How to Speed up Traffic Simulation and Routing Calculation

Experiences using Simplified Traffic Flow Models, Multi-Core Programming Interface, and Cloud Computing Architecture

TRANSIMS: Applications and Development Workshop
April 8–9, 2010

Prepared by
Dr. Xuesong Zhou and Hao Lei (Ph.D. student)
Univ. of Utah

Motivations

- Existing technical barriers: (based on DTA user survey, TRB network modeling committee)
  - Require too many input data: 47%
  - Take too long to run: 35%
  - Model is unclear: 35%

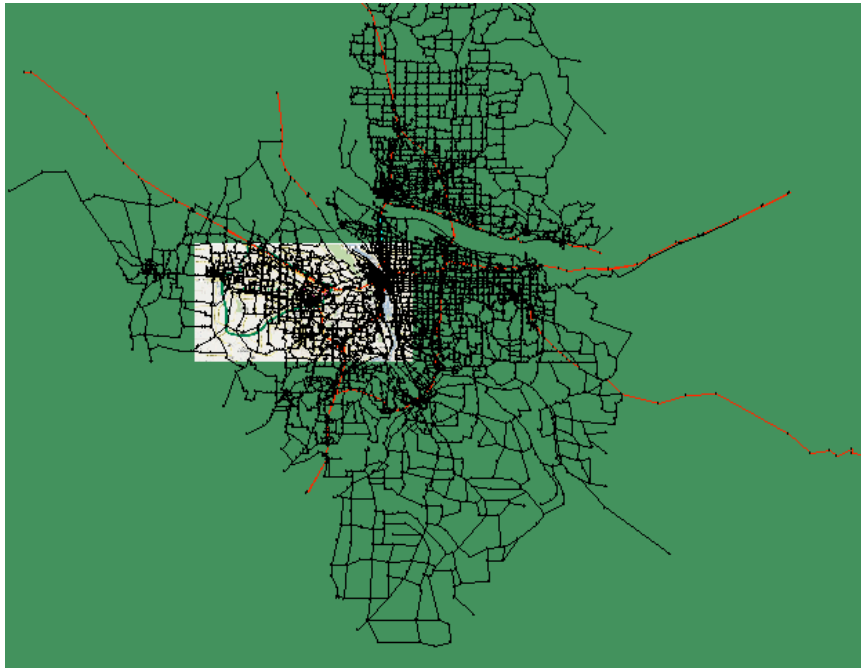
Our goals

- Simplified data input from static traffic assignment
- Use parallel computing capability, simplified routing and simulation
- Open-source

Other Goals for Developing DTALite

- Leverage free and user-friendly GUI: NEXTA for DYNASMART and TRANSIMS
  - Allow users to quickly convert data from existing static assignment packages
  - Interactive network editor
  - Visualization platform for time-dependent link MOE, path and vehicle animation
- Provide free assignment tool for undergraduate students learning 4-step process: seeing is believing
- Clean source code with open-source GPL license

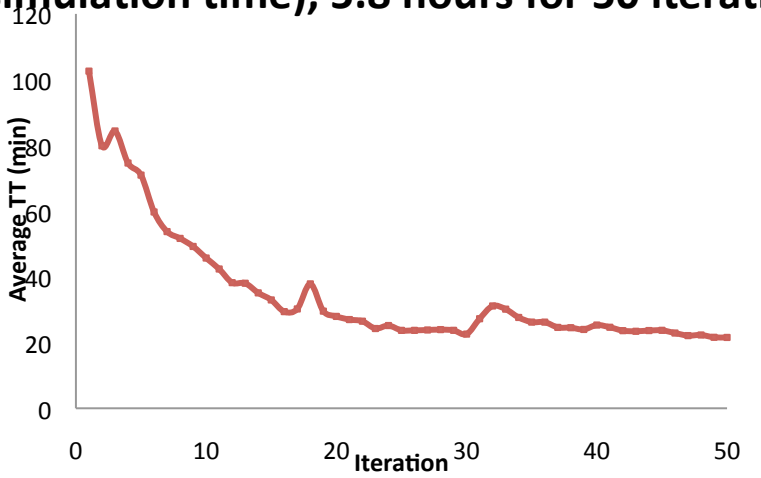
Preliminary Testing Results



Portland Network
(from Portland Metro VISUM network)

- **# of Zones = 2,013**
- **# of Nodes = 9,905**
- **# of Links = 22,748**
- **# of Vehicles = 1.2 million**
- **4 hour Demand interval: 15:00-19:00**

CPU time: 7 min for each iteration (5-6 hour simulation time), 5.8 hours for 50 iterations



Intel Core 2 Duo CPU
(4 processors)

Component 0: Network Representation

- Nodes
  - Control type
  - Connector type
- Links
  - # of lanes, length
  - link type
  - Free-flow Speed
  - Actual reduced capacity (v/c in BPR function)

  - No signal timing information

  - No # of left turn or right turn bays
  - No turning movement restrictions
  - Do not distinguish different approaches of yield signs
  - Do not distinguish protected vs. unprotected left-turns

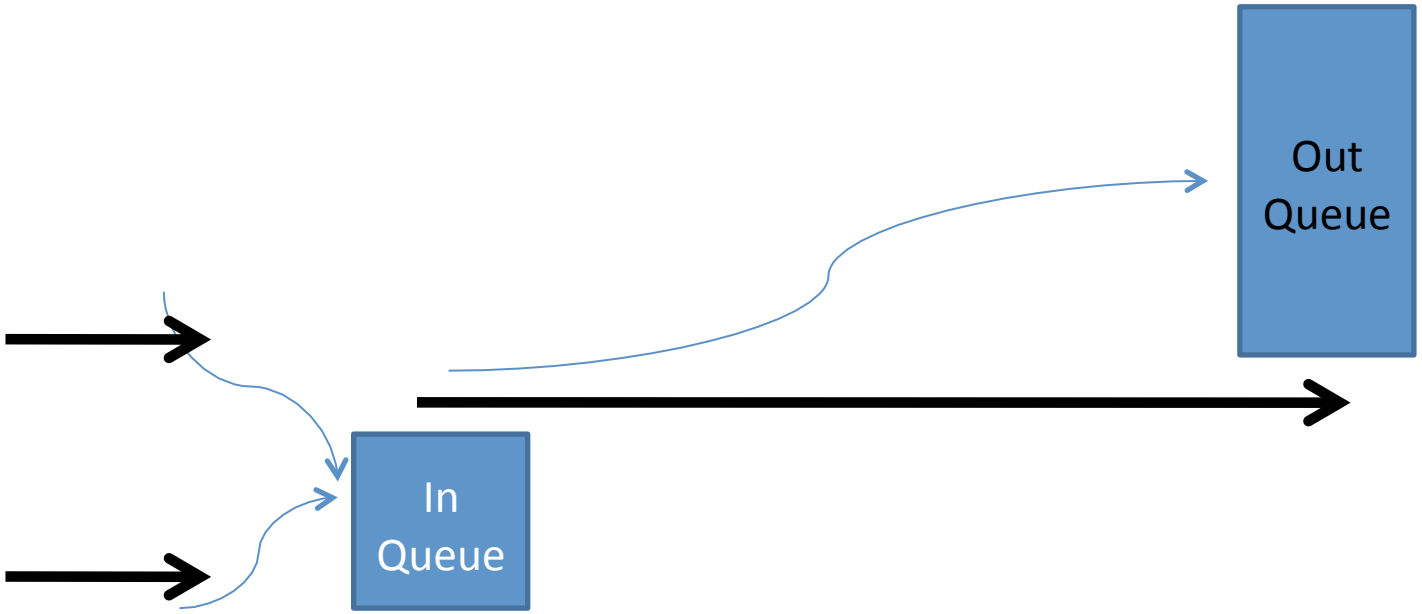
Component 1: Traffic Flow Simulation

- Use average link capacity depending on downstream node type
  - Do not simulate signal control logics at each simulation interval
  - Do not consider impact of flow from conflicting directions at stop or yield sign
- Simple point queue model that tracks shockwave
  - Newell's approach: theoretically equivalent to cell transmission model (CTM)
    - CTM's variables are density, incoming and outgoing flows and speed
    - Newell's variables are incoming and outgoing cumulative flow counts, from which the density and flows can be derived. We do not have direct speed measurements and link travel times are calculated from vehicle trajectories (with arrival times at upstream node and departure time at downstream node)

