High Performance Spatial Visualization of Traffic Data

Shashi Shekhar, CT Lu, Alan Liu
Spatial Database Lab
Department of Computer Science
University of Minnesota
shekhar@cs.umn.edu
Outline

- Introduction
- Requirement Specifications
- Software Design
- Bottleneck Analysis
- Performance Tuning
- Conclusion
Motivation for Visualizing Traffic Data

- Freeway Performance Measurement System, 01’ Varaiya
  - Transportation Manager
    - How did the highways perform on a day compared to the average?
    - Which locations are worst performers?
  - Traffic Engineering
    - Where are the congestion (in time and space)?
    - Which of these recurrent congestion?
    - Which loop detection are not working properly?
    - How congestion start and spread?
  - Traveler, Commuter
    - What is the travel time on a route?
    - Will I make to destination in time for a meeting?
    - Where are the incident and events?
  - Planner and Research
    - How much can information technique to reduce congestion?
    - What is an appropriate ramp meter strategy given specific evolution of congestion phenomenon?
Question 1

- Manager: How did the highways perform on a day compared to the average?
  - *Comparison of traffic videos*

• **Left video**: Avg traffic on Thursday during January 1997
• **Right video**: Traffic on Monday, January 20, 1997
  - I494 at I35W
  - Missing data-35E South of 94, 77 South of 494
Question 2

- Engineer: Which loop detector are not working properly?

- Measure: Volume
- Space-time of data plot
  - Stations on a free way vs. Time of day.
  - Each point: 5 minute volume

![Average Traffic Volume (Time v.s. Station)](image)

- Trends:
  - Station 9 is a spatial outlier.
  - Missing data at 2:45PM for all stations.
  - Missing data for stations 58, 61, 62, and 63 from 8:35AM to 10:10AM
  - Rush hours
Question 3

- Engineer: How congestion starts and spreads?
  - **Measure**: Occupancy
  - **Space-time of date plot**
    - See right side
    - I-94 West traffic by 5-minute interval

- **Right side**
  - Space time for I-94 W on 1/10/1997.
  - Each 5-minute measurement is a point.
  - Trends: a red triangle developed.

- **Left side**: Freeway map
Demo

- Traffic Video
- Comparison of Traffic Video
- Station-Time of day matrix
- Data cube visualization
Application Domain

- Traffic sensors on highways

- Data collected
  - Volume = number of cars that pass a detector during time $t$
  - Occupancy = % of time a detector is occupied by vehicles during time $t$
Problem Statement

- High-performance Analysis Techniques
  - Traffic data grow fast
  - Hard to analyze manually
  - Performance

- Given: Traffic data

- Requirement:
  - High performance spatial tools and techniques to generate critical visualizations of general traffic data

- Our approach:
  - Use a lattice framework:
    - Generate critical visualization of traffic data.
    - Analyze historical traffic data.
Related Work

- Map Cube: A Visualization Tool for Spatial Data Warehouses (Shekhar, ’99)
  - Map cube operators: an album of maps

- Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals (Jim Gray, ’97)
  - Cube operators based on a relational representation of aggregate data

- Freeway Performance Measurement System (Pravin, ’01)
  - delay, productivity Q

- Traffic Visualization (Francis Li, ’00)
  - Traffic view from camera
  - Plot of speed/volume vs time
Basic Concept: Data Cube

- **Data cube**
  - *Dimensions*: determining attributes
  - *Measures*: determined attribute.
  - *Each dimension and measure is a field in a relational database table.*

- **Example: DTS cube**
  - *D: date, T: time of day, S: station id*
  - *volume = f(d,t,s)*

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Sid</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997-01-01</td>
<td>06:00</td>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>138</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>77</td>
</tr>
<tr>
<td>1997-01-02</td>
<td></td>
<td></td>
<td>107</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>181</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>237</td>
</tr>
<tr>
<td>1997-01-03</td>
<td></td>
<td></td>
<td>90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>180</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>226</td>
</tr>
</tbody>
</table>

Table 1: SQL Table for Data Cube
Basic Concept: Concept Hierarchy

- **Time dimension**

  - Hour
  - Minute
  - Second

- **Date dimension**

  - Year
  - Season
    - Month
    - Week
    - Day of week
    - Day

- **Space dimension**

  - County
  - Freeway
    - Freeway+Direction
    - Station
  - Region
    - Detector
Basic Concept: Dimension Lattice

- Dimension lattice
  - Any combination of dimensions produces a different view.
  - $N$ dimensions give $2^N$ lattice nodes/subsets.

