TrajStore: an Adaptive Storage System for Very Large Trajectory Data Sets

Philippe Cudré-Mauroux
Eugene Wu
Samuel Madden

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

ICDE 2010, March 2
Long Beach, CA, USA

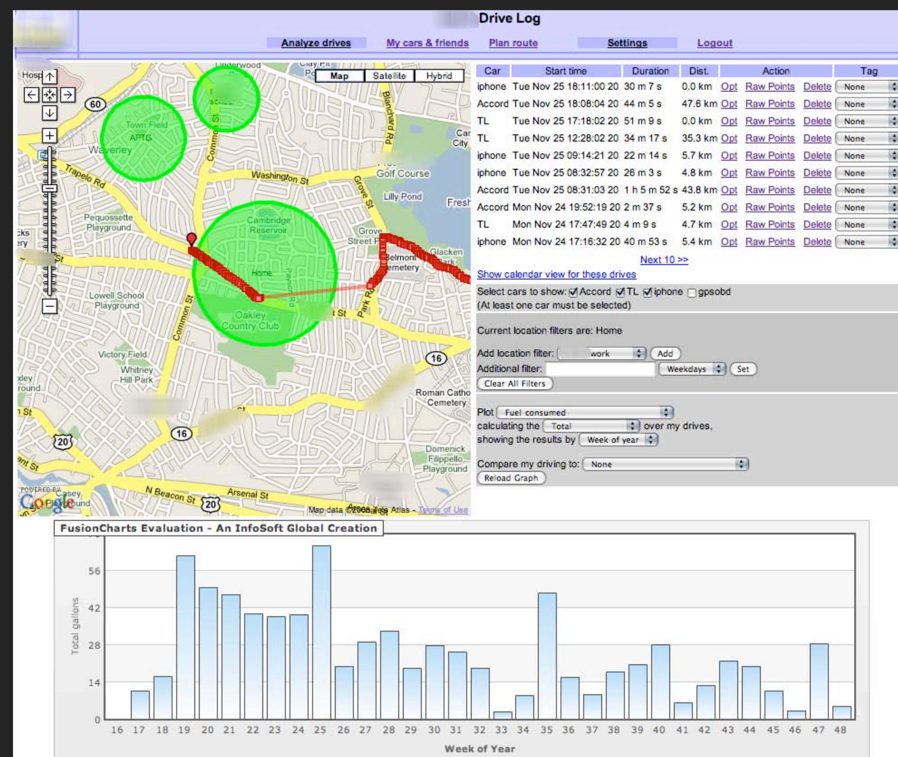
CSAIL

Motivation (1/2)

Explosion of position-aware devices & apps

MIT's CarTel project

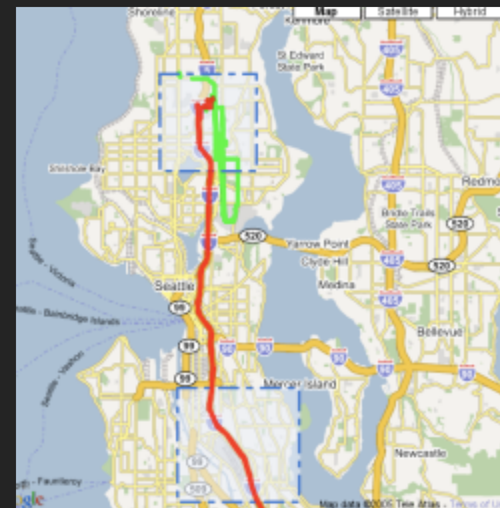
(Balakrishnan, Madden)



CSAIL

Motivation (2/2)

- CarTel
  - Massive amounts of GPS data
  - Real-time, high insert rates
  - Large spatiotemporal queries



➡ New class of applications
  - Live feeds from large fleets of mobile objects

- Current solutions (e.g., PostGIS) failed
  - Designed for (relatively) sparse data

