StarTrack Next Generation: A Scalable Infrastructure for Track-Based Applications

Maya Haridasan, Iqbal Mohomed, Doug Terry, Chandramohan A. Thekkath, and Li Zhan

Presentation by Maciej Klimek

Department of Mathematics, Computer Science and Mechanics
University of Warsaw

October 26, 2011

Outline

Outline

Outline

Outline

Outline

Outline

Whats the problem?

- Most of the mobile devices produced nowadays are equipped with some kind of hardware that provides their physical location.
- We can use try to use this information to provide enhanced functionality to these users.

Whats the problem?

- Most of the mobile devices produced nowadays are equipped with some kind of hardware that provides their physical location.
- We can use try to use this information to provide enhanced functionality to these users.

What is a track?

Track is a time ordered sequence of GPS locations recorded by mobile device, representing a route.

What is a track-based application?

Track-base applications uses tracks collected by users to provide better user experience.

Introductory note

Instead of using "raw" tracks – sequence of coordinates reported by GPS, StarTrack uses it's canonical form. It represents a track as a sequence of points drawn from a fixed set, such as road intersections. More on canonicalization in the later part of the presentation.

What is StarTrack?

What is StarTrack?

StarTrack is the first service designed to manage tracks of GPS location coordinates obtained from mobile devices and to facilitate the construction of track-base applications.

StarTrack Next Generation vs. StarTrack

This presentation is about StartTrack Next Generation, this is actually second version of StarTrack system. The first version was essentially a single database server with a thin veneer of software providing the API. Thanks to authors experience with the first version many aspects such as API, performance were revised resulting in StarTrack Next Generation.

Ride-sharing service

- Most of the time not all seats in a car are occupied.
- We can try to utilize this empty seats. This can help to lower the worldwide fuel consumption and transportation costs.
- Every company could have their own ride-sharing service for their employees.
- We can also use existing social networks to establish trust between drivers and passengers.
- Working example – http://www.rideshareonline.com/

Ride-sharing service

- Most of the time not all seats in a car are occupied.
- We can try to utilize this empty seats. This can help to lower the worldwide fuel consumption and transportation costs.
- Every company could have their own ride-sharing service for their employees.
- We can also use existing social networks to establish trust between drivers and passengers.
- Working example – http://www.rideshareonline.com/

Ride-sharing service

- Most of the time not all seats in a car are occupied.
- We can try to utilize this empty seats. This can help to lower the worldwide fuel consumption and transportation costs.
- Every company could have their own ride-sharing service for their employees.
- We can also use existing social networks to establish trust between drivers and passengers.
- Working example – http://www.rideshareonline.com/

Ride-sharing service

- Most of the time not all seats in a car are occupied.
- We can try to utilize this empty seats. This can help to lower the worldwide fuel consumption and transportation costs.
- Every company could have their own ride-sharing service for their employees.
- We can also use existing social networks to establish trust between drivers and passengers.
- Working example – http://www.rideshareonline.com/

Ride-sharing service

- Most of the time not all seats in a car are occupied.
- We can try to utilize this empty seats. This can help to lower the worldwide fuel consumption and transportation costs.
- Every company could have their own ride-sharing service for their employees.
- We can also use existing social networks to establish trust between drivers and passengers.
- Working example – http://www.rideshareonline.com/

Personalized driving directions

- Current navigation systems provide each user with detailed – turn-by-turn directions of driving route.
- It is often a case that a driver will know some parts of the route almost by heart.
- If we could know what the driver knows, we could try to provide him with personalized driving directions.

Personalized driving directions

- Current navigation systems provide each user with detailed – turn-by-turn directions of driving route.
- It is often a case that a driver will know some parts of the route almost by heart.
- If we could know what the driver knows, we could try to provide him with personalized driving directions.

Personalized driving directions

- Current navigation systems provide each user with detailed – turn-by-turn directions of driving route.
- It is often a case that a driver will know some parts of the route almost by heart.
- If we could know what the driver knows, we could try to provide him with personalized driving directions.

Other applications

These two previous applications were actually build by authors using StarTrack for purpose of its evaluation.
Here are some other example applications:

- Traffic jams forecasting.
- Personalized advertising.
- Social applications.

How can StarTrack help?

StarTrack

As we will see later StarTrack facilitates the construction of such applications by providing necessary framework for handling tracks.

Track Collection

Track collection

Track collection is just a grouping of individual track. Most of the StarTrack's API take track collections as arguments.

How do we create a track collection?

```
TrackCollxn MakeCollection(GrpCriteria[] gCrit,
bool unique)
```

There are three types of criteria available: geographic, time, user. The unique parameter specifies if the tracks that are "highly similar" should be reported once or multiple times.

GetSimilarTracks

- Many track-base applications need to find track similar to a particular track.
- Give two tracks we define their similarity as the ratio of the length of all the common segments and the union of the segments present in either of them.

```
TrackCollxn GetSimilarTracks(TrkCollxn tC,
Trk refTrk, float simThresh)
```

GetSimilarTracks

- Many track-base applications need to find track similar to a particular track.
- Give two tracks we define their similarity as the ratio of the length of all the common segments and the union of the segments present in either of them.

```
TrackCollxn GetSimilarTracks(TrkCollxn tC,
Trk refTrk, float simThresh)
```

GetCommonSegments

GetCommonSegments takes a track collection and frequency threshold and returns the road segments shared by at least that fraction of tracks in the collection, merged in the smallest number of contiguous routes possible.