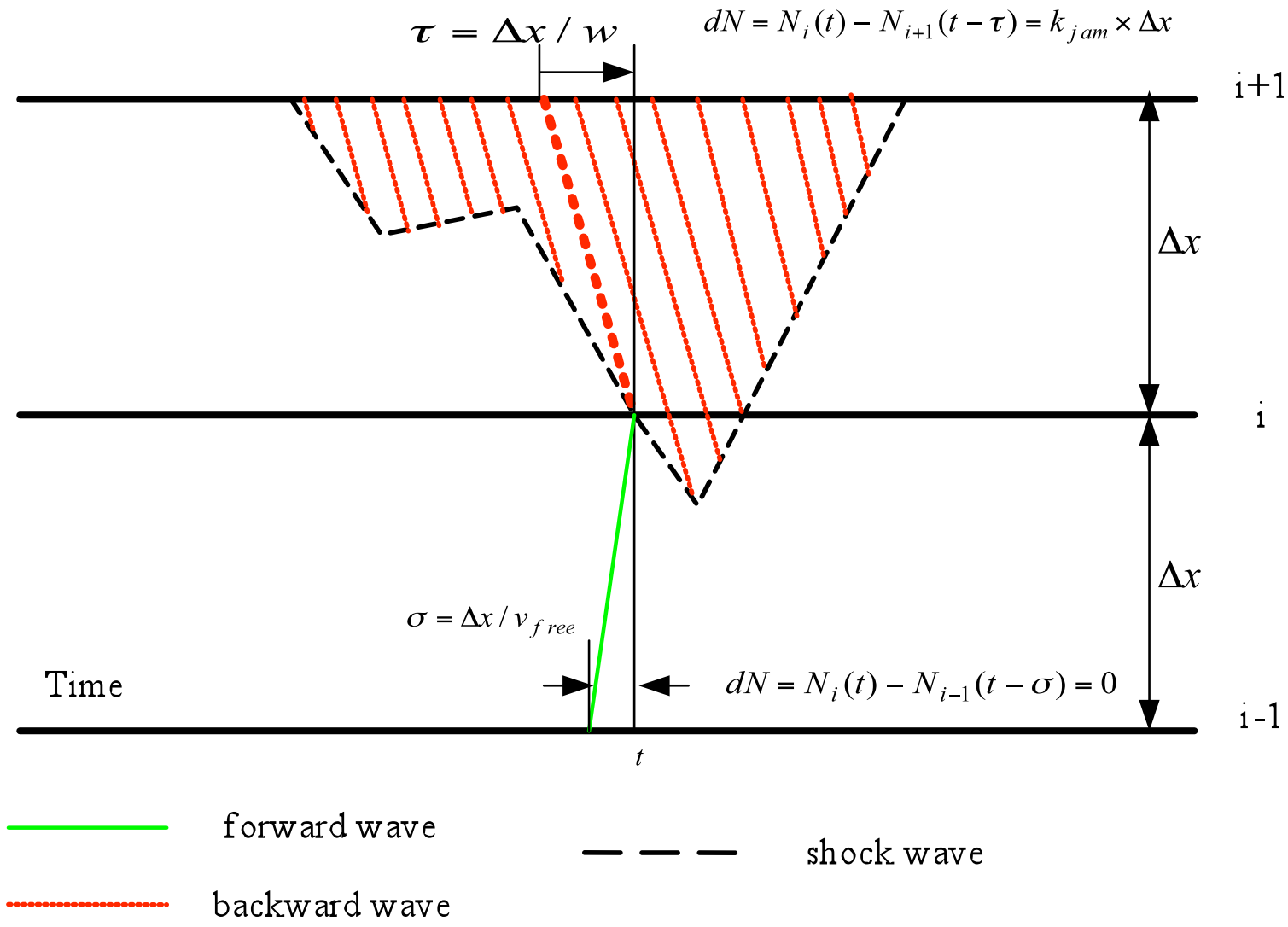
Simulation Logic

  - Can be viewed as pseudo event-based simulation and we do not simulate how a vehicle moves inside the link

- Vehicle is moved into a link-in queue at time t_a
- Calculate time entering the link-out queue as t_a+FFTT (free-flow travel time)
- If the current simulation time t equals to or is later than t_a+FFTT, if the link out capacity is still available, move this vehicle to the next link, otherwise stay in the link out queue



Available capacity at every simulation interval



References: Newell, G. F., 1993a. A simplified theory on kinematic waves in highway traffic, part I: general theory. Transportation Research Part B, Vol. 27(4), pp. 281-287.

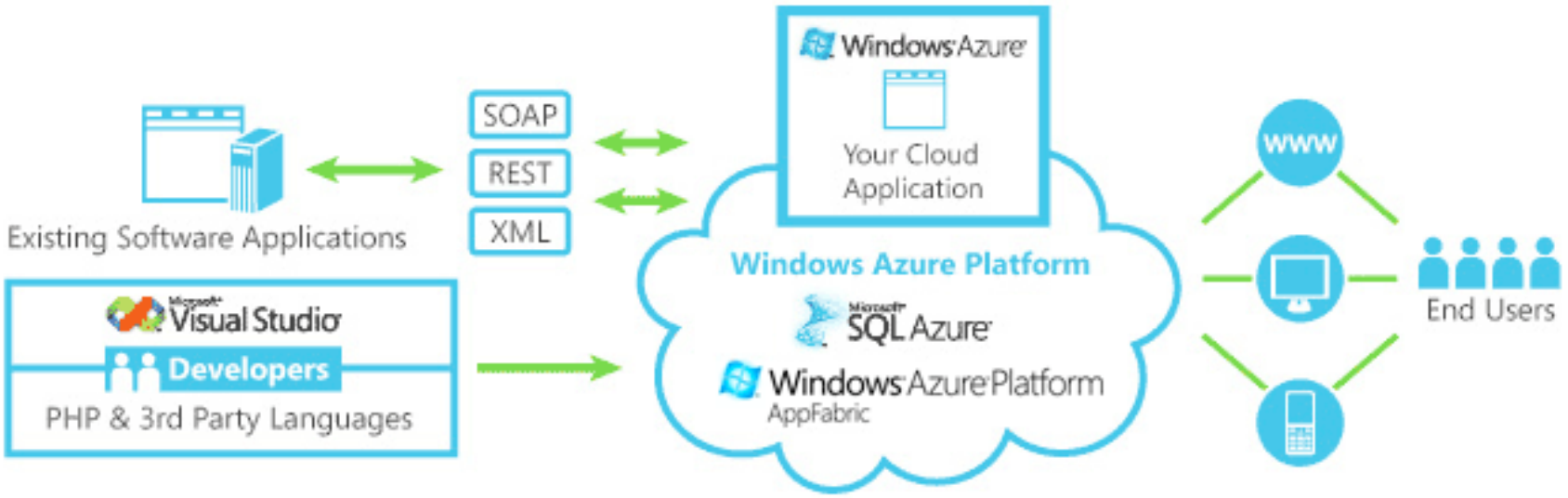
Component 2: Simplified Shortest Path Algorithm

- C++ implementation
  - Use standard template library (STL) for complicating data structure (e.g. multi-dimensional vector, list, map (hash table)
  - Customized efficient structure for shortest path algorithm
- Multiple processor-oriented
  - OpenMP technique for using multiple processors
- Shortest path algorithm
  - Label correcting with deque
  - No turning-movement delay
  - Single departure time, origin-based.
    - The algorithm is called iteratively for each departure time to calculate shortest paths for all departure times

Why Cloud Computing?

- Reduce the effort and costs of IT management
- Bring your ideas to market faster and pay as you go
- Consume computing resources ONLY when the needs arise.

Overall Cloud Computing Architecture



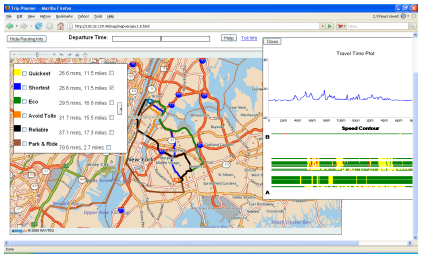
Routing Engine

Travel time database

XML
Web service

Internet-based OS

Route interface



http://www.microsoft.com/windowsazure/products/

What if we want to deploy TRANSIMS routing engine to a cloud computing platform…

- How much it costs?
- How many users we can support simultaneously ?
- How many CPUs the cloud computing OS can use?
  - Individual instance (hided behind physical hardware/ network configurations)
  - Dedicated computing power and memory for each instance
- Response time
- Database support
- Data communication overhead

Summary Costs

Windows Azure Instances

Number of Windows Azure instances:*	5
Average use hours per day:*	12.0
Average use days per year:*	180

Total		Initial / Month	Initial / Year
		$220.97	$2,652

Detailed Cost Inventory

	Usage**			Rate***		Initial / Month	Initial / Year
Windows Azure	Computing	900	hrs / month	$0.12	/ hour	$108.00	$1,296
	Storage (unstructured) (GB)	200.0	GB	$0.15	/ GB / month	$30.00	$360
	Storage transactions (tx)	2,000	10K tx / month	$0.01	/10K tx	$20.00	$240
SQL Azure	1GB databases	1	units	$9.99	/ month	$9.99	$120
	10GB databases	0	units	$99.99	/ month	$0.00	$0
AppFabric	Service Bus connections	2	connections / month	$3.99	/ connection	$7.98	$96
	Access Control transactions (tx)	0	100K tx / month	$1.99	/ 100K tx	$0.00	$0
Bandwidth	Inbound *	1.00	avg. GB / hr	$0.10	/ GB hour	$18.00	$216
	Outbound *	1.00	avg. GB / hr	$0.15	/ GB hour	$27.00	$324