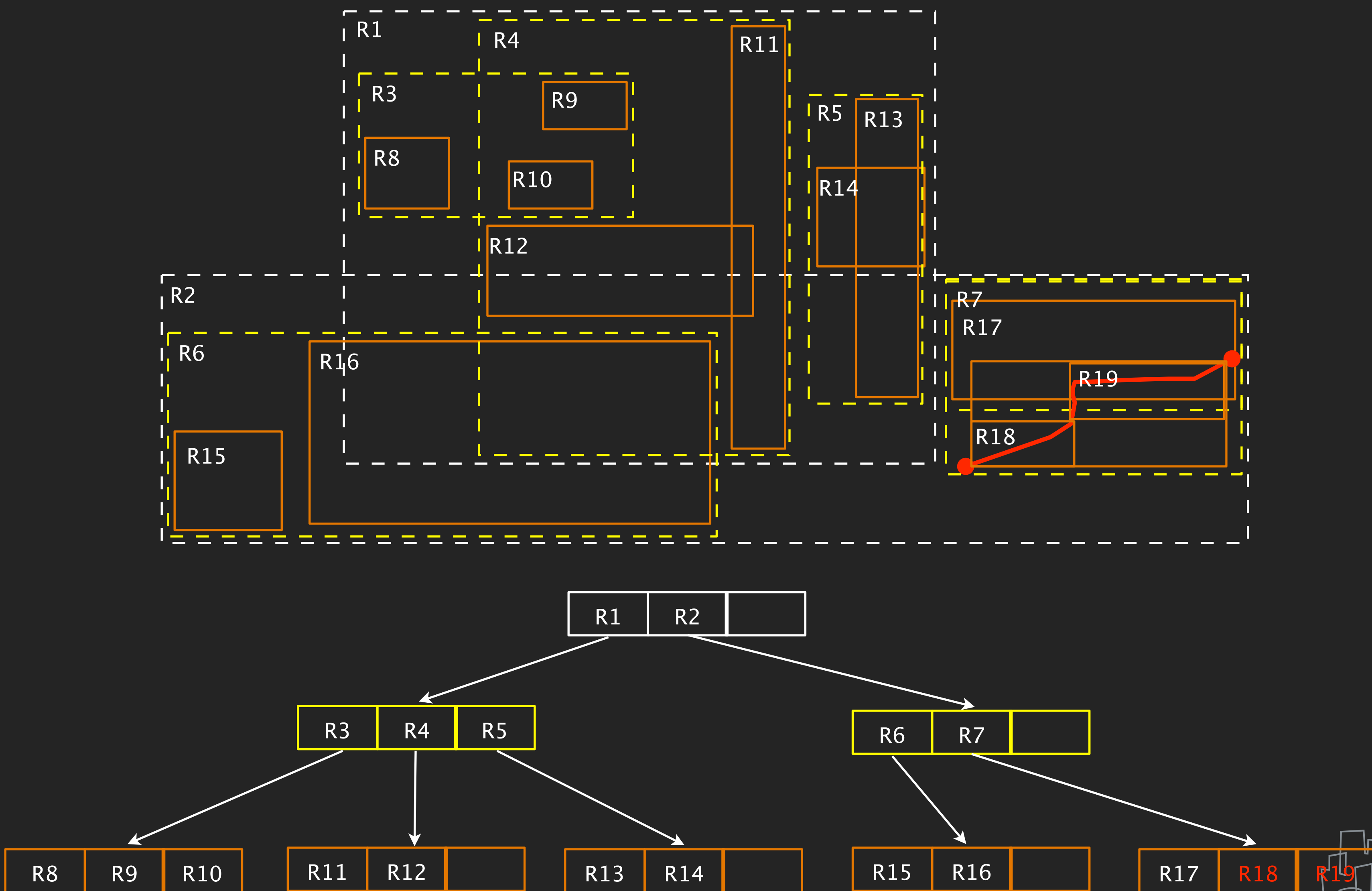
CSAIL

Outline

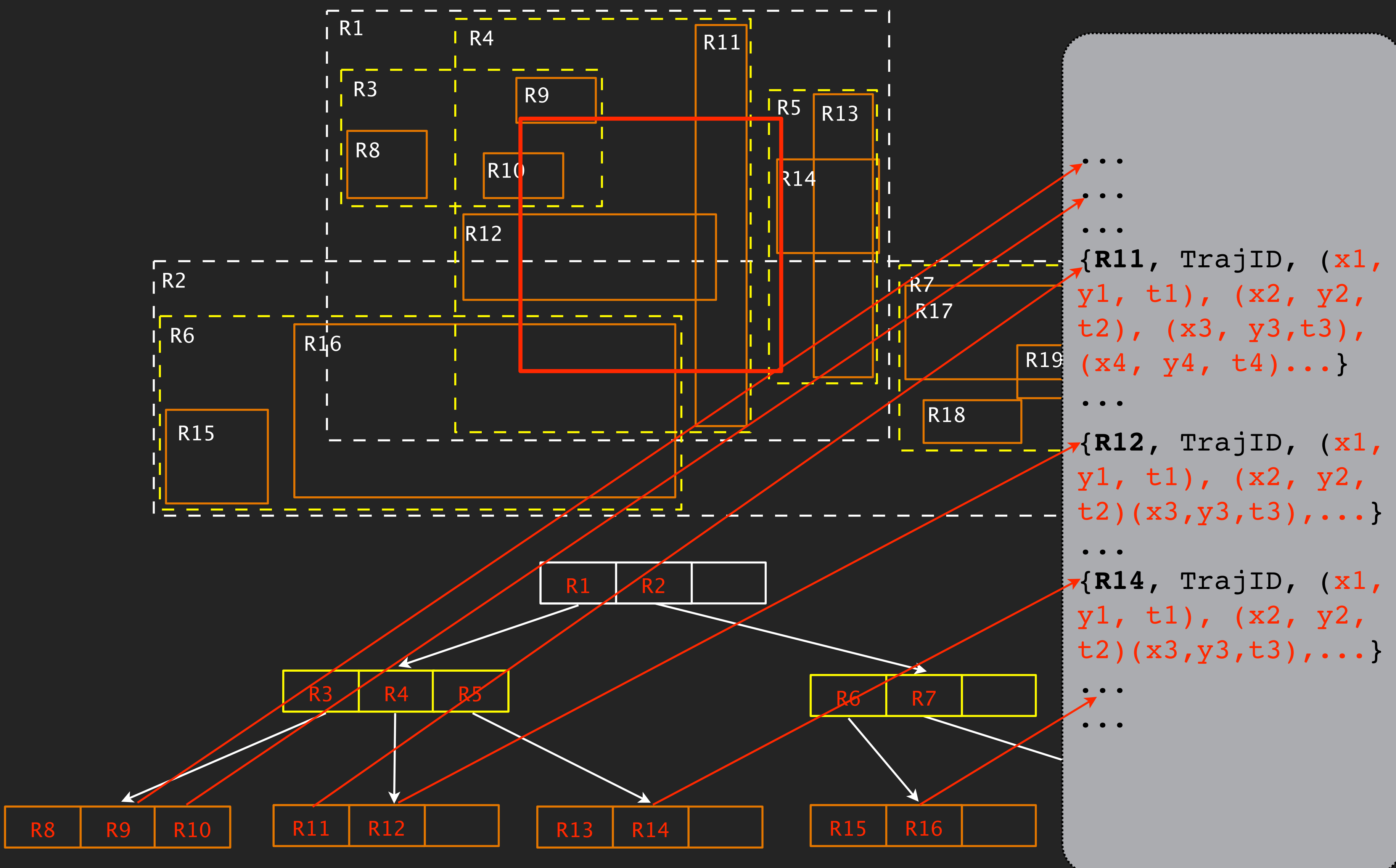
- Motivation
  - *Large-Scale GPS Data Mining*
- Conventional approach
  - *R-Trees & Trajectory-Segmentation*
- TrajStore
  - *Architecture*
  - *Sparse Spatial Index*
  - *Adaptivity*
  - *Compression*
- Performance
- Conclusions

CSAIL

Inserting [Conventional Approach]



Querying [Conventional Approach]



CSAIL

Issues with Current Systems

- Efficient for sparse data only
- Catastrophic for large, dense, overlapping data
  - Slow inserts
    - Bounding boxes creation
    - Multiple index updates per new trajectory
  - Slow queries
    - Index considers a very high number of overlapping objects
    - Inefficient *selects* of records