Segment

Segment is a part of the track between two points, with no other points inbetween.

```
TrackCollxn GetCommonSegments(TrkCollxn tC,
float freqThresh)
```

GetPassByTracks

GetPassByTracks is give a track collection and an array of Area objects and returns all tracks in the collection that pass through all the areas.

```
TrackCollxn GetPassByTracks(TrkCollxn tC,
Area[] areas);
```

How to retrieve tracks from track collection?

```
int GetTrackCount(TrkCollxn tC);
Track[] GetTracks(TrkCollxn tC, int start, int count);
```

It's obvious what their semantics is.

GetCommonSegments, GetSimilarTracks, GetCommonSegments and GetTracks are only a part of the API, but they are important because they help to illustrate some of the concepts in the design of StarTrack.

API continued

```
TrackCollxn JoinTrkCollections(TrkCollxn tCs[],
bool unique);
TrackCollxn SortTracks(TrkCollxn tC,
SortAttribute attr);
```

Note about API

Apart from presented API calls there are also some others functions, such as adding tracks to the system. But We will omit them.

Overview of StarTrack architecture

StarTrack platform consists of:

- Database servers – stores persistent data, uses Microsoft's SQL Server 2008. Data is partitioned across multiple machines, partitions are replicated.
- StarTrack servers – handles requests to operate on tracks, builds and maintains in-memory structures, such as Track Tree.
- StarTrack Clerk – handles requests from users and sends them to StartTrack servers. Deals with server failures and balancing the load among servers.

Why is the "raw" track representation bad?

We could store a track just a sequence of coordinates that we got from GPS device, but there are some problems:

- Two GPS samples collected on the same route will be different.
- Comparing such two track is difficult.
- Sampling is error-prone.
- How do we solve the problem?

Why is the "raw" track representation bad?

We could store a track just a sequence of coordinates that we got from GPS device, but there are some problems:

- Two GPS samples collected on the same route will be different.
- Comparing such two track is difficult.
- Sampling is error-prone.
- How do we solve the problem?

Why is the "raw" track representation bad?

We could store a track just a sequence of coordinates that we got from GPS device, but there are some problems:

- Two GPS samples collected on the same route will be different.
- Comparing such two track is difficult.
- Sampling is error-prone.
- How do we solve the problem?

Why is the "raw" track representation bad?

We could store a track just a sequence of coordinates that we got from GPS device, but there are some problems:

- Two GPS samples collected on the same route will be different.
- Comparing such two track is difficult.
- Sampling is error-prone.
- How do we solve the problem?

Solution: Canonicalization

Canonicalization

Canonicalization of a "raw" track transforms it to track that only passes through some points drawn from a large fixed set.

How do we choose this large fixed set of points?

We could choose some artificially create set of points, but instead of doing it, we use road intersections as the set. The process of canonicalization to such set of points is called map matching.

StarTrack performs map matching using hidden Markov models. It takes under 250ms to canonicalize a track of length 20km with 400 GPS samples.

Map construction

It might be the case that we don't have access to the map of the region. Then we can use technologies for constructing road maps from users tracks(StarTrack doesn't do it, but it could in the future).

What is delayed evaluation?

Experimental observation

Typical application makes several API calls to narrow down the set of tracks they want to retrieve.

- Implementation of the StarTrack API uses this fact, and delays the evaluation of the tracks until either GetTrackCount, or GetTracks is called.
- This technique increases performance, because it reduces the amount of data that has to be send between the server and the client.

What is delayed evaluation?

Experimental observation

Typical application makes several API calls to narrow down the set of tracks they want to retrieve.

- Implementation of the StarTrack API uses this fact, and delays the evaluation of the tracks until either GetTrackCount, or GetTracks is called.
- This technique increases performance, because it reduces the amount of data that has to be send between the server and the client.

How to implement delayed evaluation?

Descriptor

When a MakeCollection is called, client-side stub creates a descriptor describing the call, send it to the server. The server stamps the descriptor with the current time and returns it to the caller. Assuming that the tracks are not deleted from the database, this descriptor can be interpreted as a logical view of the database.

Compound descriptors

Operations such as GetSimilarTracks, GetPassByTracks, JoinTrkCollections, ... create composition of these descriptors with no communication to the server. Note that the these descriptors create a tree, with leaves being the descriptors from MakeCollection operation.

Evaluation of a descriptor

- When user calls one of the track retrieval functions(GetTracks, GetTrackCount) on a track collection(descriptor of the track collection), system will evaluate the descriptor.
- Evaluation of different types of descriptors might trigger construction of various in-memory structures.
- Evaluation of a GetSimilarTracks might trigger the construction of Track Tree(more about Track Tree in a second).
- While the evaluation of GetPassByTracks will trigger the construction of a quad tree.
- These in-memory structures are cached with the hope that they will be reused in the future.

Evaluation of a descriptor

- When user calls one of the track retrieval functions(GetTracks, GetTrackCount) on a track collection(descriptor of the track collection), system will evaluate the descriptor.
- Evaluation of different types of descriptors might trigger construction of various in-memory structures.
- Evaluation of a GetSimilarTracks might trigger the construction of Track Tree(more about Track Tree in a second).
- While the evaluation of GetPassByTracks will trigger the construction of a quad tree.
- These in-memory structures are cached with the hope that they will be reused in the future.