- $D$: date
- $T$: time of day
- $S$: station id
Data Cube Visualization: Example 1

- DTS data cube visualization
  - Visualize DT, ST and DS nodes.
Data Cube Visualization: Example 2

- Dimensions
  - \( D \): date
  - \( T \): time of day
  - Space \( X \): \( x \) coordinates of stations
  - Space \( Y \): \( y \) coordinates of stations

- \( DTS^2 \) cube
Traffic Video from Data Cube’s Point of View

- Involved subset \((S^2, T_D)\)

- Two choices on D-dimension
  - *Slice on D-dimension for one day.*
  - *Aggregate, e.g. \( \text{avg(volume)} \) over all days.*

\[\text{D-axis: 1997/01/06}\]
Testing

- Traffic video
  - Test cases
    - Shows volumes of all stations from 00:00AM to 23:55PM on Jan 9th, 1997
    - Shows video for different types of data, i.e. occupancy, average volume
    - Shows video on different days
  - Assertions
    - For a snapshot, manually calculate the volumes of several stations and check with the output from program.
    - Visual inspection

- Data cube visualization
  - Test cases
    - Show average volume of each weekday and time of day matrix for all stations and the month of Jan 1997
    - Show average volume of each weekday and all stations on I-35W south for the month of Jan 1997
    - Show average volume of all stations on I-35W south and time of day for the month of Jan 1997
  - Assertions
    - For each test case, compare several rows of data with manually calculated data.
    - Visual inspection
Outline

- Introduction
- Requirement Specifications
- Software Design
- Bottleneck Analysis
- Performance Tuning
- Conclusion
High-level Design: Major Components

- Web-based application

  GUI  ⟷  CGI  ⟷  Database

- GUI
  - Implemented in Java Applet.
  - Sends requests to CGI to request data.
  - Displays results in graphics.
  - Contains 19 classes, 4440 lines of code

- CGI
  - Implemented in Perl.
  - Receives requests from applets.
  - Fetches data from database and send back to applets.
  - Includes 4 files, 356 lines of code

- Database: MySQL Distrib 3.23.38

- Web server
  - Sun Ultra Enterprise 250
  - Dual 400MHz UltraSPARC-II CPUs
  - 1024MB RAM
  - http://www.cs.umn.edu/research/shashi-group/vis/traffic.htm
Outline

- Introduction
- Requirement Specifications
- Software Design
- Bottleneck Analysis
- Performance Tuning
- Conclusion
Bottleneck Analysis of High-level Design

- $T_{all}$: Total time
- $T_{db}$: Database time
- $T_{network_k}$: Network time = $T_{all} - T_{db}$
- Display rate: 2 snapshots/second
Result of Code Profiling

- Experiment
  - Query: get volume for all stations from 5AM to 8AM on Jan 1th, 1997
  - Use value_per_day and five_min

<table>
<thead>
<tr>
<th>Table used</th>
<th>Total time(sec)</th>
<th>Database time(sec)</th>
<th>Network time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>value_per_day</td>
<td>95.628</td>
<td>94.829</td>
<td>0.799</td>
</tr>
<tr>
<td>five_min</td>
<td>296.487</td>
<td>295.296</td>
<td>1.191</td>
</tr>
</tbody>
</table>

- Database time $\approx 99\%$ of total time.
  - Database is the bottleneck.
Outline

- Introduction
- Requirement Specifications
- Software Design
- Bottleneck Analysis
- Performance Tuning
- Conclusion
Detailed Design Decisions

- Table design
- Index
- Pipelining
Conclusion of Experiments

- Indexing and pipelining reduce response time for traffic video
  - *Response time is 1.5 seconds.*

- Materialize partial lattice node
  - *Substantial improvement to data cube visualization.*

- Index improves performance.
  - *Significant improvement to traffic video.*
  - *Limited Improvement to data cube visualization.*
Constraints

- Servlet is not supported.
  - *Uses Perl/CGI instead.*
  - *Affects performance.*
- Database
  - *Only MySQL is available.*
  - *Limited free disk space.*
  - *Indexes are stored in B-tree*
  - *Primary indexes are not supported.*
Our Contribution