Platform Selection Considerations

- Can we use/port my legacy C++ code?
  - Based on .Net framework and right now only support C# and VB .Net
  - Can be compiled into DLL, thus usable in .Net framework
- Do I really need to put everything in a database?
  - Network data and real-time traffic information
  - Easy to manage and update
  - Robust
- MySQL? Vs. MS SQL server? SQLite?
  - MS SQL Azure, fully integrated with MS Azure platform, reduced development time and efforts

Comparison with Existing Cloud Services

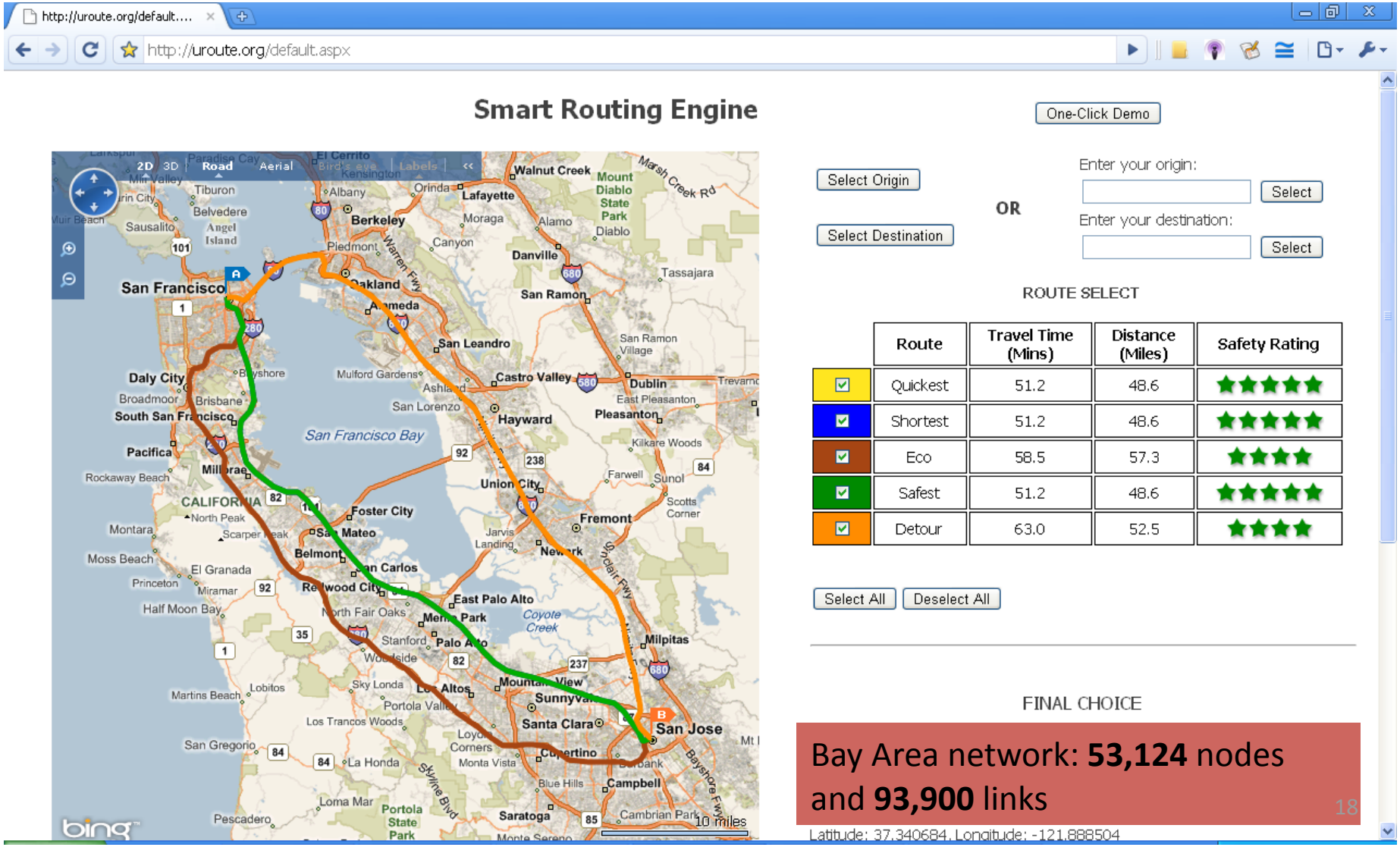
Feature	Microsoft	Amazon	Google
Availability	Yes	Yes, commercially available	In public beta
Computing Architecture	You provide .NET code for front-end and back-end servers which Microsoft then runs on Windows 2008 virtual machines according to your environment specifications (how many machines of each kind you need, and so on.)	Elastic Compute Cloud (EC2) allows you to upload your virtual machine images to the infrastructure and gives you APIs to instantiate and manage them.	You write your web application in Python or Django with a specific set of limitations set by Google and submit the application code to them.
Load balancing	Yes	Yes	Yes
Storage	Yes: application storage and SQL services	Yes: Simple Storage Service (S3) and SimpleDB	Yes: database Datastore APIs
Tied to the vendor datacenter	Yes	Yes	Yes
Development tools	Yes, integration into Visual Studio, support for any **.NET** languages,	Not applicable. Amazon simply runs your virtual machines and does not care which development platform you are using on top of the base OS.	Yes, have basic editing, local simulation, and deployment tools. Language selection limited to **Python and Django**. Application-level tools such as Google Web Toolkit (GWT) do not seem to have any integration with Google App Engine.

Why MS Azure?

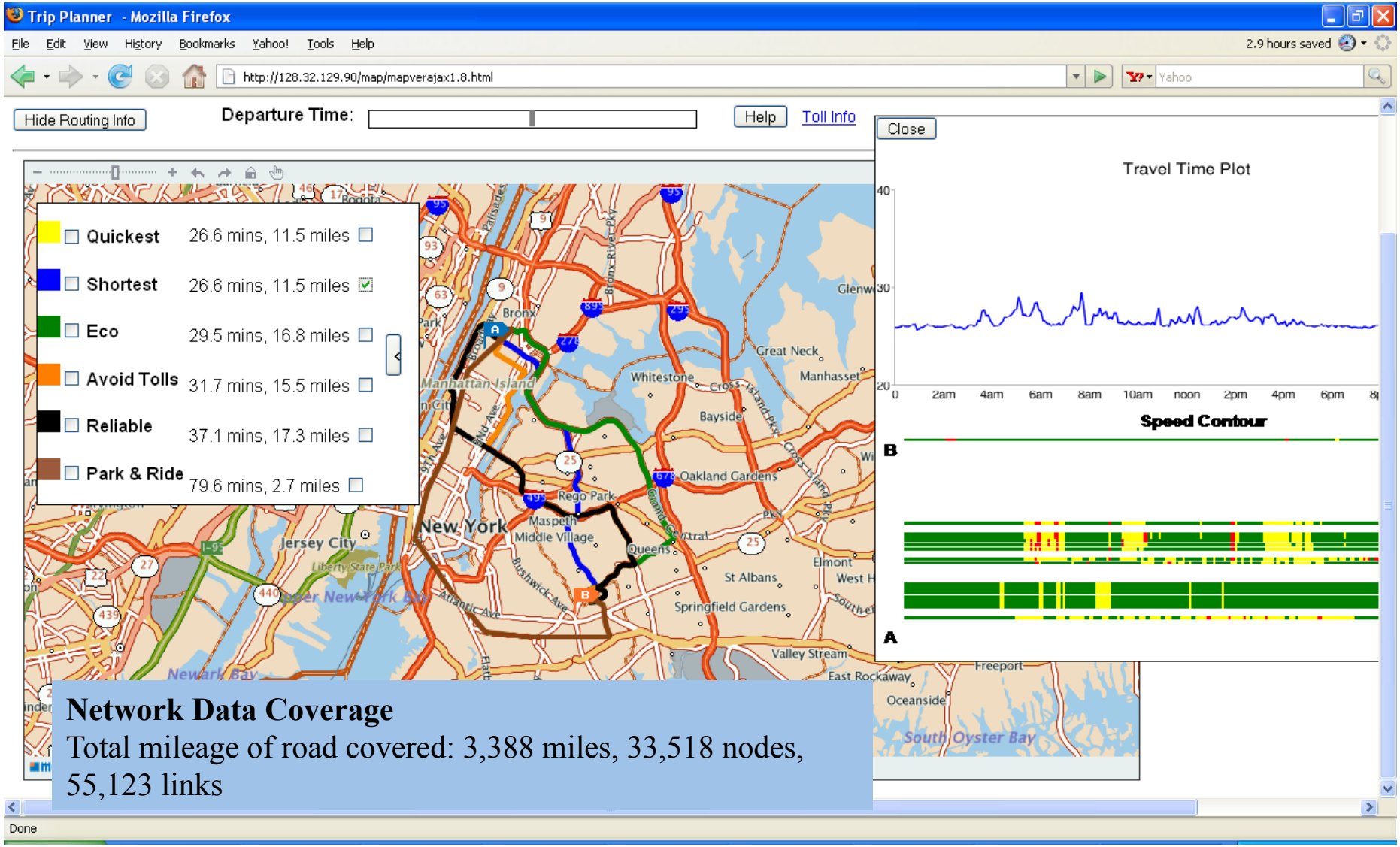
- Full compatible with the existing development tools (Visual Studio 2008)
- Easy debugging – develop and debug locally before deployment
- Easy deployment – deployment tools integrated with Visual Studio
- Minimized porting efforts

