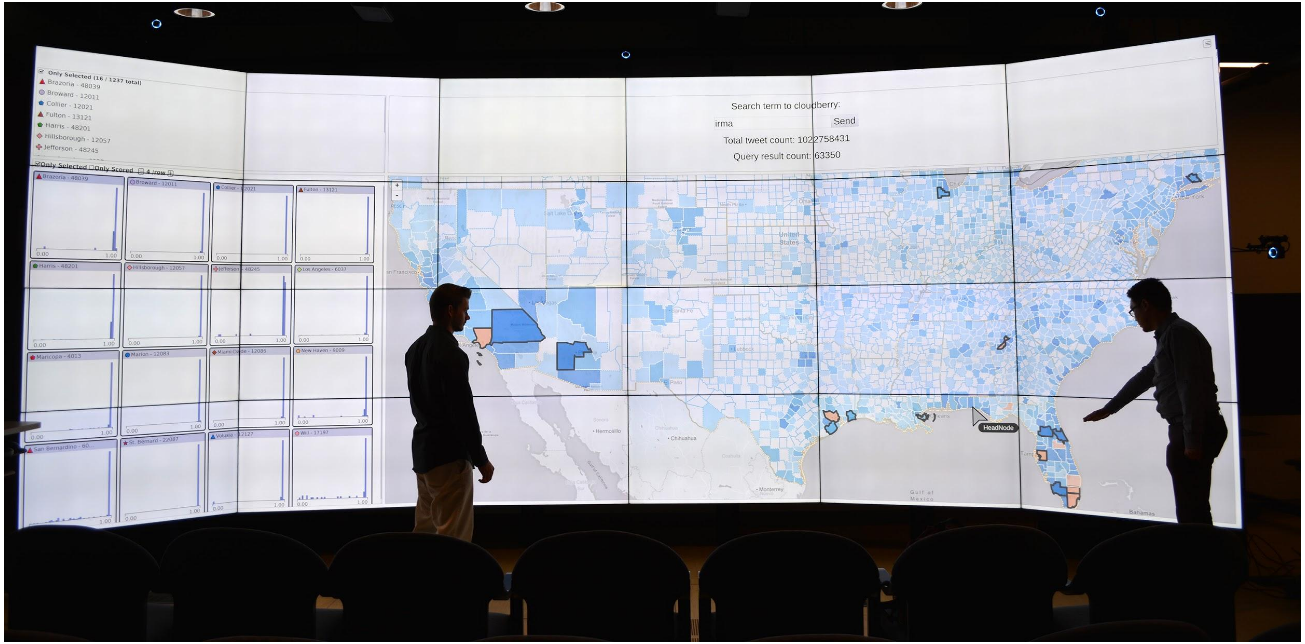
Jianfeng Jia, et al

Cloudberry Project

- Generic middleware
  - Real-time analytics and visualization on big data
  - Using materialized views and query slicing
  - Supporting different frontends and backend DBs