- Implemented a prototype high-performance visualization system.
  - **Performance**
    - For traffic video, performance issue is solved using indexes and pipelining.
    - For data cube visualization, database is the bottleneck.
    - Materialization of some lattice nodes improves performance.
  - Use a lattice framework to visualize traffic data.
Future Work

- Parallel processing
  - *Stripe*
- Improve the performance of data cube visualization
  - *Materialize the partial data cube.*
  - *Precompute frequently visited lattice nodes.*
- Support different granularity on all dimensions.
  - *Multi-resolution scheme*
- Consider concurrent users in traffic video
- Use other measures
  - *speed*
  - *productivity Q*
DT Matrix

- X axis: day of week, Y axis: time of day
- Slice for March on D-dimension
**DS Matrixx**

- **X axis:** day of week, **Y axis:** stations
- **Slice for January~March on D-dimension**
- **Slice for I-35W south on S-dimension**
- **Roll-up on D-dimension to day of week**
Pivot to D-T face.

I-35W South

I-94 East
Data Cube Operations

- **Pivot**: Rotate the cube to see a particular face.
- **Slice-dice**: Select some subset of the cube.
- **Roll-up**: Aggregate a dimension.
- **Drill-down**: Reverse of roll-up.
Software Design

- User Friendliness
  - *Web based application*
  - *Java applets provide interactive interface.*

- Portability
  - *Use Java as programming language*

- Performance
  - *Use database to store traffic data*
Benchmark Queries Design

- Two categories
  - Traffic video
    - Test 2 search conditions: date or (date, highways)
  - Data cube visualization
    - Assume performance is proportional to the number of rows accessed.
    - Test different size of data cube.
Benchmark Queries

<table>
<thead>
<tr>
<th>Num</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Get volume for all stations from 5AM to 8AM on Jan 1st, 1997.</td>
</tr>
<tr>
<td>Q2</td>
<td>Get average volume for all stations on I-35W south from 5AM to 8PM on Jan 1st, 1997.</td>
</tr>
<tr>
<td>Q3</td>
<td>Get average volume vs time of day on Jan 1st, 1997.</td>
</tr>
<tr>
<td>Q4</td>
<td>Get average volume for each time of day and each weekday from Jan 1st, 1997 to Jan 7th, 1997.</td>
</tr>
<tr>
<td>Q5</td>
<td>Get average volume for each time of day and each weekday from Jan 1st, 1997 to Jan 14th, 1997.</td>
</tr>
<tr>
<td>Q6</td>
<td>Get average volume for each time of day and each weekday from Jan 1st, 1997 to Jan 21th, 1997.</td>
</tr>
<tr>
<td>Q7</td>
<td>Get average volume for each time of day and each weekday for the month of Jan, 1997.</td>
</tr>
<tr>
<td>Q8</td>
<td>Get average volume for each time of day and each weekday for the month of Jan and Feb, 1997.</td>
</tr>
</tbody>
</table>

Table 2: Queries

- Q1~Q2 are for traffic video
  - *Tested on five_min table.*

- Q3~Q8 are for data cube visualization
  - *Tested on value_per_day table.*
Table Design

- Use relational database to facilitate data cube.
  - *five_min*

<table>
<thead>
<tr>
<th>sid</th>
<th>record_dttm</th>
<th>volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1997-01-01 06:00:00</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>1997-01-02 12:00:00</td>
<td>181</td>
</tr>
</tbody>
</table>

  - Primary key fields: sid and record_dttm
  - Data: Jan 1997

- *value_per_day*

<table>
<thead>
<tr>
<th>sid</th>
<th>record_date</th>
<th>data0</th>
<th>data1</th>
<th>...</th>
<th>data287</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1997-01-01</td>
<td>50</td>
<td>63</td>
<td>...</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>1997-01-02</td>
<td>30</td>
<td>46</td>
<td>...</td>
<td>60</td>
</tr>
</tbody>
</table>

  - Primary key fields: sid and record_date
  - data0: measure for 00:00AM, data1: 00:05AM, etc.
  - Data: Jan~Mar 1997