➡ Complex index maintenance & look-up

➡ One disk seek for each trajectory sub-segment

➡ Several minutes/hours to resolve aggregate queries

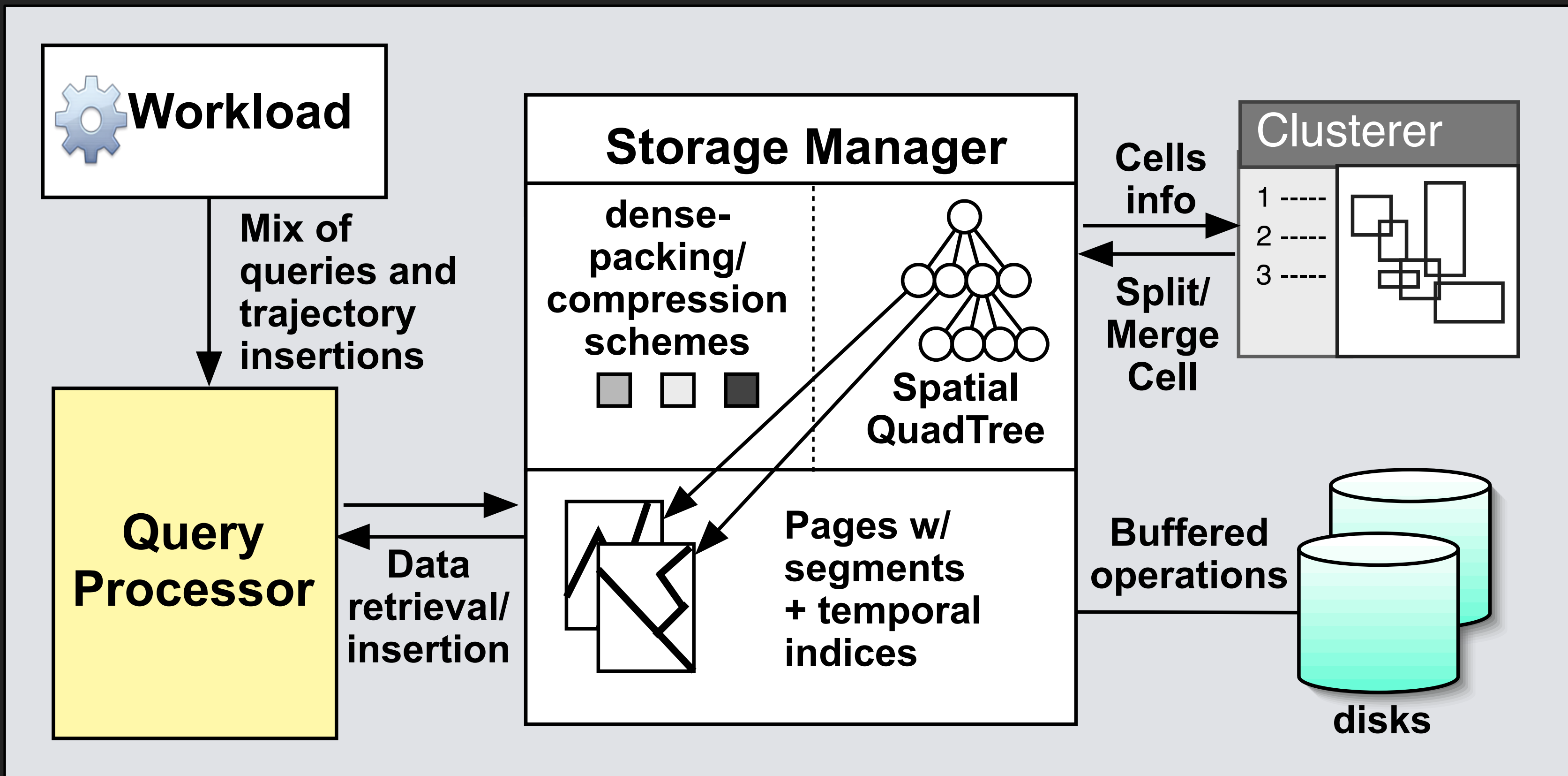
TrajStore

- Adaptive system to store & query very large trajectory data sets
  - Sparse, non-overlapping spatial index
  - Chunk-based data organization
    - co-location, dense-packing & compression
  - Buffered, amortized IO operations