Everything in the Cloud

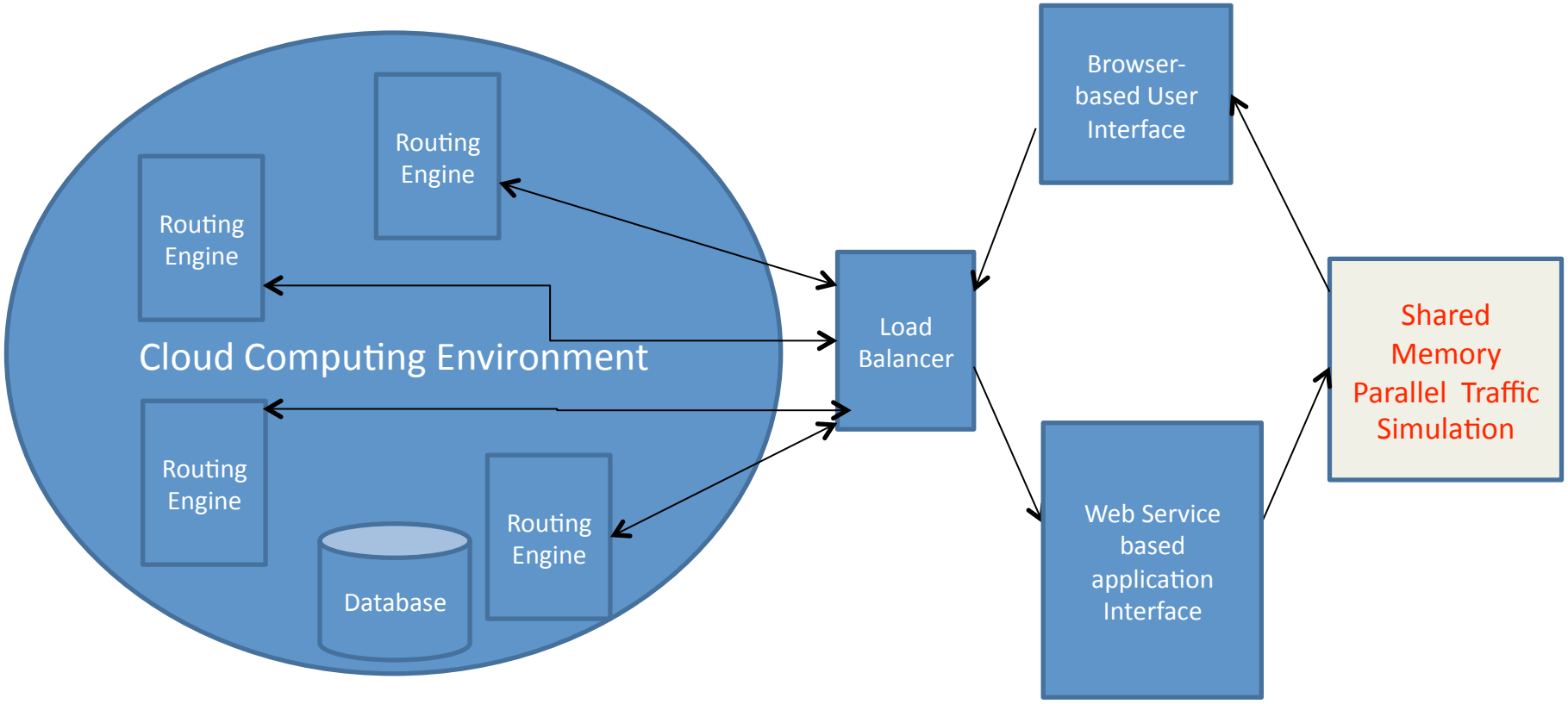
http://uroute.org/default.aspx



Other Interfaces through MapStraction: Open Javascript API -> Google, Yahoo, Map 24

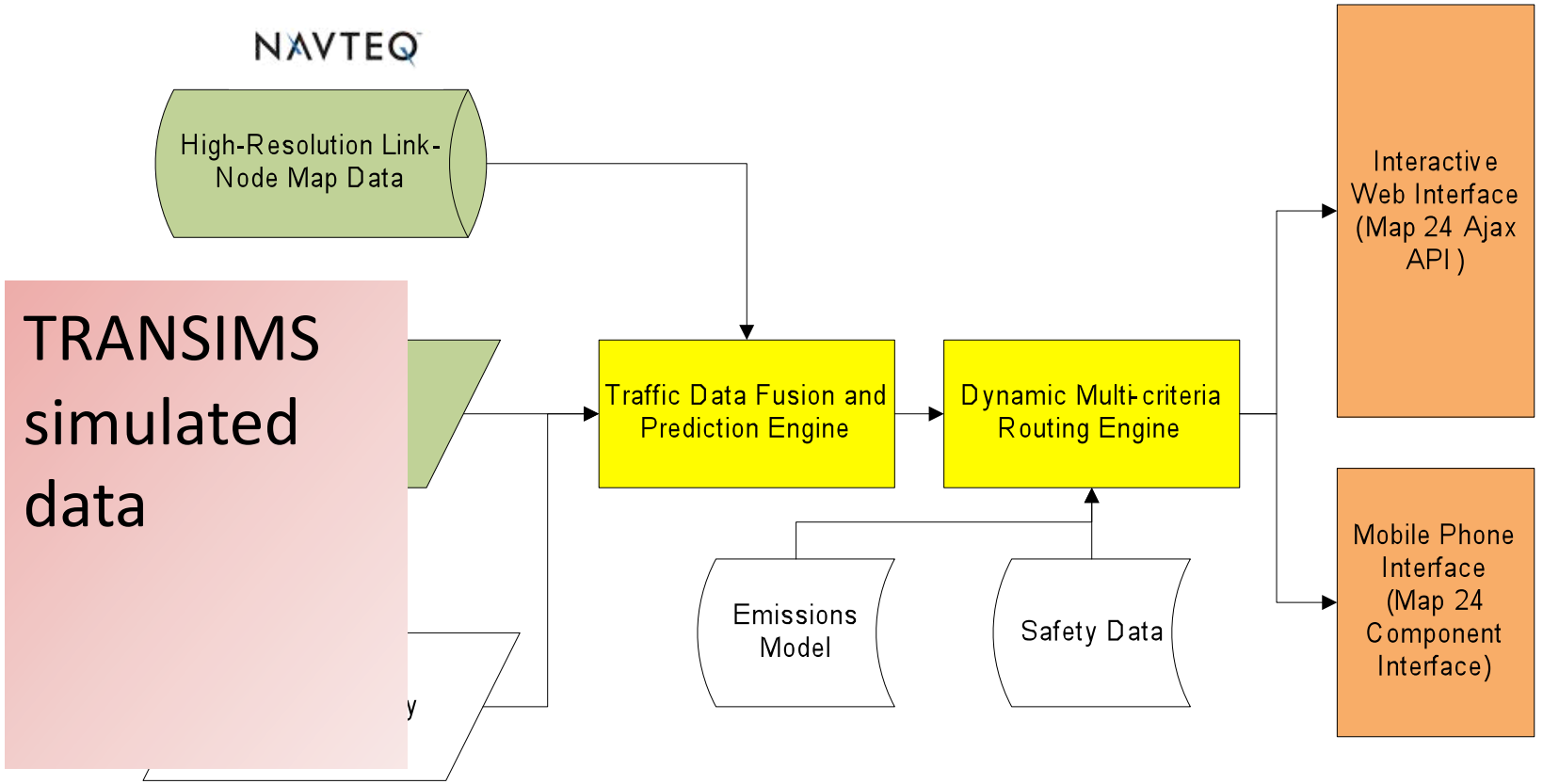


System Architecture



Multiple routing instances inside cloud computing environment – parallel computing

Data Flow


NAVTEQ
High-Resolution Link-Node Map Data
TRANSIMS simulated data
Traffic Data Fusion and Prediction Engine
Dynamic Multi-criteria Routing Engine
Emissions Model
Safety Data
Interactive Web Interface (Map 24 Ajax API)
Mobile Phone Interface (Map 24 Component Interface)

Main Features

- Built on MapStraction Javascript API
  - Can use Google Map, Yahoo Map or Map 24 as background image provider
- Integrate with Navteq Traffic historical and real-time sensor data or other data sources (e.g. PeMS)
- Consider personalized routing criteria:
  - **Travel time, distance, toll cost, fuel cost, emissions, safety and reliability**
- Dynamic routes using real-time traffic data
  - Time-dependent travel time profile for encouraging route, **departure time and mode switch**
- Speed contour for system operators to optimize traffic flow management

Testing Results on Cloud Computing Platform

- Small 1.6 GHz 1.75 GB
  500 one-to-one shortest path request: 40.3s
  - **0.08 sec per path per CPU core**

  Medium 2 x 1.6 GHz 3.5 GB

  Large 4 x 1.6 GHz 7 GB

  Extra large 8 x 1.6 GHz 14 GB

How Much Time It takes?