- *dt_cube*

<table>
<thead>
<tr>
<th>space</th>
<th>record_date</th>
<th>record_time</th>
<th>avg_volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>1997-01-01</td>
<td>00:00:00</td>
<td>31.92</td>
</tr>
<tr>
<td>I-35W</td>
<td>1997-01-31</td>
<td>09:05:00</td>
<td>222.08</td>
</tr>
<tr>
<td>I-35W-S</td>
<td>1997-01-01</td>
<td>09:55:00</td>
<td>208.11</td>
</tr>
</tbody>
</table>

  - Primary key fields: record_date and record_time
  - Summary table.
  - Data: Jan 1997
Materialize Lattice Nodes

- Dimension lattice:

```
    T D S
   /   \
  /     \
 T   S
   \
 S
```

- Data cube operations
  - **Roll-up**: Aggregate a dimension.
  - **Drill-down**: Reverse of roll-up.

- Examples:
  - **Materialize DT node.**

<table>
<thead>
<tr>
<th>Highway</th>
<th>Day of week</th>
<th>Hour of day</th>
<th>Avg Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-35W-S</td>
<td>Mon</td>
<td>6</td>
<td>254.43</td>
</tr>
<tr>
<td>I-35W-S</td>
<td>Mon</td>
<td>7</td>
<td>288.01</td>
</tr>
<tr>
<td>I-35W-S</td>
<td>Mon</td>
<td>8</td>
<td>267.04</td>
</tr>
<tr>
<td>I-35W-S</td>
<td>Fri</td>
<td>6</td>
<td>260.98</td>
</tr>
<tr>
<td>I-35W-S</td>
<td>Fri</td>
<td>7</td>
<td>323.55</td>
</tr>
<tr>
<td>I-35W-S</td>
<td>Fri</td>
<td>8</td>
<td>254.15</td>
</tr>
<tr>
<td>I-94-E</td>
<td>Mon</td>
<td>6</td>
<td>265.32</td>
</tr>
<tr>
<td>I-94-E</td>
<td>Mon</td>
<td>7</td>
<td>326.01</td>
</tr>
<tr>
<td>I-94-E</td>
<td>Mon</td>
<td>8</td>
<td>278.58</td>
</tr>
<tr>
<td>I-94-E</td>
<td>Fri</td>
<td>6</td>
<td>279.31</td>
</tr>
<tr>
<td>I-94-E</td>
<td>Fri</td>
<td>7</td>
<td>340.74</td>
</tr>
<tr>
<td>I-94-E</td>
<td>Fri</td>
<td>8</td>
<td>272.97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Highway</th>
<th>Day of week</th>
<th>Hour of day</th>
<th>Avg Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Mon</td>
<td>6</td>
<td>259.88</td>
</tr>
<tr>
<td>ALL</td>
<td>Mon</td>
<td>7</td>
<td>307.01</td>
</tr>
<tr>
<td>ALL</td>
<td>Mon</td>
<td>8</td>
<td>272.81</td>
</tr>
<tr>
<td>ALL</td>
<td>Fri</td>
<td>6</td>
<td>270.15</td>
</tr>
<tr>
<td>ALL</td>
<td>Fri</td>
<td>7</td>
<td>332.15</td>
</tr>
<tr>
<td>ALL</td>
<td>Fri</td>
<td>8</td>
<td>263.56</td>
</tr>
</tbody>
</table>

**Roll-up**
Performance Tuning: Different Table Design

- Test Q3~Q7 queries.
  - Use primary key index for all tables.
  - Q8 was not tested because of limitation of disk space.

<table>
<thead>
<tr>
<th>Query</th>
<th>value_per_day</th>
<th>five_min</th>
<th>dt_cube</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q3</td>
<td>0.278</td>
<td>7.89</td>
<td>0.080</td>
</tr>
<tr>
<td>Q4</td>
<td>1.78</td>
<td>34.52</td>
<td>0.219</td>
</tr>
<tr>
<td>Q5</td>
<td>3.592</td>
<td>58.45</td>
<td>0.249</td>
</tr>
<tr>
<td>Q6</td>
<td>5.403</td>
<td>81.87</td>
<td>0.263</td>
</tr>
<tr>
<td>Q7</td>
<td>6.776</td>
<td>116.01</td>
<td>0.335</td>
</tr>
</tbody>
</table>

Table 3: Impact of indexing on performance.