Evaluation of a descriptor

- When user calls one of the track retrieval functions(GetTracks, GetTrackCount) on a track collection(descriptor of the track collection), system will evaluate the descriptor.
- Evaluation of different types of descriptors might trigger construction of various in-memory structures.
- Evaluation of a GetSimilarTracks might trigger the construction of Track Tree(more about Track Tree in a second).
- While the evaluation of GetPassByTracks will trigger the construction of a quad tree.
- These in-memory structures are cached with the hope that they will be reused in the future.

Evaluation of a descriptor

- When user calls one of the track retrieval functions(GetTracks, GetTrackCount) on a track collection(descriptor of the track collection), system will evaluate the descriptor.
- Evaluation of different types of descriptors might trigger construction of various in-memory structures.
- Evaluation of a GetSimilarTracks might trigger the construction of Track Tree(more about Track Tree in a second).
- While the evaluation of GetPassByTracks will trigger the construction of a quad tree.
- These in-memory structures are cached with the hope that they will be reused in the future.

Evaluation of a descriptor

- When user calls one of the track retrieval functions(GetTracks, GetTrackCount) on a track collection(descriptor of the track collection), system will evaluate the descriptor.
- Evaluation of different types of descriptors might trigger construction of various in-memory structures.
- Evaluation of a GetSimilarTracks might trigger the construction of Track Tree(more about Track Tree in a second).
- While the evaluation of GetPassByTracks will trigger the construction of a quad tree.
- These in-memory structures are cached with the hope that they will be reused in the future.

Track Tree construction

- Track Tree is constructed for a track collection.
- A node in a Track Tree represents a contiguous sequence of segments.
- A node stores information about tracks that contain it.
- Each road segment from the track collection is represented by a leaf in the Track Tree.
- Repeat this step: join geographically adjacent nodes.
- If there is a choice in the previous step, then join such nodes that have most tracks in common.

Track Tree construction

- Track Tree is constructed for a track collection.
- A node in a Track Tree represents a contiguous sequence of segments.
- A node stores information about tracks that contain it.
- Each road segment from the track collection is represented by a leaf in the Track Tree.
- Repeat this step: join geographically adjacent nodes.
- If there is a choice in the previous step, then join such nodes that have most tracks in common.

Track Tree construction

- Track Tree is constructed for a track collection.
- A node in a Track Tree represents a contiguous sequence of segments.
- A node stores information about tracks that contain it.
- Each road segment from the track collection is represented by a leaf in the Track Tree.
- Repeat this step: join geographically adjacent nodes.
- If there is a choice in the previous step, then join such nodes that have most tracks in common.

Track Tree construction

- Track Tree is constructed for a track collection.
- A node in a Track Tree represents a contiguous sequence of segments.
- A node stores information about tracks that contain it.
- Each road segment from the track collection is represented by a leaf in the Track Tree.
- Repeat this step: join geographically adjacent nodes.
- If there is a choice in the previous step, then join such nodes that have most tracks in common.

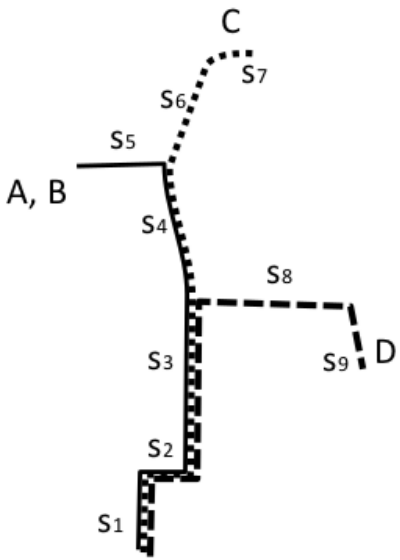
Track Tree construction

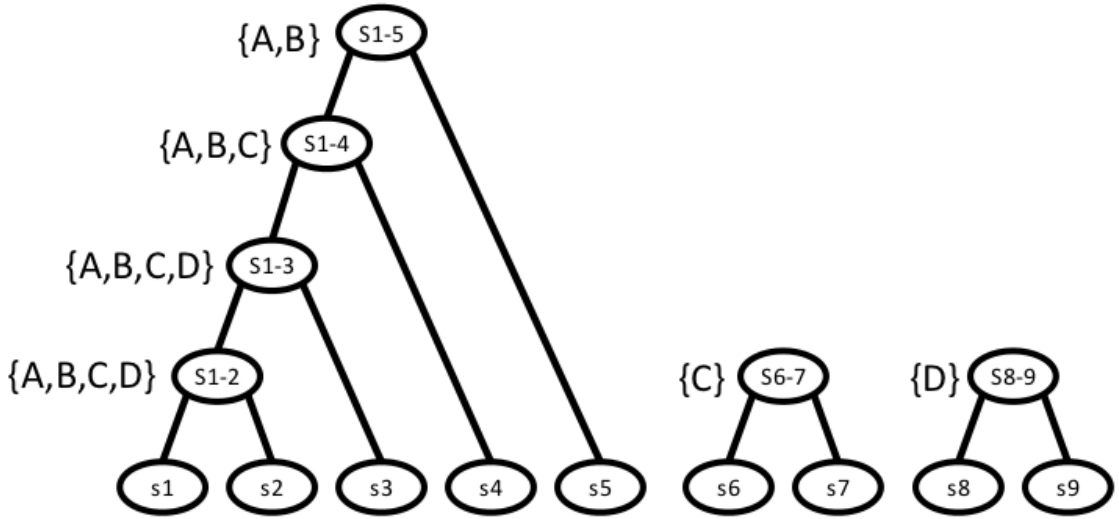
- Track Tree is constructed for a track collection.
- A node in a Track Tree represents a contiguous sequence of segments.
- A node stores information about tracks that contain it.
- Each road segment from the track collection is represented by a leaf in the Track Tree.
- Repeat this step: join geographically adjacent nodes.
- If there is a choice in the previous step, then join such nodes that have most tracks in common.