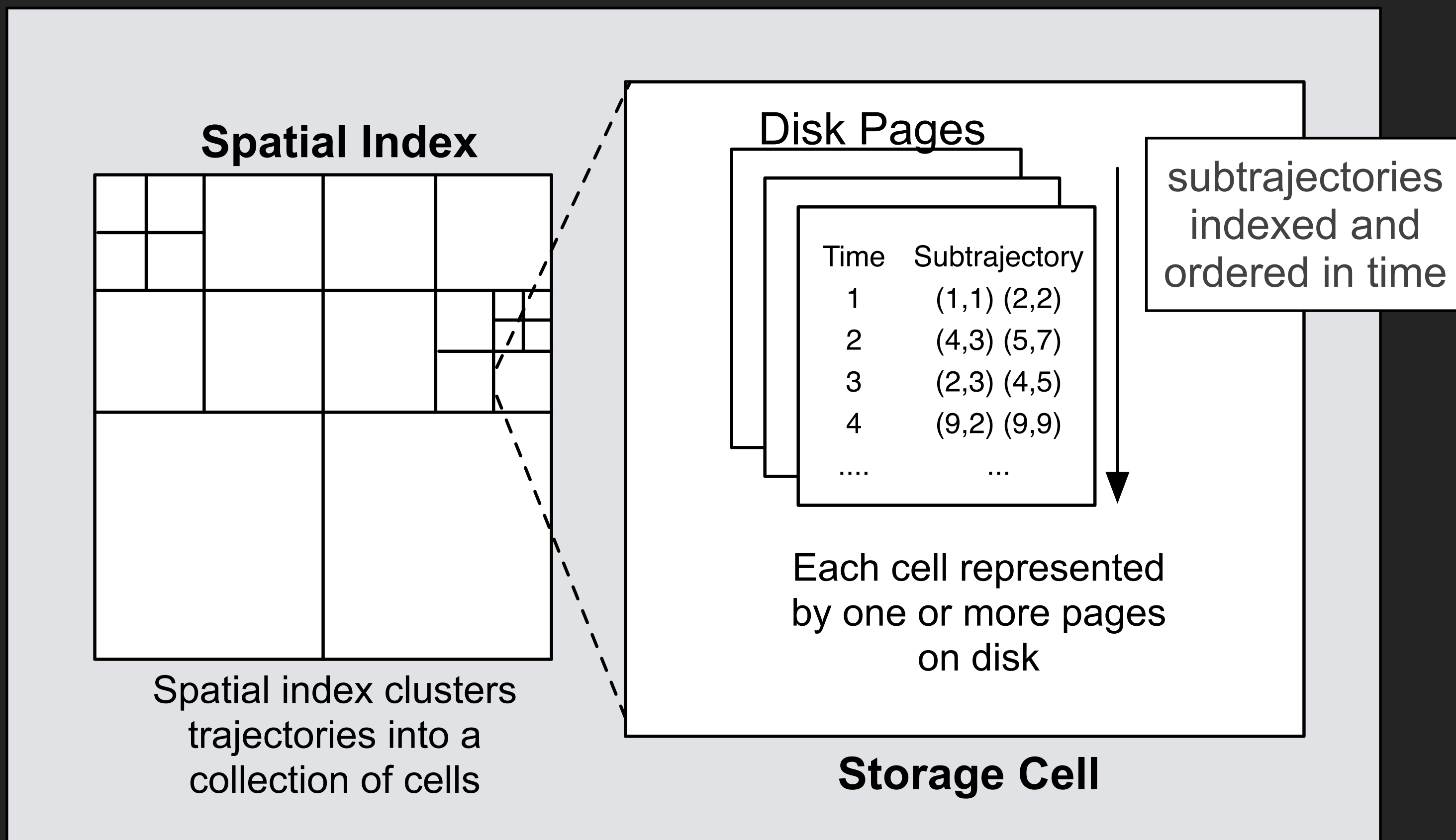
➡ Minimization of total IO cost

CSAIL

Architecture



Index & Storage

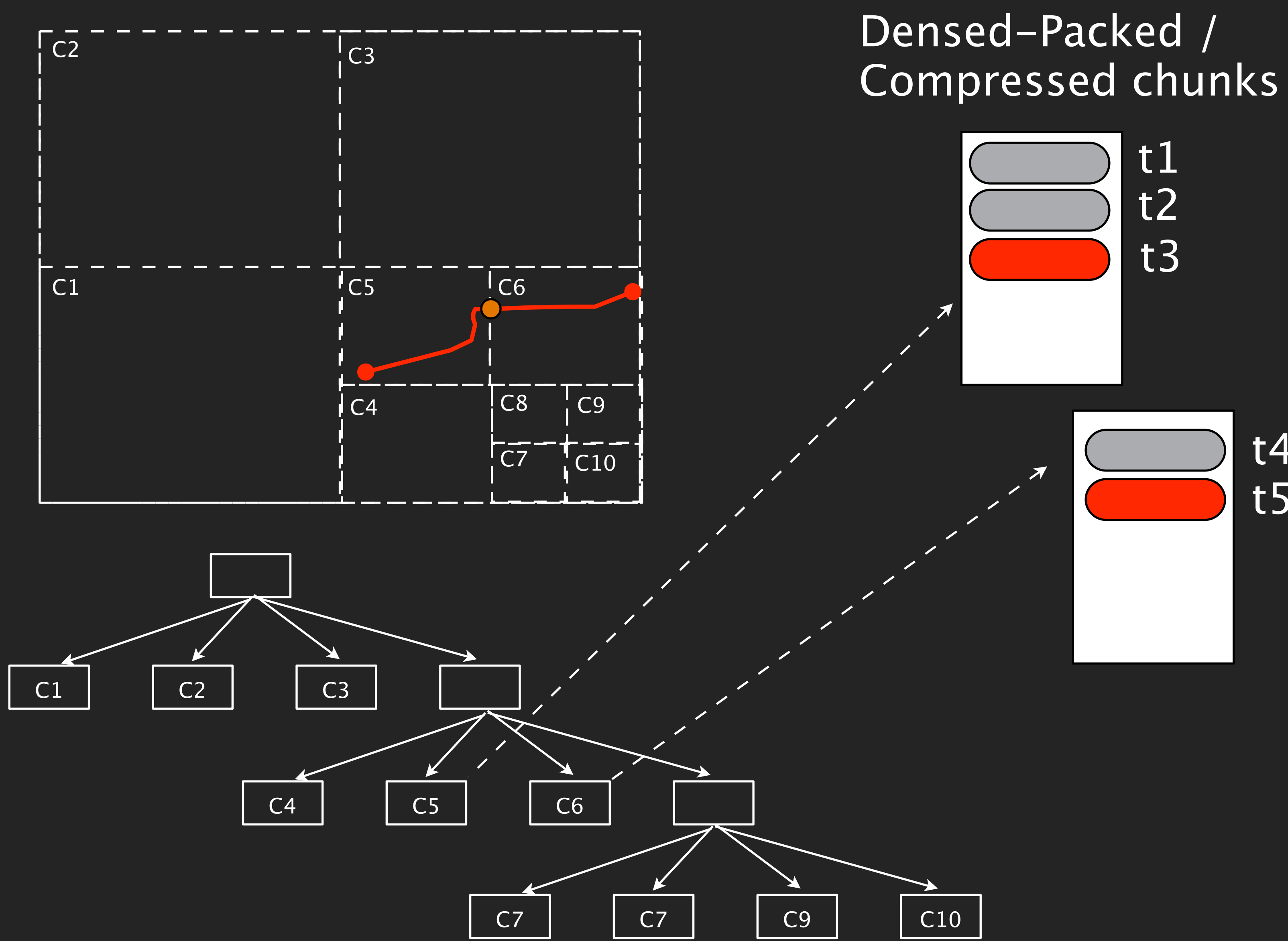


- Spatial index: quadtree

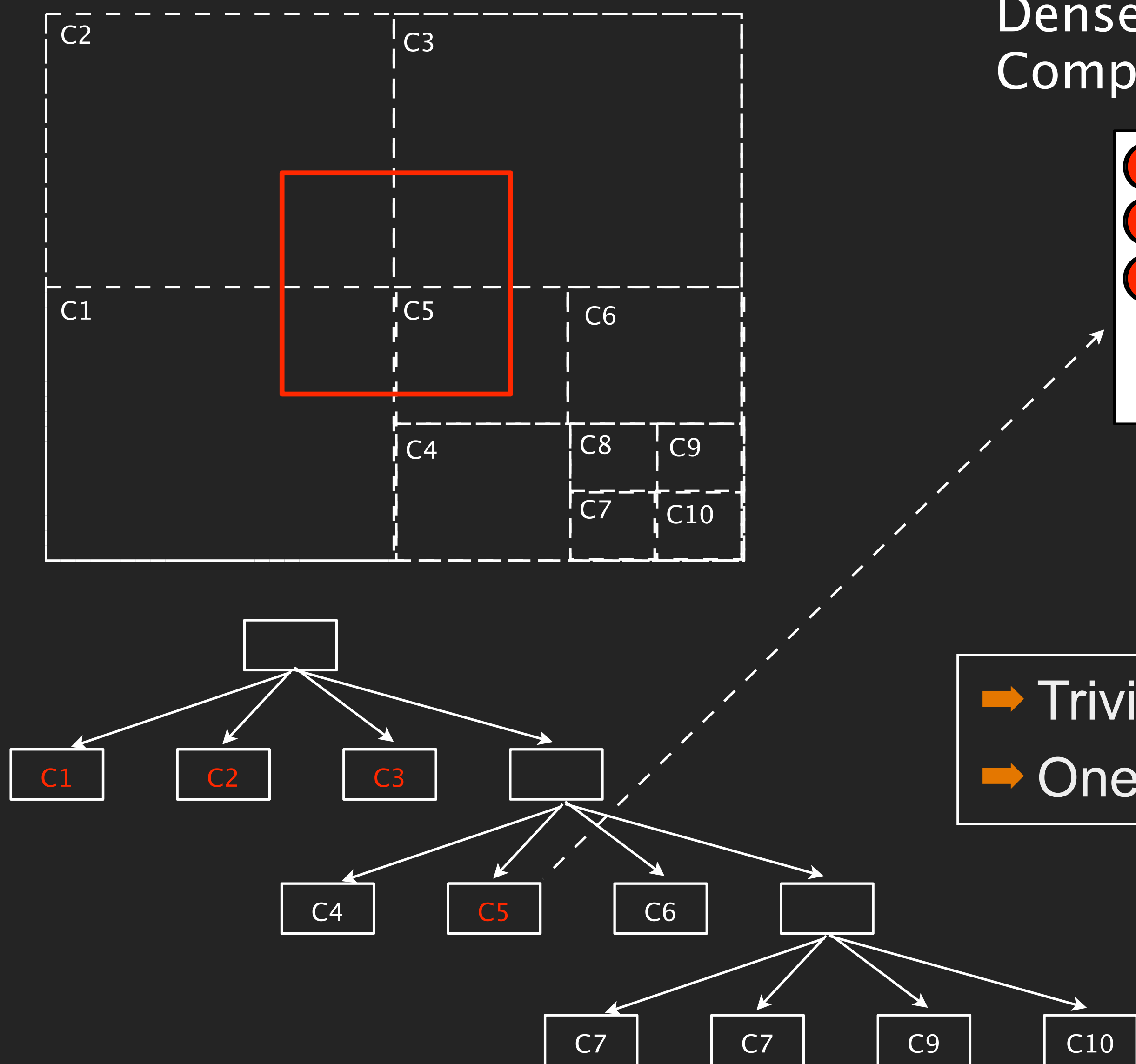
[new] Optimal quadtree construction

[new] Adaptive, index-driven data storage

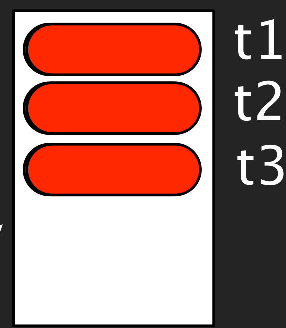
TrajStore Inserts



TrajStore Queries