http://cloudberry.ics.uci.edu/

Drum: A Rhythmic Approach to Interactive Analytics on Large Data



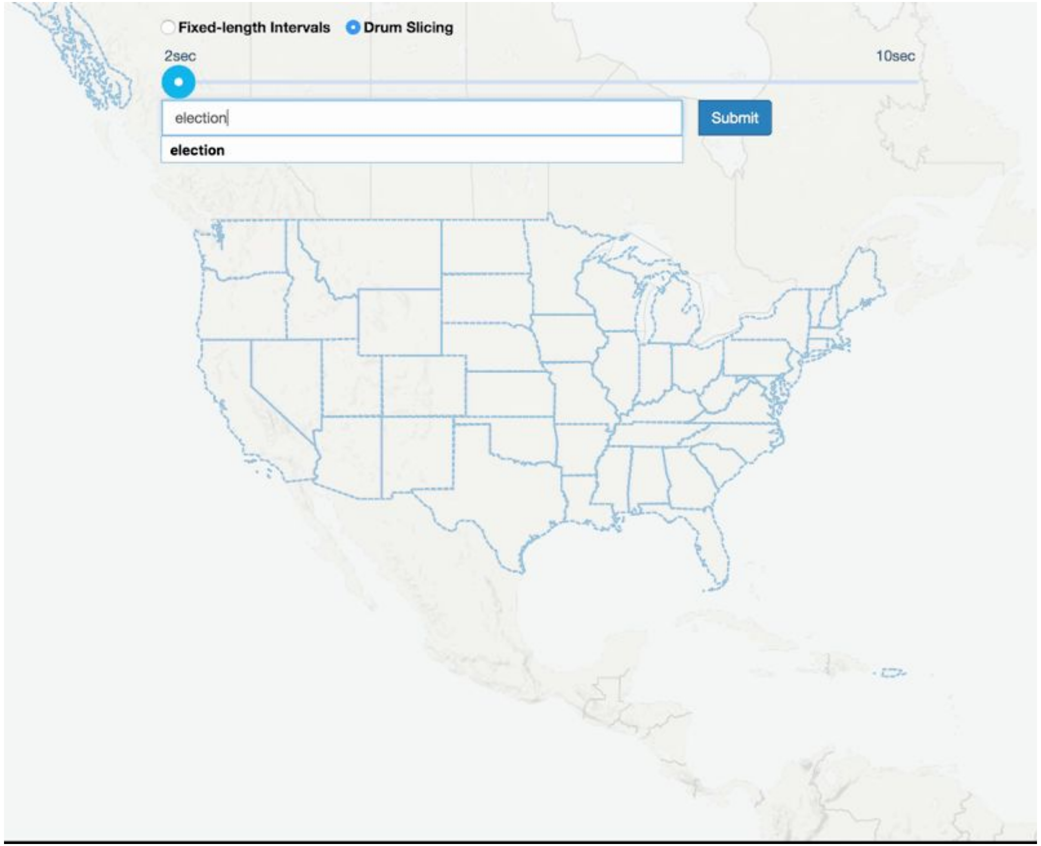
Thank you!

backup

Related Work

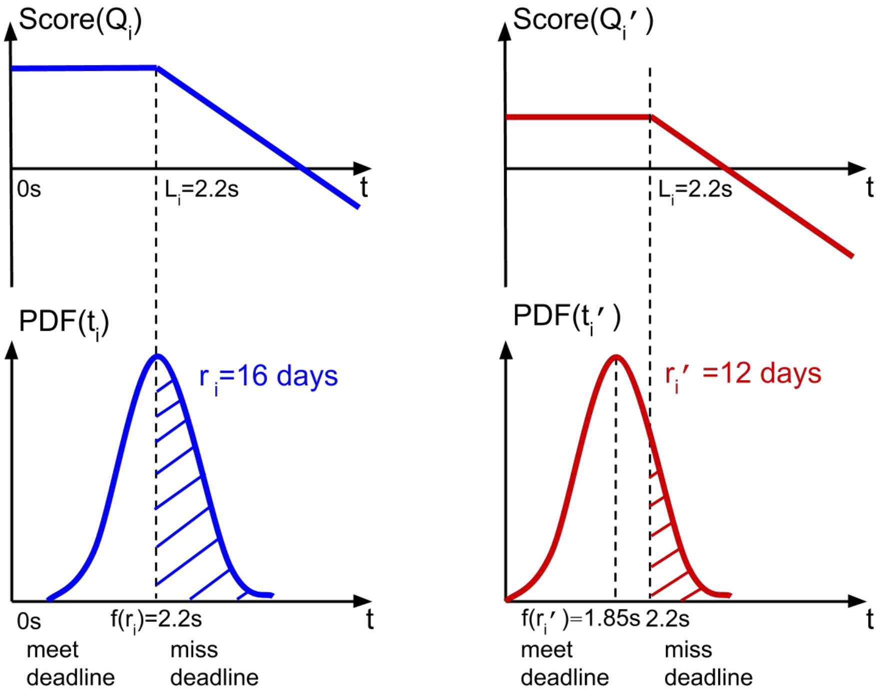
- Existing work
  - Progressive Computing
  - Waiting time in progressive computation
  - Query time estimation
- Main difference
  - middleware solution: treat DB as a “black box”
  - rhythmic slicing
  -