Track Tree construction

- Track Tree is constructed for a track collection.
- A node in a Track Tree represents a contiguous sequence of segments.
- A node stores information about tracks that contain it.
- Each road segment from the track collection is represented by a leaf in the Track Tree.
- Repeat this step: join geographically adjacent nodes.
- If there is a choice in the previous step, then join such nodes that have most tracks in common.

Sample Track Tree





Where do we use it?

- Implement GetSimilarTracks. Not always accurate, produces false-negatives, but good enough.
- In GetCommonSegments merge segments into small number of tracks.
- StarTrack tries to cache Track Tree(at StarTrack Server). A typical scenario for a ride-sharing application would be to create a track collection for some group of people. And then members of this group would call GetSimilarTrack, trying to find a ride-partner. Thus caching would provide significant performance improvements.

Where do we use it?

- Implement GetSimilarTracks. Not always accurate, produces false-negatives, but good enough.
- In GetCommonSegments merge segments into small number of tracks.
- StarTrack tries to cache Track Tree(at StarTrack Server). A typical scenario for a ride-sharing application would be to create a track collection for some group of people. And then members of this group would call GetSimilarTrack, trying to find a ride-partner. Thus caching would provide significant performance improvements.

Where do we use it?

- Implement GetSimilarTracks. Not always accurate, produces false-negatives, but good enough.
- In GetCommonSegments merge segments into small number of tracks.
- StarTrack tries to cache Track Tree(at StarTrack Server). A typical scenario for a ride-sharing application would be to create a track collection for some group of people. And then members of this group would call GetSimilarTrack, trying to find a ride-partner. Thus caching would provide significant performance improvements.

Database tables

StarTrack uses 5 tables.

- User table – contains necessary information about users.
- Track table – stores all tracks in both "raw" and canonical form.
- Representative Track table – for each user it stores representative set of tracks. Allows to speed up some operations.
- Coordinate table – points used in canonicalization process.
- Coordinate To Track table – maps coordinates to tracks going though them. Allows to speed up some operations.

Database tables

StarTrack uses 5 tables.

- User table – contains necessary information about users.
- Track table – stores all tracks in both "raw" and canonical form.
- Representative Track table – for each user it stores representative set of tracks. Allows to speed up some operations.
- Coordinate table – points used in canonicalization process.
- Coordinate To Track table – maps coordinates to tracks going though them. Allows to speed up some operations.

Database tables

StarTrack uses 5 tables.

- User table – contains necessary information about users.
- Track table – stores all tracks in both "raw" and canonical form.
- Representative Track table – for each user it stores representative set of tracks. Allows to speed up some operations.
- Coordinate table – points used in canonicalization process.
- Coordinate To Track table – maps coordinates to tracks going though them. Allows to speed up some operations.

Database tables

StarTrack uses 5 tables.

- User table – contains necessary information about users.
- Track table – stores all tracks in both "raw" and canonical form.
- Representative Track table – for each user it stores representative set of tracks. Allows to speed up some operations.
- Coordinate table – points used in canonicalization process.
- Coordinate To Track table – maps coordinates to tracks going though them. Allows to speed up some operations.

Database tables

StarTrack uses 5 tables.

- User table – contains necessary information about users.
- Track table – stores all tracks in both "raw" and canonical form.
- Representative Track table – for each user it stores representative set of tracks. Allows to speed up some operations.
- Coordinate table – points used in canonicalization process.
- Coordinate To Track table – maps coordinates to tracks going though them. Allows to speed up some operations.

Database server organization

- Track data is partitioned among database servers.
- Partitioning is done with the respect to user identifier – all the user's tracks are stored together.

How to get data to test?

- During the experiments synthetic data were used.
- Important features of real life(16,000 tracks collected in Seattle, WA) tracks were reflected in the synthetic data.
  - Each person has fixed home and workplace location.
  - On weekdays a person travels between home and workplace.
  - He also travels to some other locations, although more on weekends than on weekdays.
- Data was generated for 3-month period for 18,000 users resulting in 4.5 million tracks.

How to get data to test?

- During the experiments synthetic data were used.
- Important features of real life(16,000 tracks collected in Seattle, WA) tracks were reflected in the synthetic data.
  - Each person has fixed home and workplace location.
  - On weekdays a person travels between home and workplace.
  - He also travels to some other locations, although more on weekends than on weekdays.
- Data was generated for 3-month period for 18,000 users resulting in 4.5 million tracks.

