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

Mayurí Annerose Morais, Raphael Y. de Camargo

Mathematics, Computation and Cognition Center
Universidade Federal do ABC

June 26, 2019

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

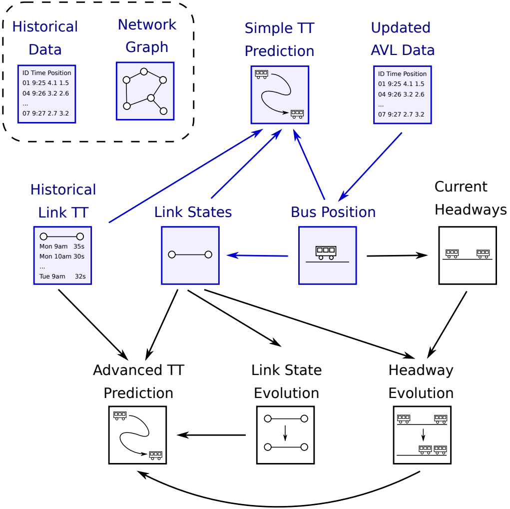
Better understand the behavior of bus system

- 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

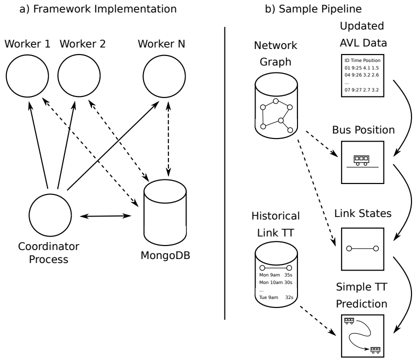

Historical Data
ID Time Position
01 9:25 4.1 1.5
04 9:26 3.2 2.6
...
07 9:27 2.7 3.2
Network Graph
Simple TT Prediction
Updated AVL Data
ID Time Position
01 9:25 4.1 1.5
04 9:26 3.2 2.6
...
07 9:27 2.7 3.2
Historical Link TT
Mon 9am 35s
Mon 10am 30s
...
Tue 9am 32s
Link States
Bus Position
Current Headways
Advanced TT Prediction
Link State Evolution
Headway Evolution

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

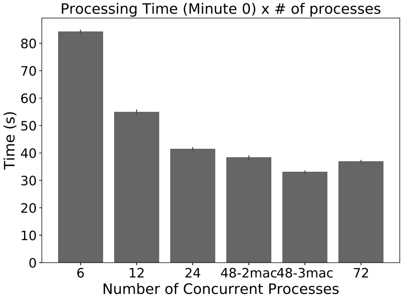
- Mean Travel Time Model: (1 per hour)
  - Historical average for each link (1977 links)
  - 7 Months data, restricted on Weekday and Hour
- K Travel Time Model (1 per 10 minutes):
  - Average from the Latest $K (= 3)$ travels on each link
- Combined Model:
  - Weighted Average from MTT $(wgt = 1)$ and KTT $(wgt = 2)$ Models

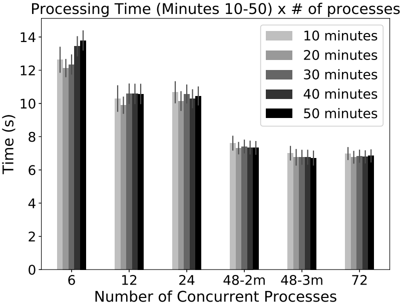
Experiments

We considered scenarios with 6 to 72 workers

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

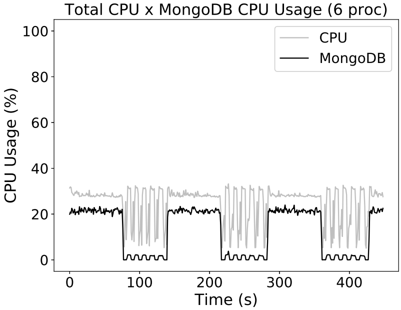
Processing Time

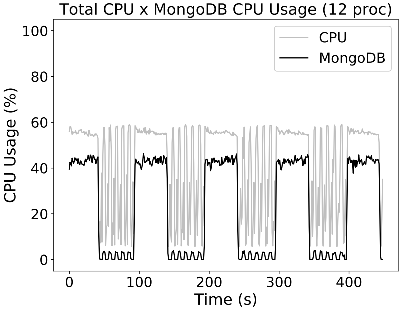




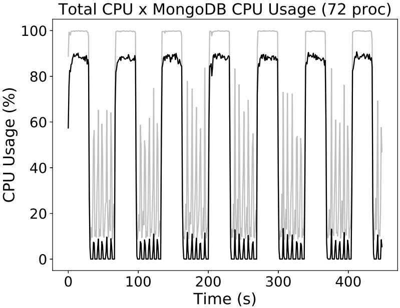
- Increasing # of workers reduces total processing time up to a certain point

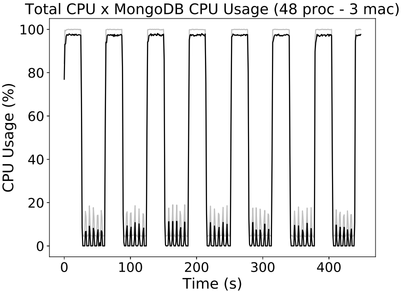
CPU Usage


Total CPU x MongoDB CPU Usage (6 proc)
CPU
MongoDB
CPU Usage (%)
Time (s)
0
20
40
60
80
100
0
100
200
300
400


Total CPU x MongoDB CPU Usage (12 proc)
CPU
MongoDB
CPU Usage (%)
Time (s)
0
20
40
60
80
100
0
100
200
300
400

CPU Usage





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

Algorithm	RMSE (min.)	MAE (min.)	MAPE (%)
MTT	9.21	6.70	11.12
KTT	12.05	8.37	13.95
Combined	8.97	6.68	11.22

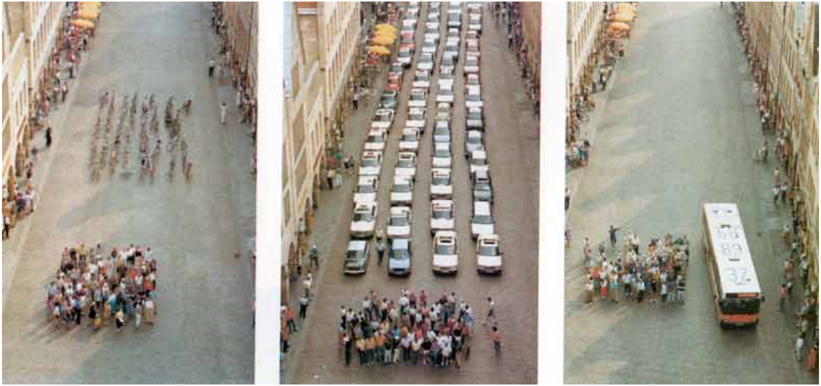
Table: Errors for full travel time predictions

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)