- 400,000 trips (peak hour, Portland)
  - 1,000 trips -> 80 seconds
- 400*80/60 min= 533 min = 8.8 hour

- Extra large Mode 8 x 1.6 GHz 14 GB
- 8.8 hours/8 = 1.1 hours

Web Service Performance Test

- **Response time**: How fast the web service is running for normal requests
- **Load test**: How the web service performs in a high traffic condition (maximum loading condition)
- **Stress test**: How the web service responses in an over-loaded environment

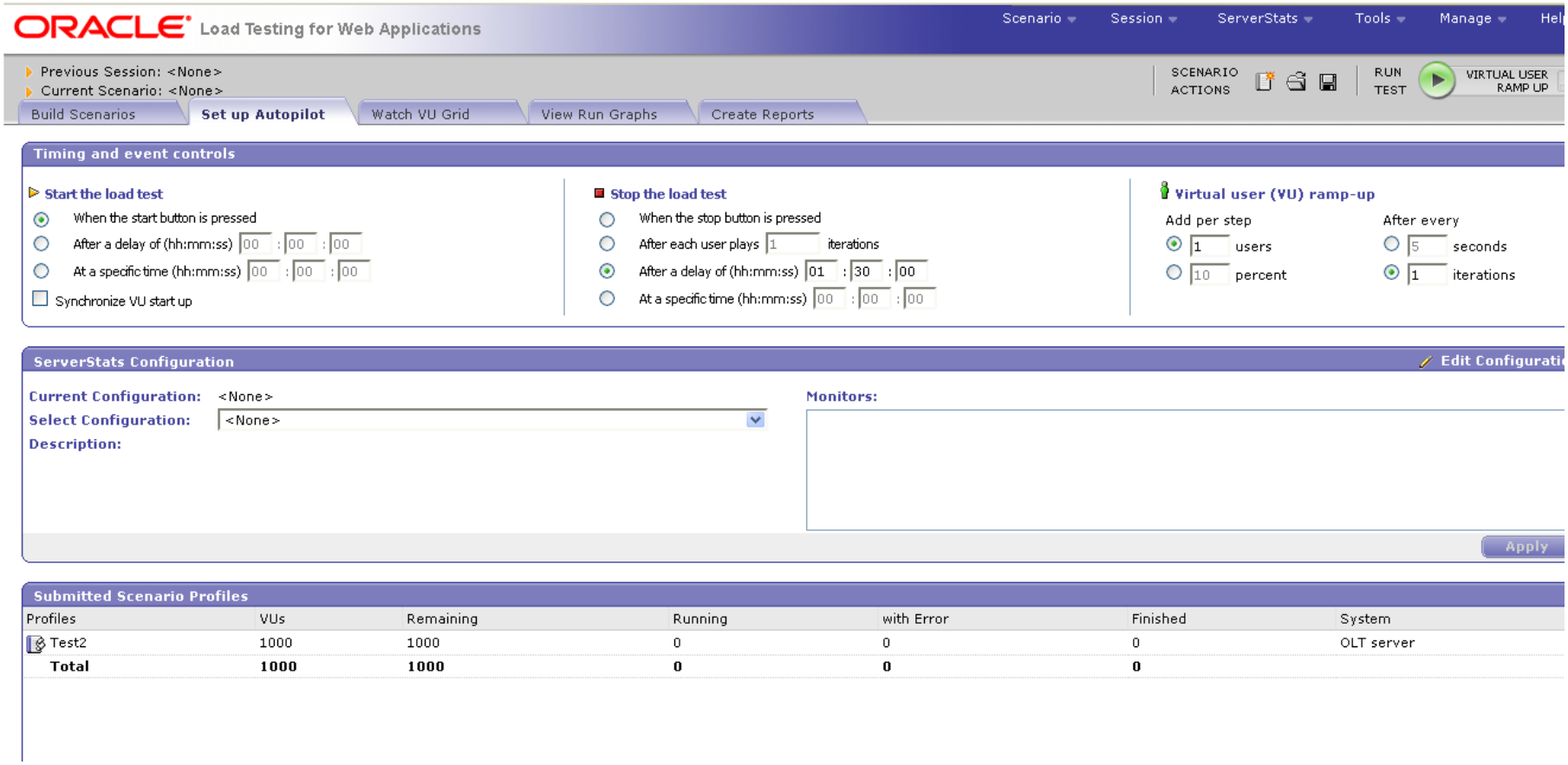
Key Words

- **Virtual User (VU):** Used to simulate the clients of the web service, usually works iteratively to simulate continuous requests.
- **Transaction:** A response the virtual user received from server side

System Capacity

- User sends request every 30 seconds
- Assume the user requests come in as Poisson distribution
- System capacity = (# of request handled per second) x (30 seconds)

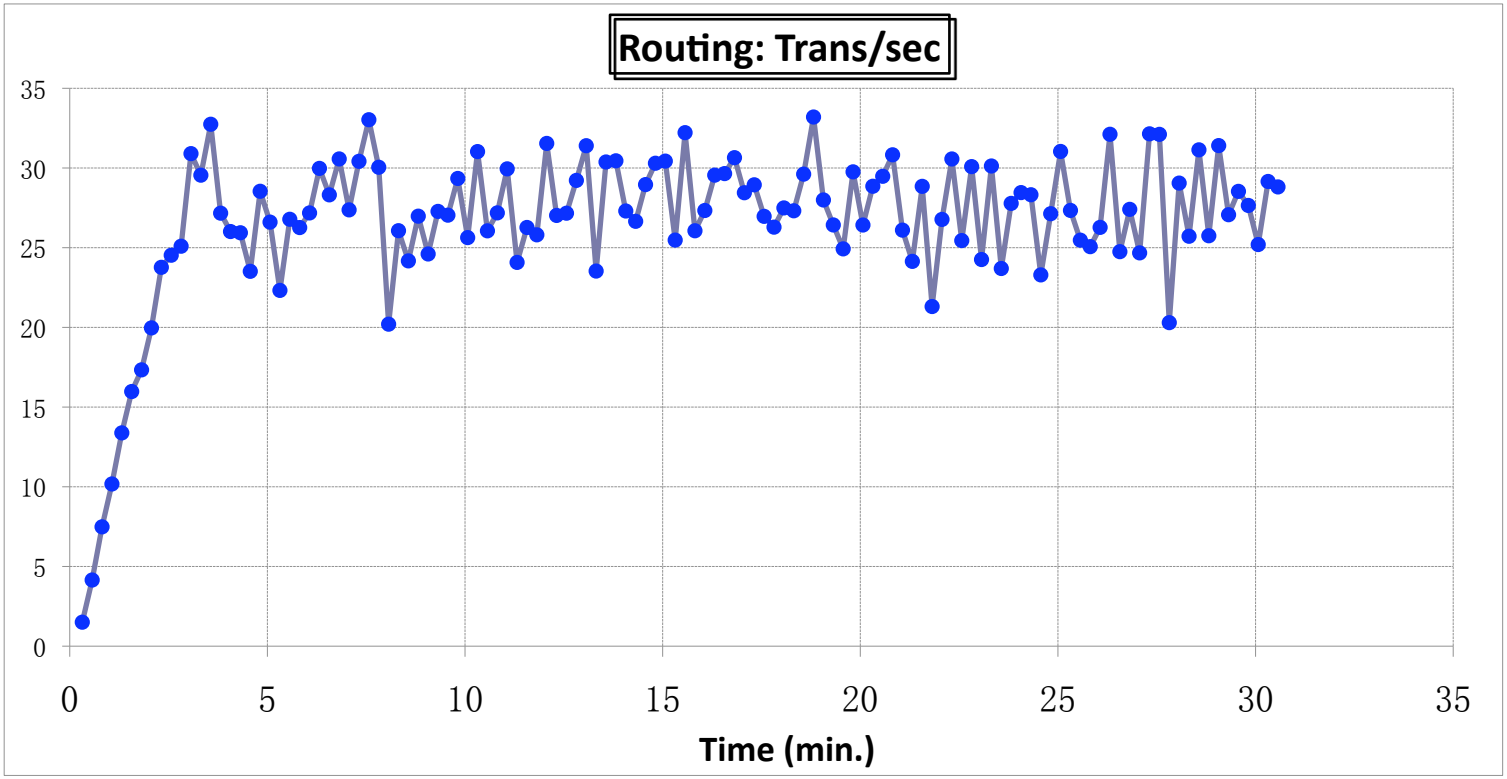
Oracle Testing Application Testing Suite