How to get data to test?

- During the experiments synthetic data were used.
- Important features of real life(16,000 tracks collected in Seattle, WA) tracks were reflected in the synthetic data.
  - Each person has fixed home and workplace location.
  - On weekdays a person travels between home and workplace.
  - He also travels to some other locations, although more on weekends than on weekdays.
- Data was generated for 3-month period for 18,000 users resulting in 4.5 million tracks.

How to get data to test?

- During the experiments synthetic data were used.
- Important features of real life(16,000 tracks collected in Seattle, WA) tracks were reflected in the synthetic data.
  - Each person has fixed home and workplace location.
  - On weekdays a person travels between home and workplace.
  - He also travels to some other locations, although more on weekends than on weekdays.
- Data was generated for 3-month period for 18,000 users resulting in 4.5 million tracks.

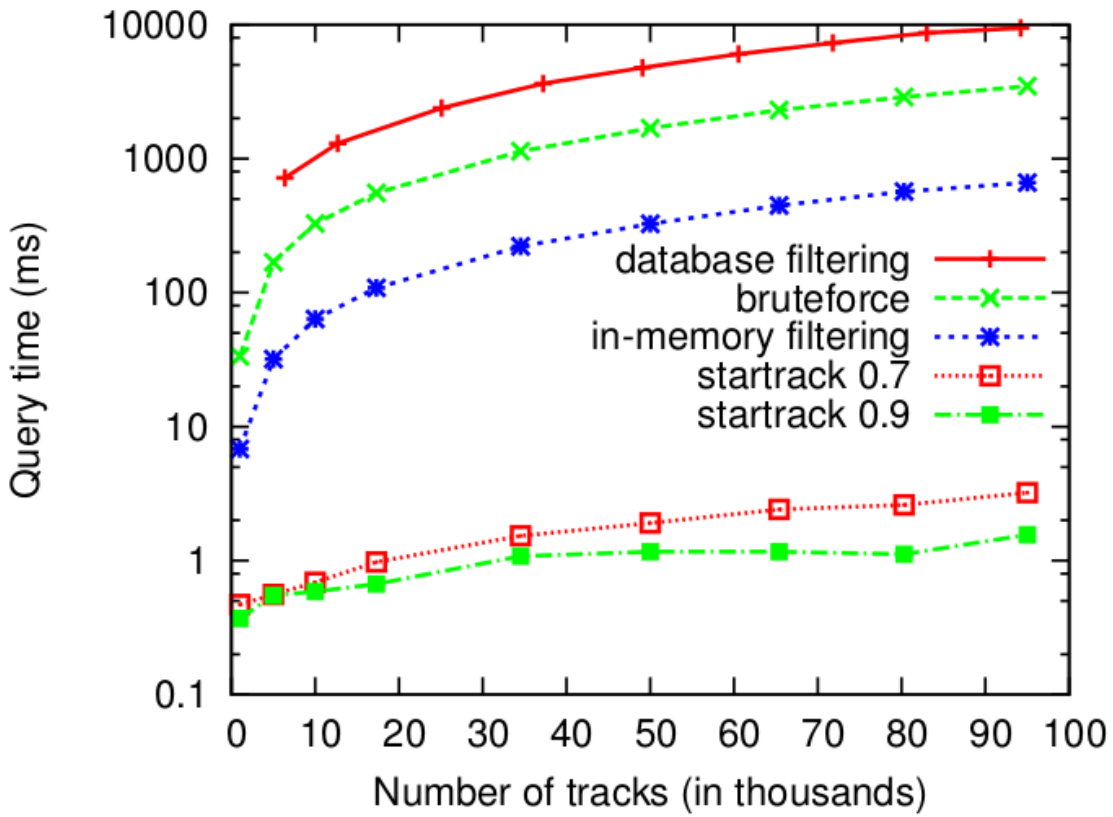
How to get data to test?

- During the experiments synthetic data were used.
- Important features of real life(16,000 tracks collected in Seattle, WA) tracks were reflected in the synthetic data.
  - Each person has fixed home and workplace location.
  - On weekdays a person travels between home and workplace.
  - He also travels to some other locations, although more on weekends than on weekdays.
- Data was generated for 3-month period for 18,000 users resulting in 4.5 million tracks.

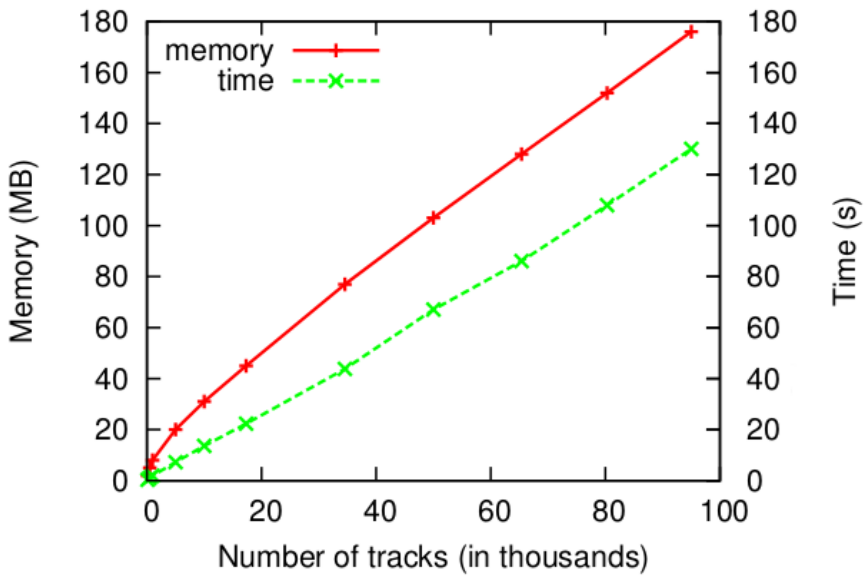
How to get data to test?