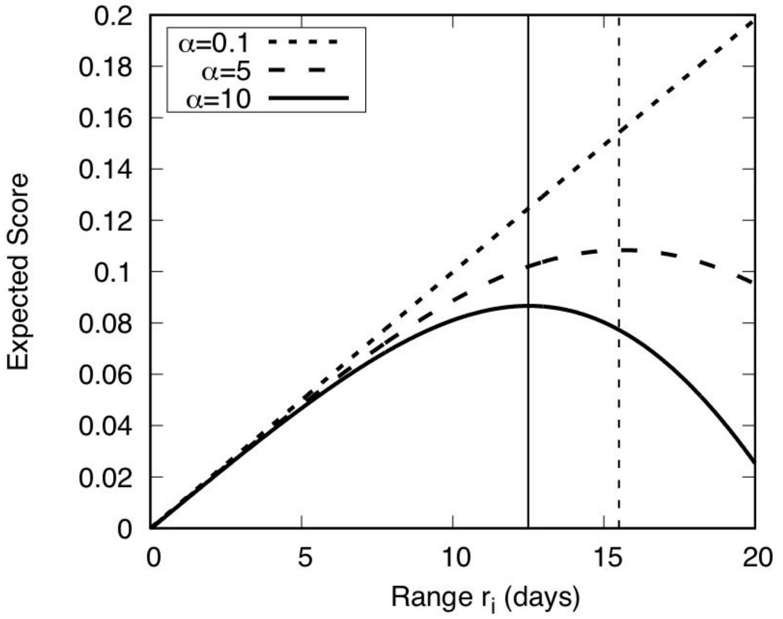
Query Slicing with A Rhythm



Jianfeng Jia, et al

Tradeoff of Running Time and Penalty

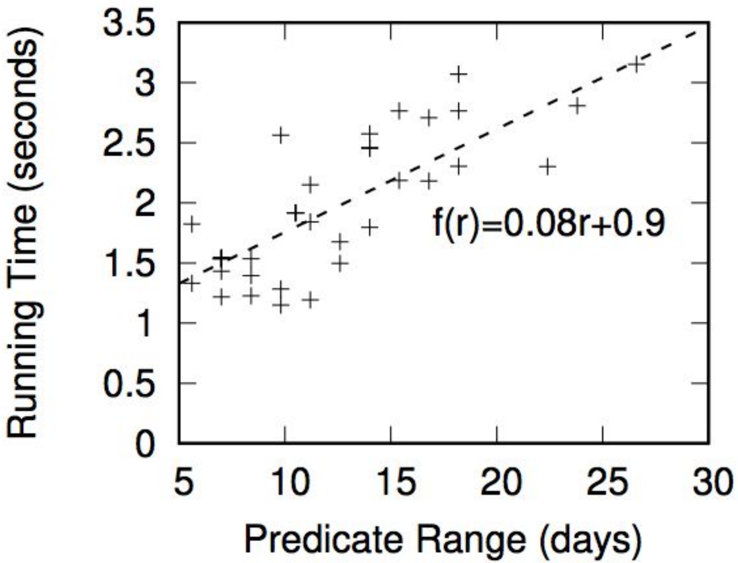