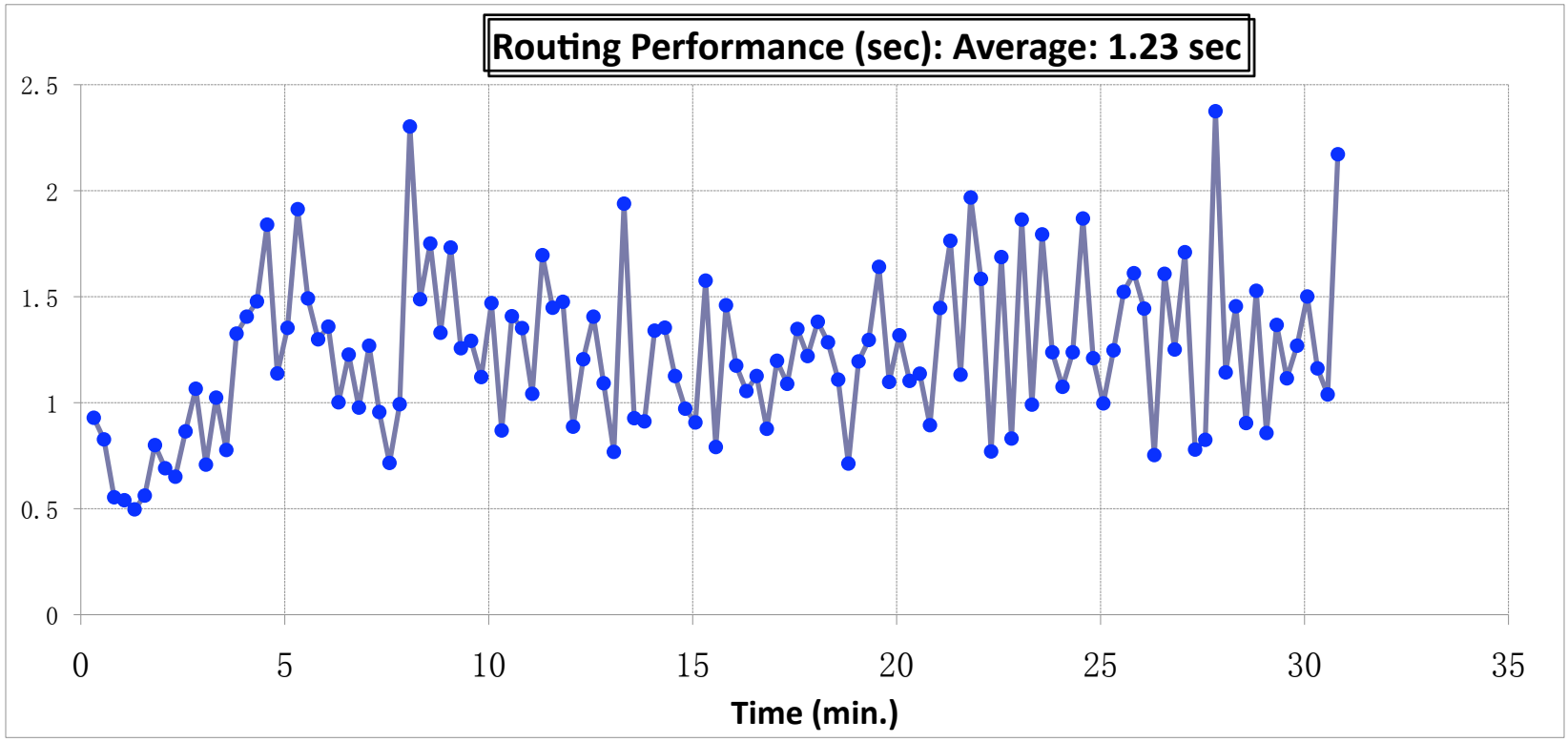
Routing Service

- Testing Environment:
  - CPU: Intel Core 2 Due 1.8 GHz
  - Memory: 2 GB
- Testing Scenario:
  - 90 VUs
  - 2 sec iteration delay
- Testing Results:
  - # of request can be handled: 25 – 30 per second
  - System capacity: 750 – 900 users if a user makes a request every 30 seconds