- During the experiments synthetic data were used.
- Important features of real life(16,000 tracks collected in Seattle, WA) tracks were reflected in the synthetic data.
  - Each person has fixed home and workplace location.
  - On weekdays a person travels between home and workplace.
  - He also travels to some other locations, although more on weekends than on weekdays.
- Data was generated for 3-month period for 18,000 users resulting in 4.5 million tracks.

Performance of track comparison(GetSimilarTracks)

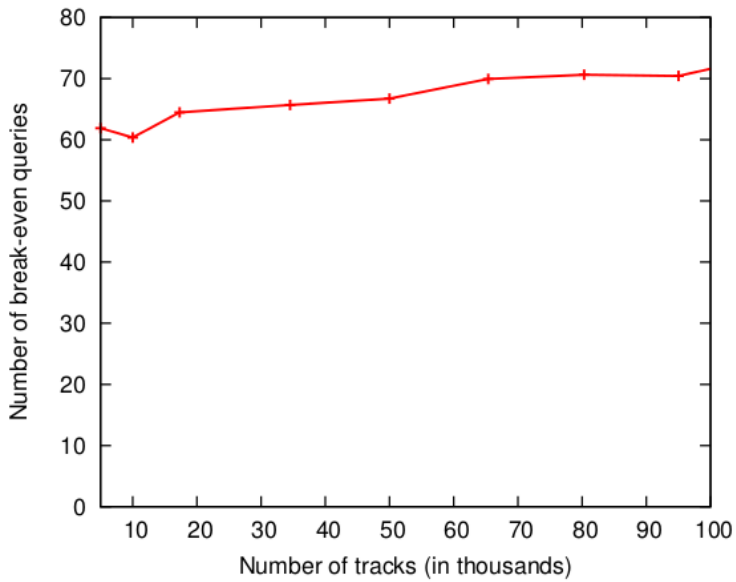


Cost of building Track Tree



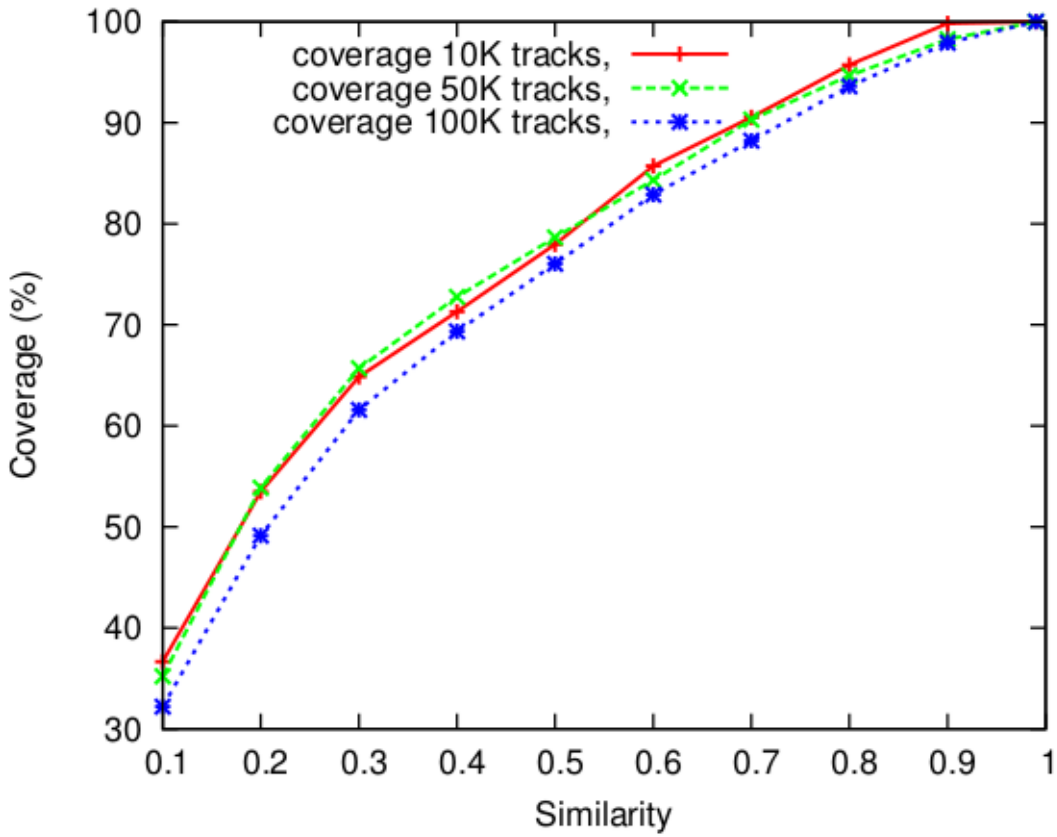
As we see the cost of building a track tree is quite high, how many times do we have to use such tree to make it cost effective?