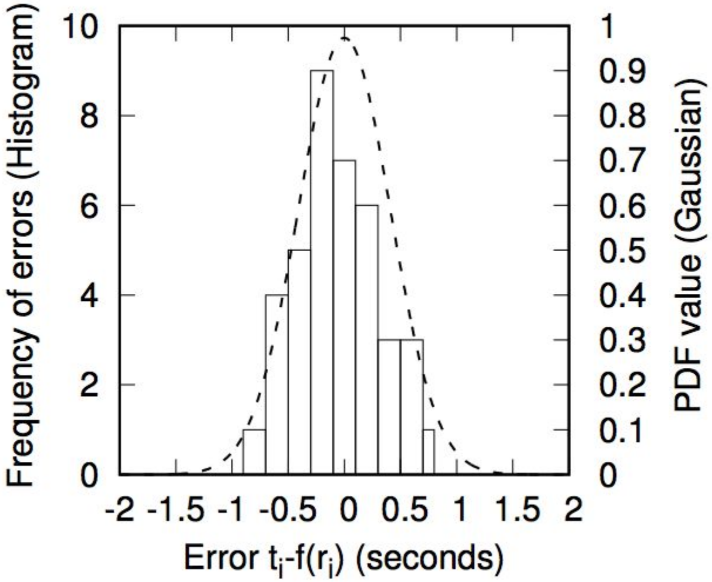


UCIRVINE

Jianfeng Jia, et al

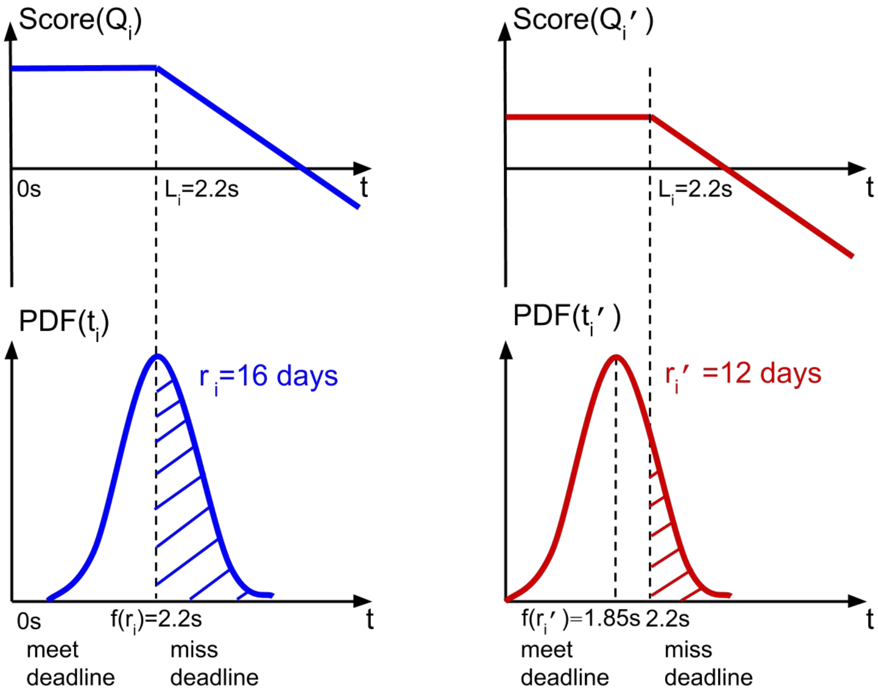
Linear Regression Function and Its Uncertainty