- Results:
  - Summary table is the fastest table.
    - 346 times faster than the slowest table five_min
Performance Tuning: Index

- Index provides a powerful way to speed up the retrieval of data

- Indexes in MySQL
  - Primary index: index built on primary key fields.
    - Records are not physically ordered.
  - Unique index: index built on fields which have unique value.
  - Normal index: index built on fields of a table. No constraints such as uniqueness.
Results of Benchmark Queries: Traffic Video

- Indexes
  - Table: five_min
  - Primary index: (record_dttm, sid)
  - Secondary index: record_dttm
  - Total number of rows: 5,649,984 (Jan 1997)

<table>
<thead>
<tr>
<th>Query</th>
<th>no index</th>
<th>primary index</th>
<th>secondary index</th>
<th>rows accessed</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>283.776</td>
<td>2.000</td>
<td>2.090</td>
<td>22860</td>
<td>0.4%</td>
</tr>
<tr>
<td>Q2</td>
<td>313.359</td>
<td>0.638</td>
<td>1.057</td>
<td>4428</td>
<td>0.08%</td>
</tr>
</tbody>
</table>

Table 4: Impact of indexing on performance.

- When displaying all stations, speed-up~140
- When displaying all highways, speed-up~491
Results of Benchmark Queries: Data Cube Visualization

- Indexes
  - Table: value_per_day
  - Primary index: (record_date, sid)
  - Secondary index: record_date
  - Total number of rows: 171122 (Jan~Mar 1997)

<table>
<thead>
<tr>
<th>Query</th>
<th>no index</th>
<th>primary index</th>
<th>secondary index</th>
<th>rows accessed</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q3</td>
<td>1.517</td>
<td>0.278</td>
<td>0.320</td>
<td>1897</td>
<td>1.1%</td>
</tr>
<tr>
<td>Q4</td>
<td>2.67</td>
<td>1.78</td>
<td>1.841</td>
<td>13331</td>
<td>7.79%</td>
</tr>
<tr>
<td>Q5</td>
<td>3.708</td>
<td>3.592</td>
<td>3.615</td>
<td>26637</td>
<td>15.57%</td>
</tr>
<tr>
<td>Q6</td>
<td>4.944</td>
<td>5.403</td>
<td>5.354</td>
<td>39898</td>
<td>23.32%</td>
</tr>
<tr>
<td>Q7</td>
<td>6.627</td>
<td>6.776</td>
<td>6.819</td>
<td>58964</td>
<td>34.46%</td>
</tr>
<tr>
<td>Q8</td>
<td>11.772</td>
<td>11.966</td>
<td>11.849</td>
<td>112191</td>
<td>65.56%</td>
</tr>
</tbody>
</table>

Table 5: Results of Q3~Q8
Results of Benchmark Queries: Data Cube Visualization

- Indexes help when accessing fewer than 15% of rows (Point A).
  - Speed-up ~ 7
- Between 15% and 30%, no index is faster
- After 30% (Point B), database always does a table scan
Performance Tuning: Response Time

- Query: Video visualization
- Original design: Read all data, then display
  - Avoids jitter.
Performance Tuning: Pipelining

- Each transaction transfers data for a snapshot
- Use buffer to store some snapshots in advance.
- Two threads work concurrently.
  - Thread 1: displays video
  - Thread 2: receives data from CGI
Response Time of Traffic Video: Improved

- **Buffer size**=3
  - **Network is fast**
    - GUI consumes 2 snapshots per second.
    - CGI produces more than 10 snapshots per second.
  - **Database performance improved after using index**
    - Database produces about 18 snapshots per second.

- **Results**

<table>
<thead>
<tr>
<th>Hours</th>
<th>No Pipelining</th>
<th>Pipelining (five_min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5AM ~ 8AM</td>
<td>5.067</td>
<td>1.512</td>
</tr>
<tr>
<td>5AM ~ 10AM</td>
<td>6.579</td>
<td>1.523</td>
</tr>
<tr>
<td>5AM ~ 12AM</td>
<td>8.482</td>
<td>1.512</td>
</tr>
<tr>
<td>5AM ~ 2PM</td>
<td>11.617</td>
<td>1.562</td>
</tr>
<tr>
<td>5AM ~ 4PM</td>
<td>13.509</td>
<td>1.512</td>
</tr>
<tr>
<td>5AM ~ 6PM</td>
<td>16.684</td>
<td>1.512</td>
</tr>
</tbody>
</table>

Table 6: Response Time

- **Use of pipelining and index reduces the response time to 1.5 seconds.**