Break-even numbers



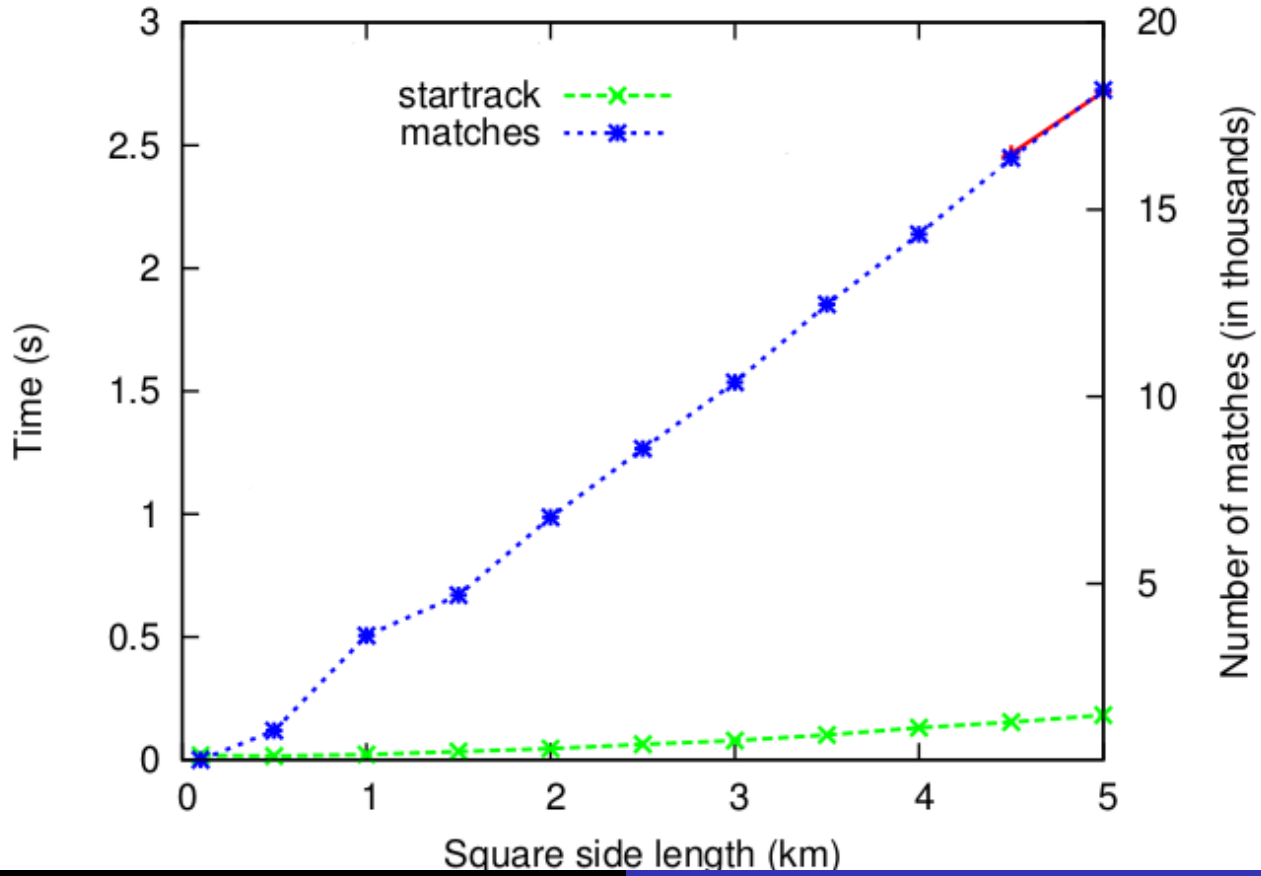
With track collection of size 100K we have to perform at least 70 queries to amortize the cost of track tree construction.