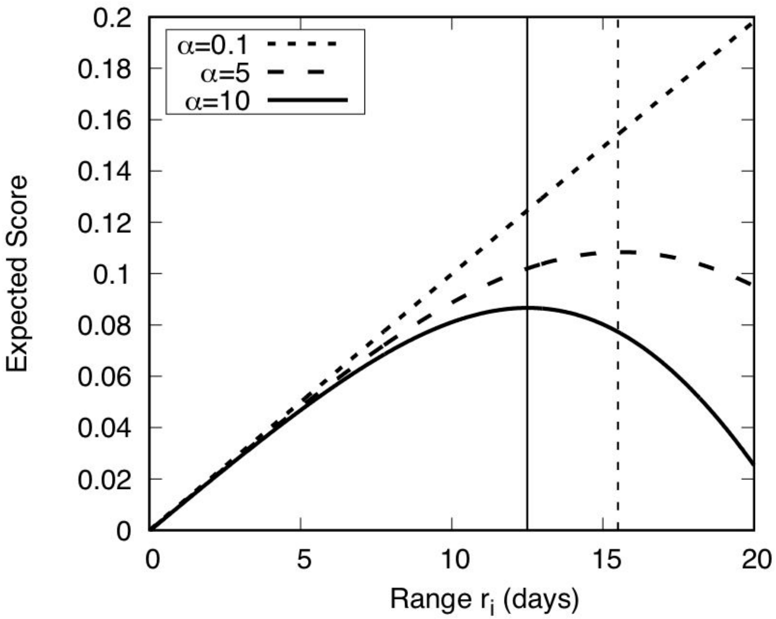
Tradeoff of Running Time and Penalty

$$score(Q_i) = \begin{cases} \frac{r_i}{I} & \text{if } t_i <= L_i \\ \underbrace{\boxed{\frac{r_i}{I}}}_{\text{Progress}} - \alpha \frac{\overbrace{\boxed{t_i - L_i}}^{\text{Delay}}}{C_i L_i} & \text{otherwise.} \end{cases}$$



- r_i : Size of range predicate
- I : Total interval of the slicing attribute
- α : Weight on penalty
- t_i : Actual running time
- L_i : Time Limit of Q_i
- C_i : Total number of mini-queries (estimation)

Choosing r_i to Maximize the Expected Score



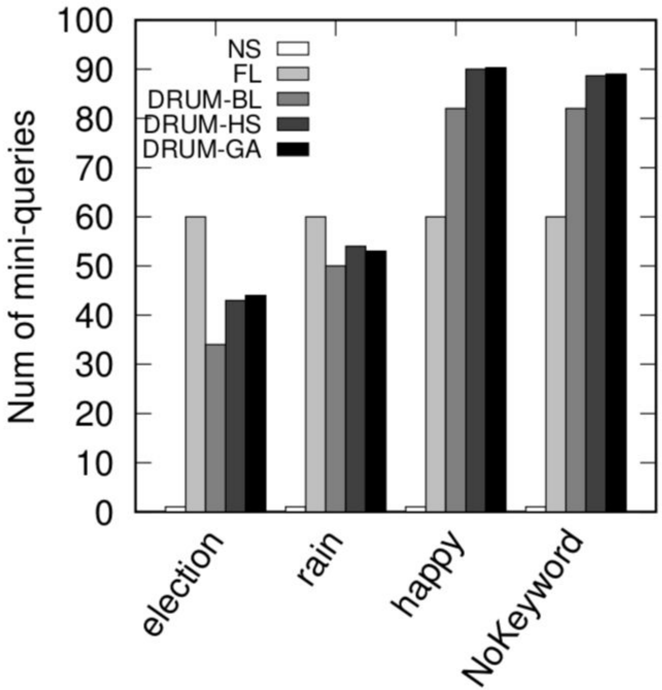
$$E(score(Q_i)) = \frac{r_i}{I} - \alpha \int_{L_i}^{\infty} \frac{(t - L_i)}{C_i L_i} P(t|f(r_i))dt.$$

