A Framework for Scalable Data Analysis and Model Aggregation for Public Bus Systems

1 Introduction

2 Framework for Modeling the Bus Network

3 Experiments

4 Conclusions
Introduction
Introduction

Efficient urban mobility

- Individual Transport?
  - Cars, Uber, taxis
  - Congestion at peak hours
- Rail transport systems?
  - Efficient but high cost
- Bus transport systems?
  - Reasonable cost
  - Delays
Better understand the behavior of bus system
  • Many aspects of the bus system
  • Only a few efforts using those aspects together
Proposal of a framework to:
  • Maintain the state of the network
  • Predict the travel time between arbitrary points in a bus route
  • Understand the evolution of the link states during the day
  • Model the occurrence of bus bunching
Framework for Modeling the Bus Network
Dask: framework for distributed computing in Python

- Creation of tasks for execution on different machines
- Integrated with Python frameworks, such as Pandas and Scikit-learn

MongoDB: storage of non-relational data, providing more flexibility

- Permits information sharing between models
- Can be replicated if required
Estimation Models

- Bus Position Model
  - Travelled distance on the bus route
- Graph Model
  - Based on GTFS (General Transit Feed Specification) data

Prediction Models

- Mean Travel Time Model: (1 per hour)
  - Historical average for each link (1977 links)
  - 7 Months data, restricted on Weekday and Hour

- $K$ Travel Time Model (1 per 10 minutes):
  - Average from the Latest $K (=3)$ travels on each link

- Combined Model:
  - Weighted Average from MTT ($wgt = 1$) and KTT ($wgt = 2$) Models
Experiments
We considered scenarios with 6 to 72 workers

- First experiment: Execution time
- Second Experiment: total travel time prediction accuracy
Experiments

First experiment: Execution time

- Increasing # of workers reduces total processing time up to a certain point
Experiments

First experiment: Execution time

CPU Usage

Total CPU x MongoDB CPU Usage (6 proc)

CPU Usage (%)

Time (s)

Total CPU x MongoDB CPU Usage (12 proc)

CPU Usage (%)

Time (s)
Experiments

First experiment: Execution time

- Since each worker access the DB concurrently, it becomes the bottleneck
## Experiments

### Second Experiment: total travel time prediction accuracy

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RMSE (min.)</th>
<th>MAE (min.)</th>
<th>MAPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTT</td>
<td>9.21</td>
<td>6.70</td>
<td>11.12</td>
</tr>
<tr>
<td>KTT</td>
<td>12.05</td>
<td>8.37</td>
<td>13.95</td>
</tr>
<tr>
<td>Combined</td>
<td>8.97</td>
<td>6.68</td>
<td>11.22</td>
</tr>
</tbody>
</table>

**Table:** Errors for full travel time predictions

KTT uses less data and is faster, but it is more sensitive to data errors and travel deviations.
Conclusions
• Public bus systems are difficult to model and predict
• Important to combine different models in the prediction
  • Historical travel times, link state status and evolution, bus headways, etc.
• Proposed framework combines those models
• Bottleneck on the shared database
• Next steps
Obrigada

Fonte: State University of NY (2000)