Accuracy of Track Trees



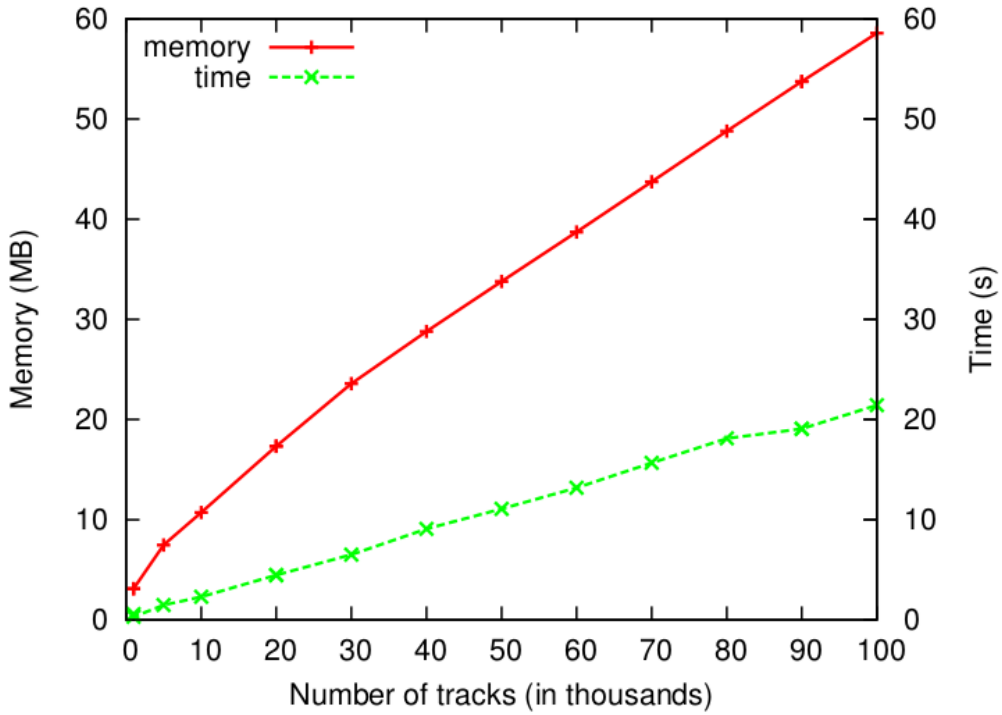
Geographic queries to the database

In the implementation of GetPassByTracks we use "Coordinate Table" and "Coordinate To Track Table" to speed up the process.

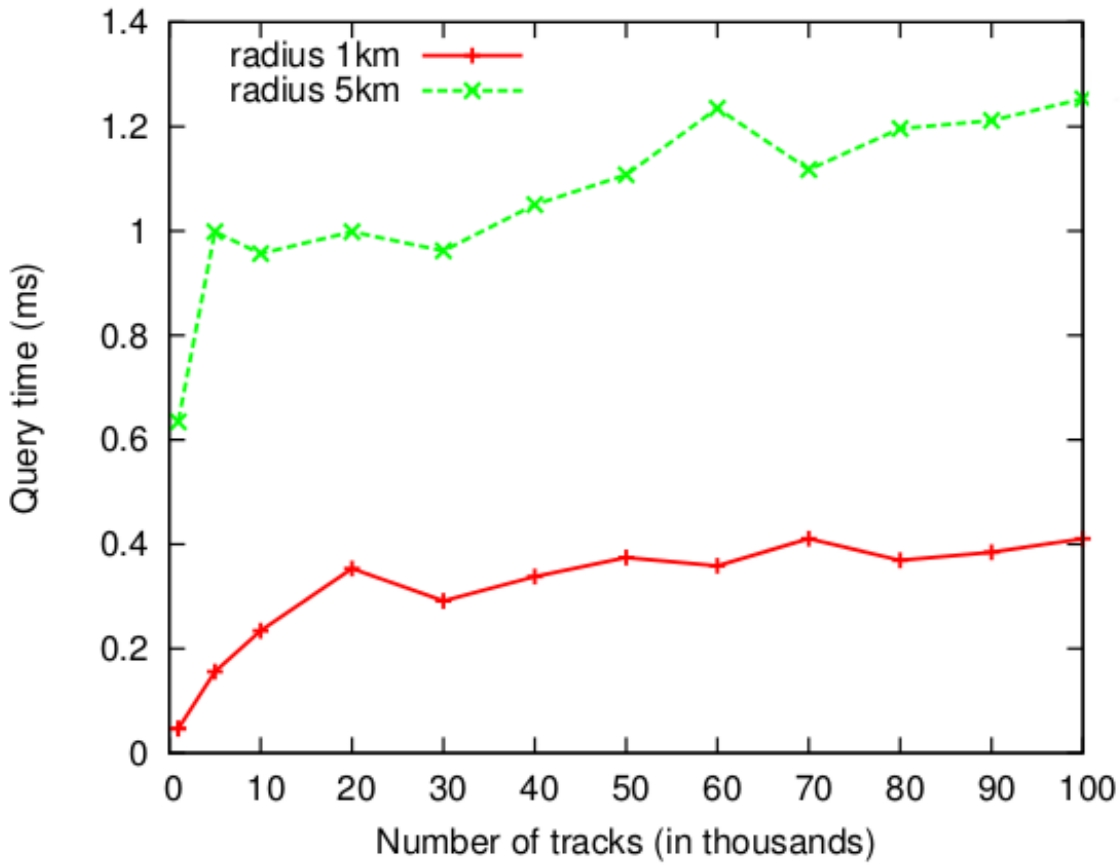


Quad tree construction performance

As we said earlier evaluation of GetPassByTracks can yield the construction of quad tree. Here is the performance of constructing it.



Quad tree query time



- StarTrack facilitates the construction of a broad class of track-based applications
- Provides significant performance and API improvements over the previous version of StarTrack
- Uses some innovative structures(Track Tree) and interesting techniques(delayed execution, canonicalization)

- StarTrack facilitates the construction of a broad class of track-based applications
- Provides significant performance and API improvements over the previous version of StarTrack
- Uses some innovative structures(Track Tree) and interesting techniques(delayed execution, canonicalization)

- StarTrack facilitates the construction of a broad class of track-based applications
- Provides significant performance and API improvements over the previous version of StarTrack
- Uses some innovative structures(Track Tree) and interesting techniques(delayed execution, canonicalization)

Thank You!