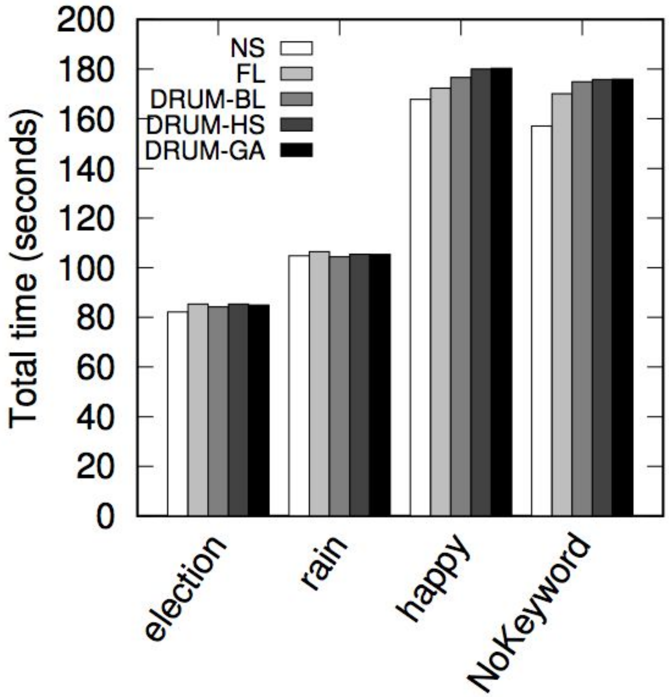
Optimized r_i using Gaussian model:

$$r_i = \frac{\sqrt{2}\sigma z_{max} + L_i - a_0}{a_1}.$$

$$z_{max} = \mathrm{erf}^{-1}\left(\frac{2C_iL_i}{a_1 I \alpha} - 1\right).$$

of mini-queries and total running time

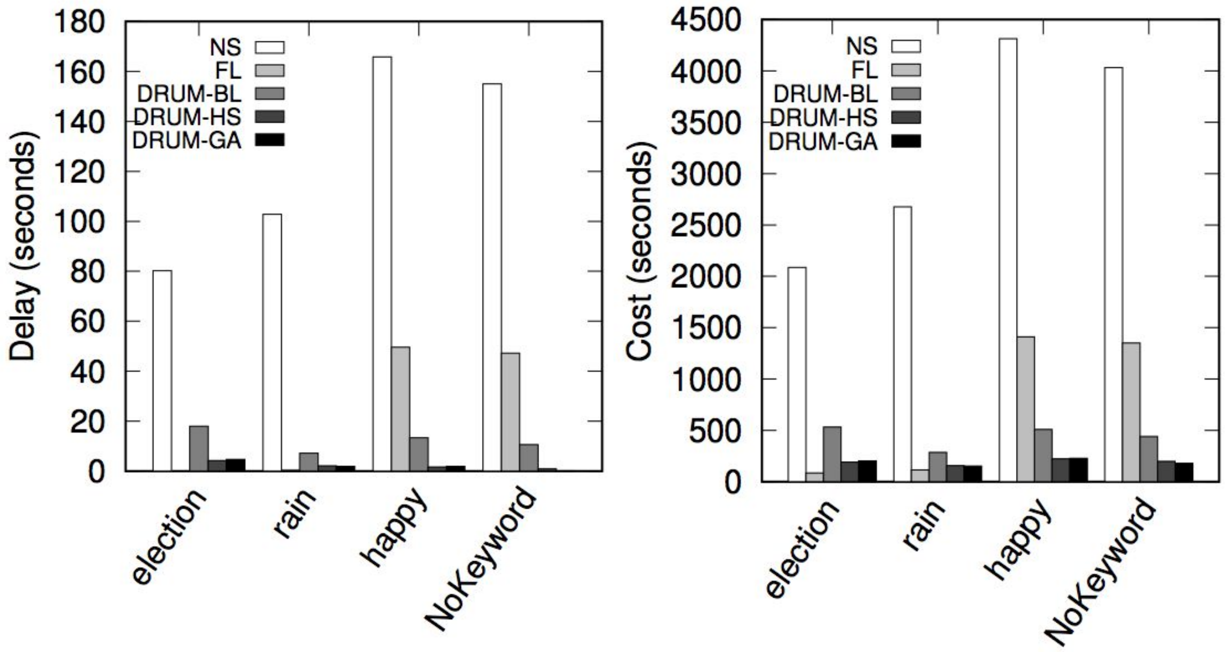




UCIRVINE

Jianfeng Jia, et al

Total delay and overall cost



α=25

$$Cost(S) = TotalTime(S) + \alpha \sum D_i$$

Jianfeng Jia, et al

Understand Big Data