StarTrack Next Generation
A Scalable Infrastructure for Track-Based Applications

Maya Haridasan
Iqbal Mohomed
Doug Terry
Chandu Thekkath
Li Zhang

MICROSOFT RESEARCH
SILICON VALLEY

OSDI 2010
Location-Based Applications

• Many phones already have the ability to determine their own location
  - GPS, cell tower triangulation, or proximity to WiFi hotspots

• Many mobile applications use location information
Track

Time-ordered sequence of location readings

Latitude:  37.4013
Longitude:  -122.0730
Time:  07/08/10  08:46:45.125
Application: Personalized Driving Directions

Goal: Find directions to new gym

1. Depart Kirk Ave toward Balsam Ave < 0.1 mi
2. Turn right onto E Arbor Ave 0.1 mi
3. Turn left onto Borregas Ave, and then immediately turn right onto W Maude Ave 0.3 mi
4. Turn right onto N Mathilda Ave 0.7 mi
   VALERO on the corner
5. Take ramp right for US-101 North / Bayshore Fwy toward San Francisco 2.6 mi
6. At exit 399A, take ramp right and follow signs for Shoreline Blvd 0.6 mi
7. Turn left onto N Shoreline Blvd 1.5 mi
8. Turn right onto California St 0.8 mi
9. Arrive at near 2012 California St, Mountain View, CA 94040 on the right
   The last intersection is Escuela Ave
   If you reach S Rengstorff Ave, you've gone too far
Application: Personalized Driving Directions

Goal: Find directions to new gym

≈ Take US-101 North

1. Depart Kirk Ave toward Balsam Ave < 0.1 mi
2. Turn right onto E Arbor Ave 0.1 mi
3. Turn left onto Borregas Ave, and then immediately turn right onto W Maude Ave 0.3 mi
4. Turn right onto N Mathilda Ave VALERO on the corner 0.7 mi
5. Take ramp right for US-101 North / Bayshore Fwy toward San Francisco 2.6 mi
6. At exit 399A, take ramp right and follow signs for Shoreline Blvd 0.6 mi
7. Turn left onto N Shoreline Blvd 1.5 mi
8. Turn right onto California St 0.8 mi
9. Arrive at near 2012 California St, Mountain View, CA 94040 on the right

The last intersection is Escuela Ave If you reach S Rengstorff Ave, you’ve gone too far
# A Taxonomy of Applications

<table>
<thead>
<tr>
<th>Class of Applications</th>
<th>Personal</th>
<th>Social</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Current location</strong></td>
<td>Driving directions, Nearby restaurants</td>
<td>Friend finder, Crowd scenes</td>
</tr>
<tr>
<td><strong>Past locations</strong></td>
<td>Personal travel journal, Geocoded photos</td>
<td>Post-it notes, Recommendations</td>
</tr>
<tr>
<td><strong>Tracks</strong></td>
<td>Personalized Driving Directions, Track-Based Search</td>
<td>Ride sharing, Discovery, Urban sensing</td>
</tr>
</tbody>
</table>

**Class of applications enabled by StarTrack**
StarTrack System

- Insertion
- Retrieval
- Manipulation
- Comparison

Insertion Application
Location Manager
Application
ST Client
ST Server
ST Server
ST Server
System Challenges

1. Handling error-prone tracks
2. Flexible programming interface
3. Efficient implementation of operations on tracks
4. Scalability and fault tolerance
Challenges of Using Raw Tracks

Advantages of Canonicalization:

- More efficient retrieval and comparison operations
- Enables StarTrack to maintain a list of non-duplicate tracks
Track Collections (TC): Abstract grouping of tracks

- **Programming Convenience**
- **Implementation Efficiency**
  - Prevent unnecessary client-server message exchanges
  - Enable delayed evaluation
  - Enable caching and use of in-memory data structures
StarTrack API: Track Collections

Creation
- TC **MakeCollection**(GroupCriteria criteria, bool removeDuplicates)

Manipulation
- TC **JoinTrackCollections** (TC tCs[], bool removeDuplicates)
- TC **SortTracks** (TC tC, SortAttribute attr)
- TC **TakeTracks** (TC tC, int count)
- TC **GetSimilarTracks** (TC tC, Track refTrack, float simThreshold)
- TC **GetPassByTracks** (TC tC, Area[] areas)
- TC **GetCommonSegments** (TC tC, float freqThreshold)