Transition/second

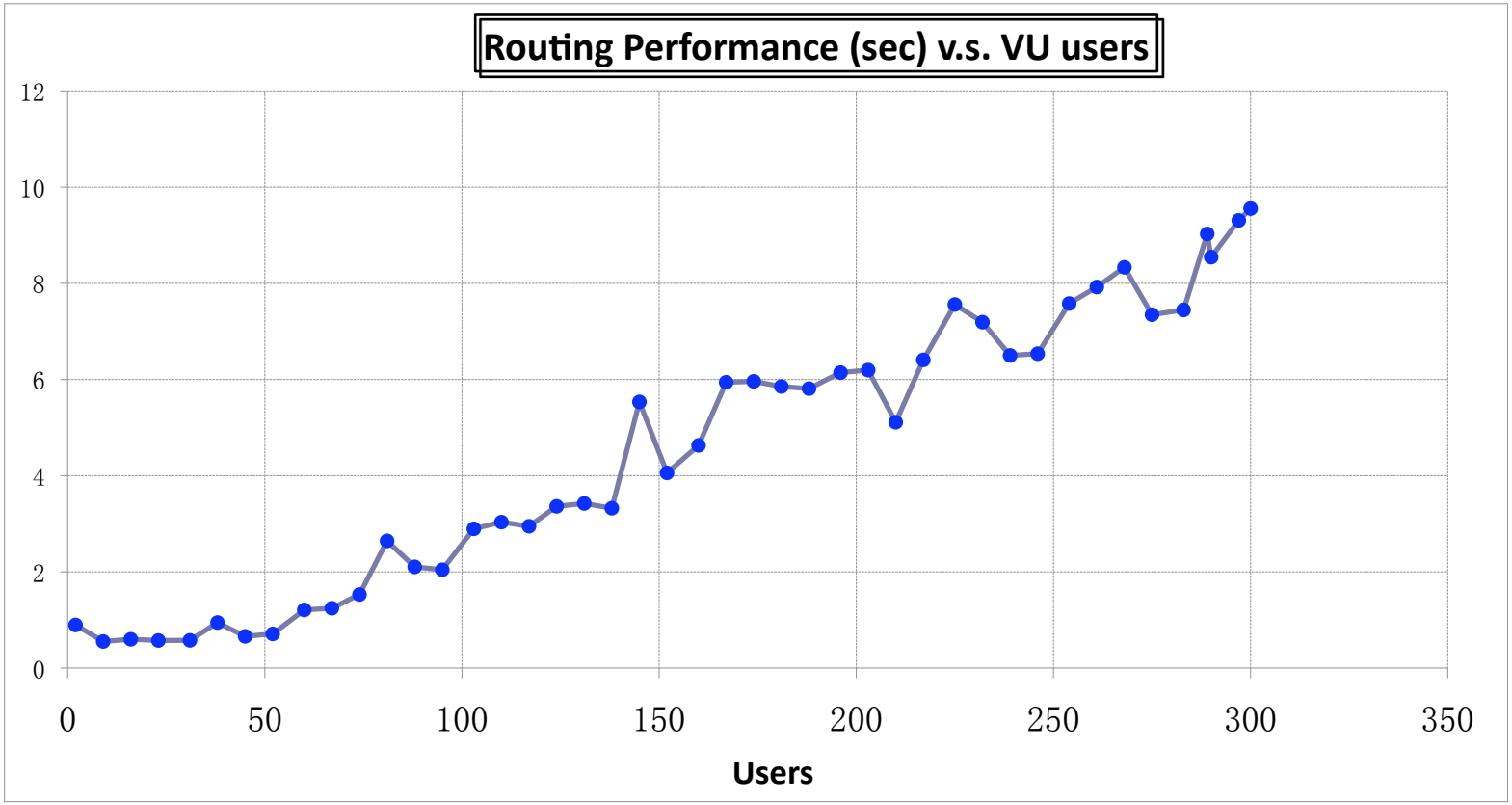


Performance in Responding Time

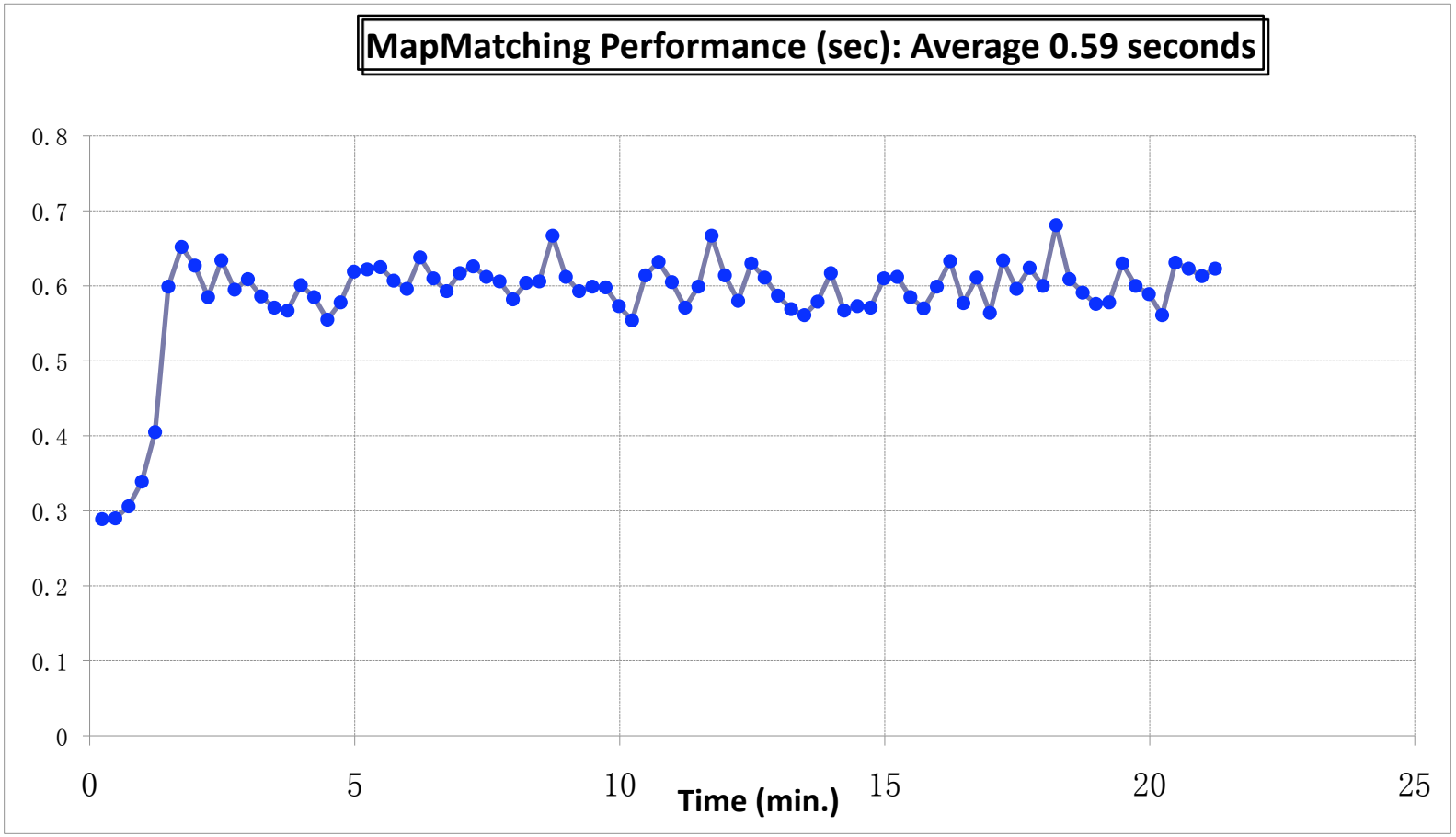


Over-Loaded System Performance

- With over-loaded request traffic, the responding time increases dramatically



Performance in Responding Time



Understand Different Criteria Used in Route Finding

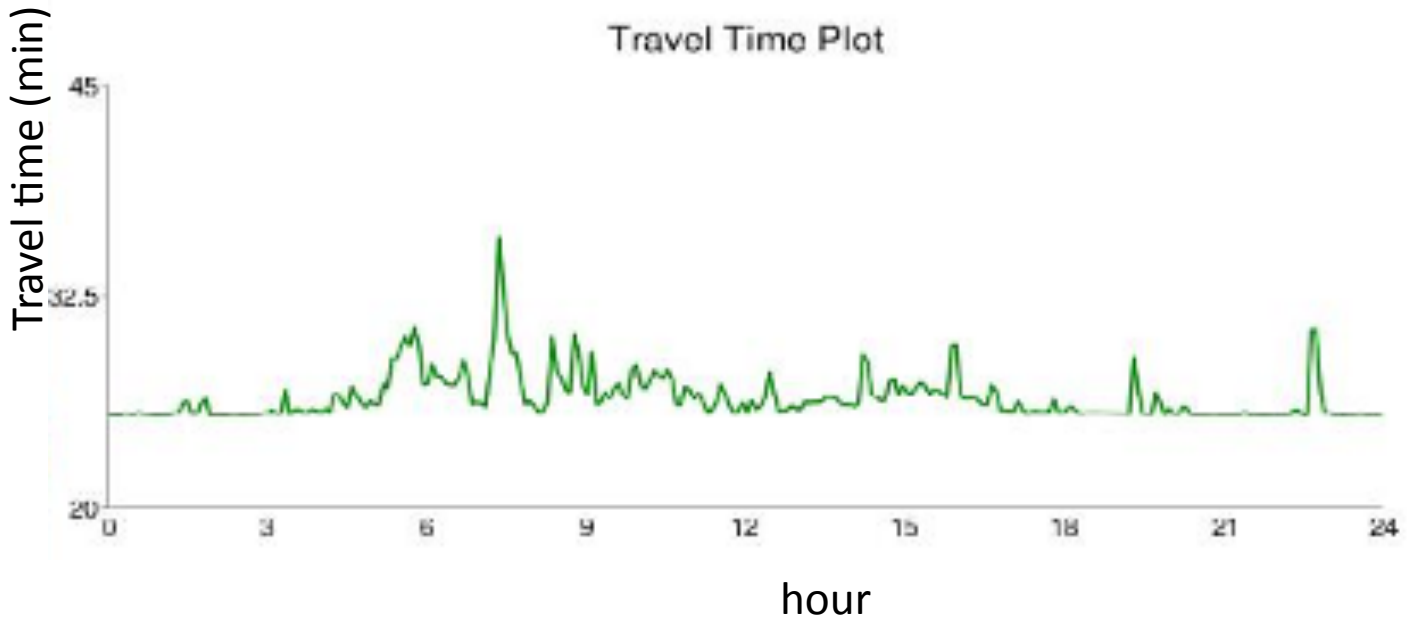
Fastest path (Yellow):	the least travel time
Shortest path (Blue):	the shortest travel distance
Eco path (Green):	the average speed is close to eco-driving speed (50-60 mph)
Avoid toll path (Orange):	no toll along the route or the lowest tolling fee
Safe path (Red):	the smallest probability of seeing/being involved in a traffic incident during the whole trip
Reliable (Black):	the highest travel time reliability
Park & Ride path (Brown):	use park & ride intermodal option

Remarks:

1) Different optimization criteria could lead to the same path.
2) Historical and live traffic data come from Traffic.com (NAVTEQ)
3) Transit and tolling data come from MTA.

Travel Time Profile

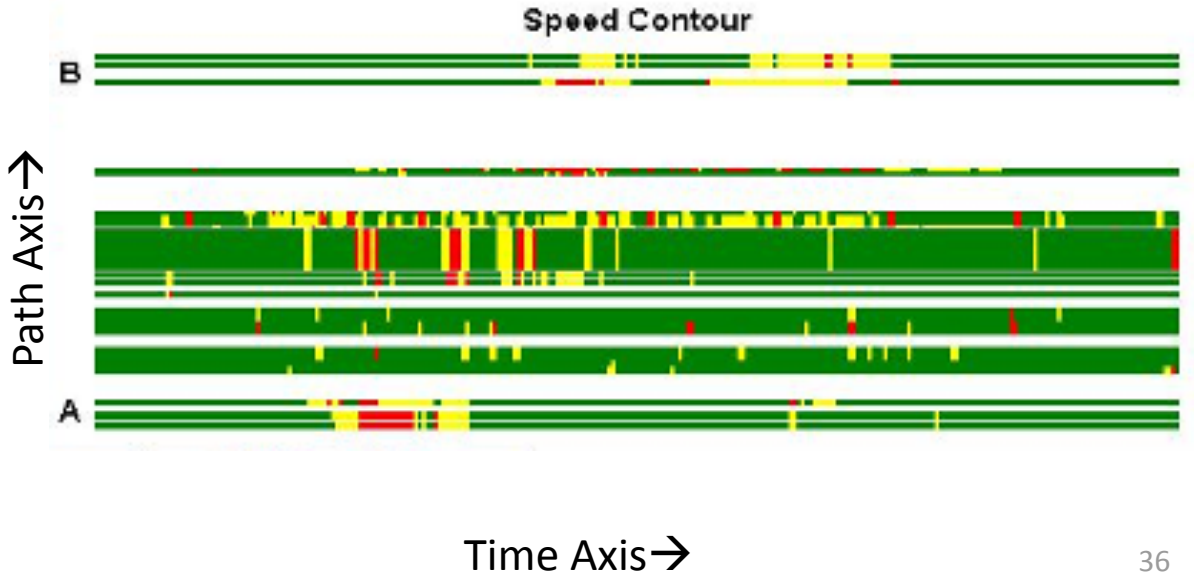
- Travel times at different departure times of day