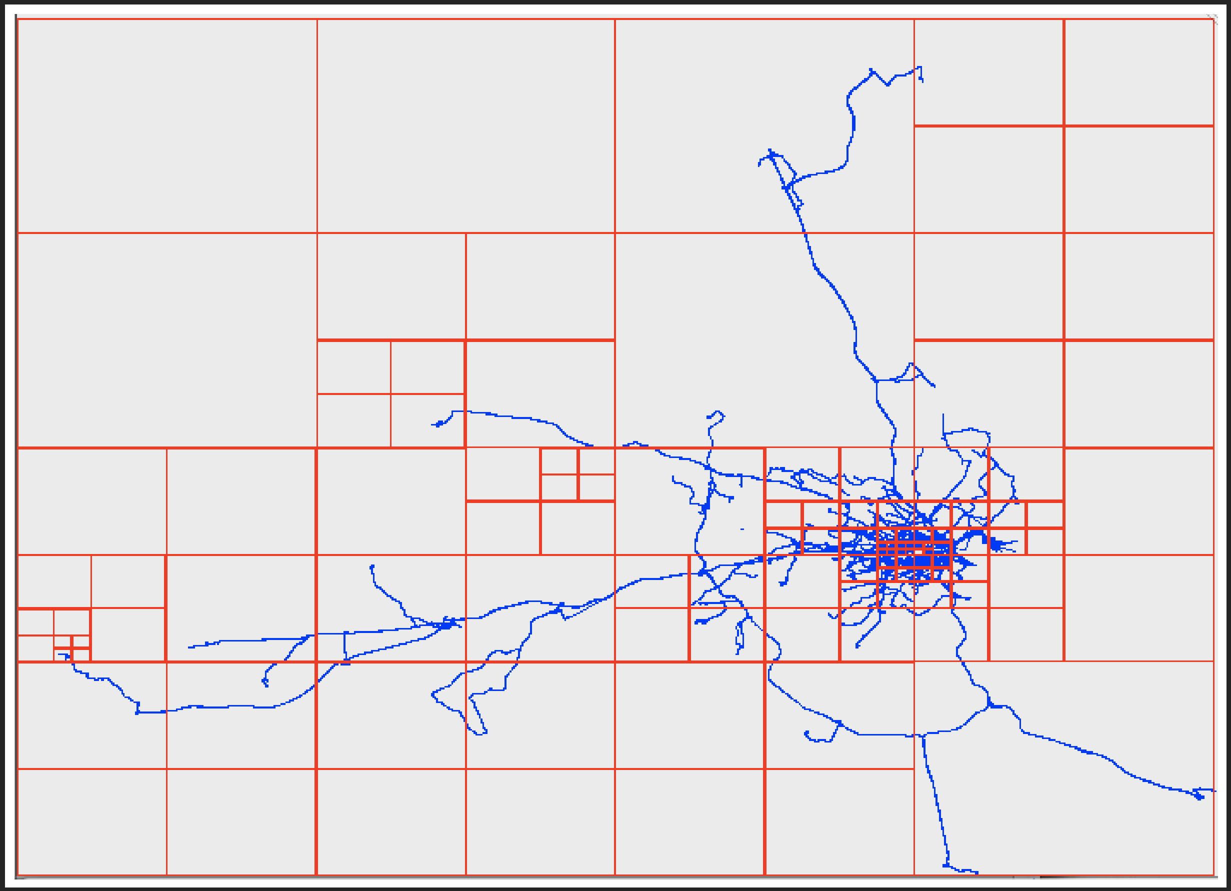
Densed-Packed /
Compressed chunks



- ➡ Trivial index look-up
- ➡ One seek per cell

Sparse Spatial Index (1/2)

- Cost-based, optimal spatial partitioning
  - Efficient, hierarchical partitioning



CSAIL

Sparse Spatial Index (2/2)

- Basic idea
  - Cost-model for query execution times based on #cells accessed
  - Optimal quadtree construction based on cost-model, query workload, local density & page size
    - $cellSize_{opt}(\mathcal{Q}, \mathcal{D}, pageSize)$
- Optimal balance between
  - Oversized cells
    - potentially retrieves data that is not queried
  - Undersized cells
    - seek not amortized if too little data read
    - unnecessary seeks if dense data and relatively large query