Retrieval
- Track[] **GetTracks** (TC tC, int start, int count)
API Usage: Ride-Sharing Application

// get user’s most popular track in the morning
TC myTC = MakeCollection("name = Maya", [0800 1000], true);
TC myPopTC = SortTracks(myTC, FREQ);
Track track = GetTracks(myPopTC, 0, 1);

// find tracks of all fellow employees
TC msTC = MakeCollection("name.Employer = MS", [0800 1000], true);

// pick tracks from the community most similar to user’s popular track
TC similarTC = GetSimilarTracks(msTC, track, 0.8);
Track[] similarTracks = GetTracks(similarTC, 0, 20);

// Find owners of tracks, and verify that each track is frequently traveled
User[] result = FindOwnersOfFrequentTracks(similarTracks);
API Usage: Ride-Sharing Application

// get user’s most popular track in the morning
TC myTC = MakeCollection("name = Maya", [0800 1000], true);
TC myPopTC = SortTracks(myTC, FREQ);
Track track = GetTracks(myPopTC, 0, 1);

// find tracks of all fellow employees
TC msTC = MakeCollection("name.Employer = MS", [0800 1000], true);

// pick tracks from the community most similar to user’s popular track
TC similarTC = GetSimilarTracks(msTC, track, 0.8);
Track[] similarTracks = GetTracks(similarTC, 0, 20);

// Find owners of tracks, and verify that each track is frequently traveled
User[] result = FindOwnersOfFrequentTracks(similarTracks);
Efficient Implementation of Operations

• StarTrack exploits redundancy in tracks for efficient retrieval from database
  - Set of non-duplicate tracks per user
  - Separate table of unique coordinates

• StarTrack builds specialized in-memory data-structures to accelerate the evaluation of some operations
  - Quad-Trees for geographic range searches
  - Track Trees for similarity searches
Track A = Track B = S1, S2, S3, S4, S5
Track C = S1, S2, S3, S4, S6, S7
Track D = S1, S2, S3, S8, S9
Limited database support for computing track similarity

Track A = Track B = S1, S2, S3, S4, S5
Track C = S1, S2, S3, S4, S6, S7
Track D = S1, S2, S3, S8, S9

\[
\text{SIM}(A,B) = \frac{|S1-5|}{|S1-5|} = 1
\]

\[
\text{SIM}(A,C) = \frac{|S1-4|}{|S1-4| + |S5| + |S6-7|}
\]
Track Tree

GetSimilarTracks, GetCommonSegments
Evaluation

- Performance of our Track Tree approach
- Performance of 2 sample applications
  - Personalized Driving Directions
  - Ride-sharing

- Configuration
  - Synthetically generated tracks
  - Up to 9 StarTrack Servers + 3 Database Servers
  - Server Configuration:
    - 2.6 GHz AMD Opteron Quad-Core Processors
    - 16 GB RAM
Evaluation: Track Tree

• Evaluation of GetSimilarTracks

• Alternative approaches:
  - Database filtering
    Pre-filter tracks that intersect ref track at database
  - In-memory filtering
    Pre-filter tracks that intersect ref track in memory
  - In-memory brute force
    Compute similarity between each track and ref track in memory
Get Similar Tracks – Query Time

Query Time (ms)

Number of tracks (thousands)

Database Filtering

In-Memory Brute Force

In-Memory Filtering

Track Tree
Track Tree Construction Costs

- Memory
- Time

Number of Tracks (thousands) vs. MBytes and Seconds.
Performance of Applications

**Personalized Driving Directions**
- Track Collection for single user at a time
- Calls to GetCommonSegments
- 30 requests/s at about 100 ms (uncached)
- 250 requests/s at about 55 ms (cached)

**Ride Sharing**
- Track Collection on multiple users
- Calls to GetSimilarTracks
- 30 requests/s at about 170 ms
Summary

• StarTrack is a scalable service designed to manage tracks and facilitate the construction of track-based applications

• Important Design Features
  ▪ Canonicalization of Tracks
  ▪ API based on Track Collections
  ▪ Use of Novel Data Structures

• Availability:
  ▪ We are looking for users of our infrastructure. Please contact one of the authors if you are interested.