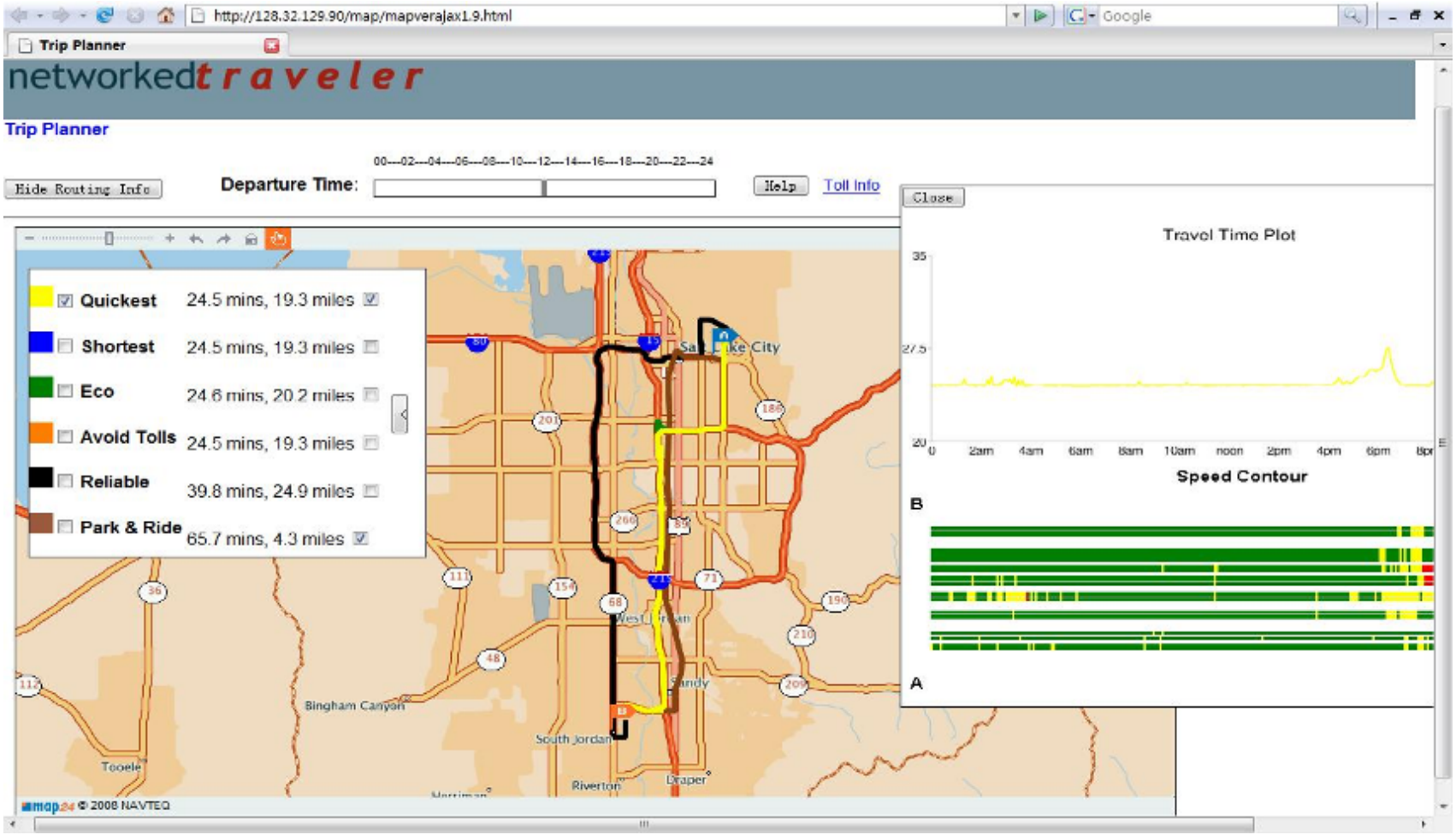
Traffic Conditions along Route

- traffic speed data collected from road sensors along the selected path (from origin A to destination B).
- The X axis represents time of day.
- The Y axis represents space along the path from A to B.

•Red: highly congested.
•Yellow: relatively congested.
•Green: free-flow.
•Blank/White: no sensor data.

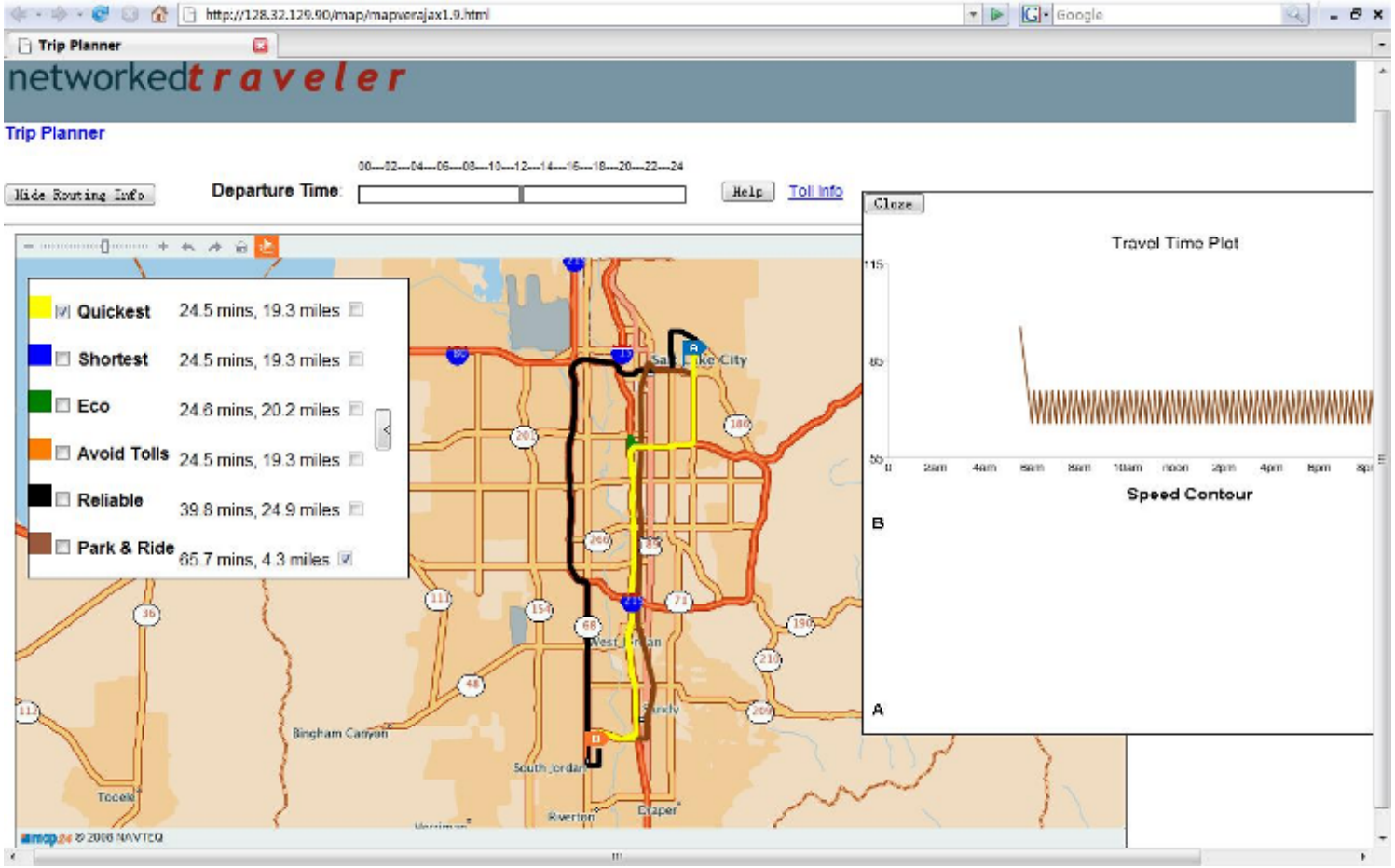


Quickest Route (Yellow)



Remark: driving-only mode uses traffic data from Navteq Inc.

Park & Ride Route (Brown)



Remarks: The zig-zap time-dependent travel time profile reflects the different waiting times due to transit headway

Mobile Phone-Based Interface