Drum: A Rhythmic Approach to Interactive Analytics on Large Data

Jianfeng Jia, Chen Li, Michael Carey
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

Simple JPEG
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

```
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
What Is A Good Slicing Schedule?

Schedule $S_1$

Schedule $S_2$

Schedule $S_3$
Measuring Schedule Cost

- Total running time
- Smoothness of result delivery

\[ \text{Cost}(S) = \text{TotalTime}(S) + \alpha \sum D_i \]

\( \alpha \) : Penalty weight
Query Slicing with A Rhythm
Drum: Adaptive Framework for Query Slicing

1. Request to Mini-query Generator
2. Mini-query Generator generates Mini-query $Q_i$
3. $Q_i$ is sent to Mini-Query Executor
4. Estimator provides Estimation, Regression Function, and Uncertainty
5. Mini-Query Executor provides Run-time Statistics
6. Results are generated

UCIrvine
Example: choose $Q_4$

$Q_1 = 1$ day
$Q_2 = 2$ days
$Q_3 = 4$ days
$Q_4 = ?$

$L_1 = 2$
$L_2 = 3.5$
$L_3 = 4$
$L_4 = 4.2$

Running Time (seconds)

$\text{PDF value (Gaussian)}$

Expected Score

Predicate Range (days)

Error $t_i - f(r_i)$ (seconds)

Range $r_i$ (days)
Example: choose $Q_5$

$Q_1 =$ 1 day
$Q_2 =$ 2 days
$Q_3 =$ 4 days
$Q_4 =$ 9 days
$Q_5 =$ ?

$L_1 = 2$
$L_2 = 3.5$
$L_3 = 4$
$L_4 = 4.2$
$L_5 = 3.2$

Running Time (seconds)

PDF Value (Gaussian)

Expected Score

Predicate Range (days)

Error $t_f(r_f)$ (seconds)

Range $r_f$ (days)
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

- **Left Diagram:**
  - **Y-axis:** Num of mini-queries
  - **X-axis:** high-selective, low-selective
  - **Legend:**
    - NS
    - FL
    - DRUM
  - **Observation:**
    - DRUM has the highest number of mini-queries.

- **Right Diagram:**
  - **Y-axis:** Total time (seconds)
  - **X-axis:** high-selective, low-selective
  - **Legend:**
    - NS
    - FL
    - DRUM
  - **Observation:**
    - DRUM has the highest total running time.
    - Little overhead noted.

---

UCIRVINE

Jianfeng Jia, et al
Total delay and overall cost

\[ \text{Cost}(S) = \text{TotalTime}(S) + \alpha \sum D_i \]
Conclusions

- **Problem:** query slicing to answer big queries responsively
- **Solution:** Drum, slicing on a dimension adaptively
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
Tradeoff of Running Time and Penalty

Score($Q_i$)

PDF($t_i$)

$0s \rightarrow L_i=2.2s \rightarrow t$

$r_i=16$ days

meet deadline

miss deadline

Score($Q'_i$)

PDF($t'_i$)

$0s \rightarrow f(r'_i)=1.85s 2.2s \rightarrow t$

$r'_i=12$ days

meet deadline

miss deadline

Expected Score

Range $r_i$ (days)

$\alpha=0.1$ 

$\alpha=5$

$\alpha=10$
Linear Regression Function and Its Uncertainty

- **Equation**: $f(r) = 0.08r + 0.9$

- **Graph**: Depicts the relationship between running time (seconds) and predicate range (days). The linear regression line is shown, along with data points.

- **Histogram**: Illustrates the frequency of errors (histograms) and their PDF values (Gaussian distribution). The histogram shows the distribution of errors, with the PDF curve fitting the data.
Tradeoff of Running Time and Penalty

\[ \text{score}(Q_i) = \begin{cases} \frac{r_i}{I} - \frac{t_i - L_i}{\alpha C_i L_i} & \text{if } t_i \leq L_i \\ \text{otherwise.} & \end{cases} \]

- \( r_i \): Size of range predicate
- \( I \): Total interval of the slicing attribute
- \( \alpha \): 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

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

Optimized $r_i$ using Gaussian model:

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

$$z_{max} = \text{erf}^{-1}(\frac{2C_iL_i}{a_1I\alpha} - 1).$$
# of mini-queries and total running time

![Graph showing the number of mini-queries and total running time for different scenarios. The graphs compare various techniques such as NS, FL, DRUM-BL, DRUM-HS, and DRUM-GA across different datasets including election, rain, happy, and NoKeyword.](image-url)
Total delay and overall cost

\[ Cost(S) = TotalTime(S) + \alpha \sum D_i \]
Understand Big Data