CSAIL

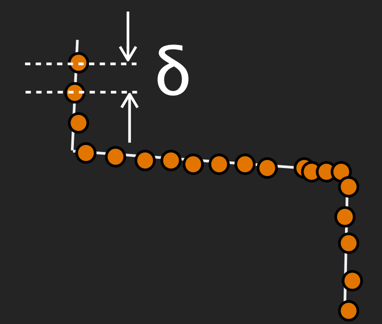
System Adaptivity

- Data evolution
  - Adapt the index & storage with every incoming trajectory
    - No-op / Split() / Merge()
    - Very fast, incremental operations
- Query evolution
  - Highly-skewed queries in practice
  - Per-cell query statistics
  - EWMA-based re-clustering

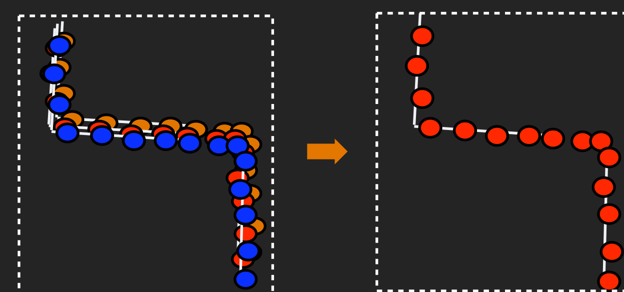
CSAIL

Compression

- Unique opportunities due to high spatial redundancy
  - Intra-segment redundancy
    - High-sampling rate, bounded speed
    - ➡ Delta encoding (lossless)
    - ➡ Linear interpolation (lossy/lossless)



  - Inter-segments redundancy
    - Repeated trips
    - Spatially constraint by roads, paths
    - ➡ Online cluster-detection
    - ➡ Cluster compression (lossy)



  - Combination of approaches based on user needs
    - Bounded total error

CSAIL

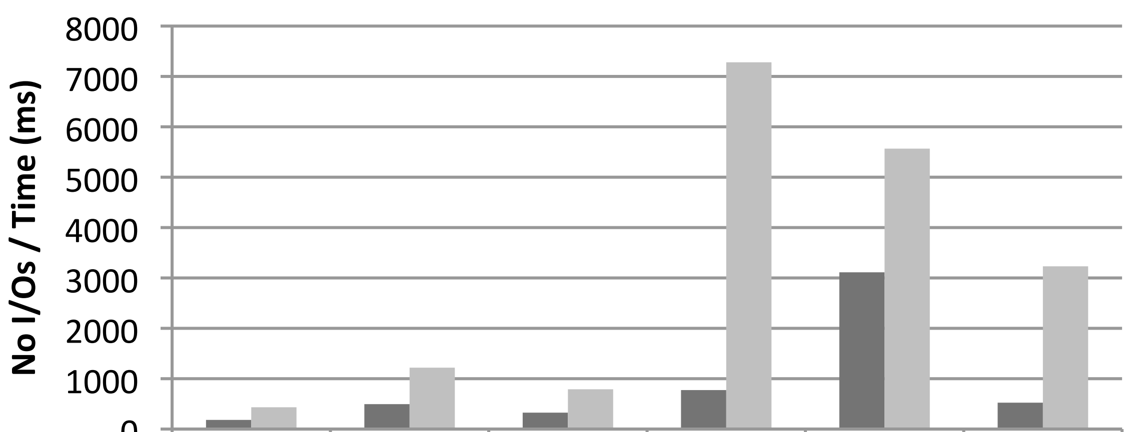
Experimental Setup

- Query answering on 40-200M GPS readings
  - CarTel data
  - Large queries (0.1% / 1% / 10%)
- Approaches compared
  - PostGIS
  - Optimal trajectory segmentation
  - TrajStore
- TrajStore variants
  - Fixed grid
  - Capacity-bound quadtree
  - Compression schemes

CSAIL

Experimental Results (1/2)

- Blazing fast query execution
  - 1 - 2 orders of magnitude faster than existing approaches
- Superior indexing scheme



	Adapt.	Grid1/1	Grid1/2	Clust. Split	NoSplit	Capacity Quad
No. I/Os	182	495	326	774	3112	524
Runtime	433	1219	790	7281	5567	3231

■ No. I/Os ■ Runtime [query size = 1%]

adaptivity & compression turned off

Experimental Results (2/2)

- Further results
  - High-insert rate
    - 100K GPS points / s on average
  - Scalable
  - Very resilient to data & query evolution
    - ≠ fixed grid
  - Compression (1m)
    - 1:8 compression ratio
    - 2.5 performance improvement

➡ See paper for full results

CSAIL

Conclusions

- Explosion of location-aware devices & applications
  - Urgent need to support very large-scale GPS analytics
- TrajStore: rethink both index & storage layers in combination to provide
  - Sparse, adaptive, non-overlapping index
    - optimal w.r.t. IO cost-model
  - Index-driven data co-location
  - High compression ratios
    - intra + inter-segments compression

➡ System of choice for analytical queries over very large collections of trajectories

CSAIL