Real-Time Aggregation of High-Velocity OLAP Data

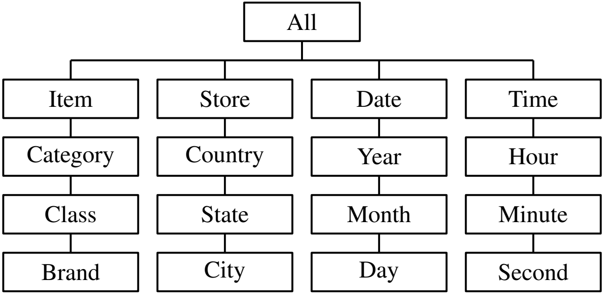
David E. Robillard

School of Computer Science
Carleton University

October 2016

Problem Domain

- OLAP data has many *dimensions* and one or more *measures*
- Dimensions ⇔ "Key", Measures ⇔ "Value"
- Dimensions are hierarcical



Some hierarchical dimensions for sales from the TPC-DS data set.

Goals

- Aggregate large fractions of data quickly
- Maximize throughput (high velocity), particularly insertion
- Support concurrent insertion and querying

Related Work

The Hilbert PDC tree is based on two key ancestors:

- PDC-tree[1]
- Hilbert R-tree[2]

[1] Frank Dehne and Hamdireza Zaboli. "Parallel real-time OLAP on multi-core processors". In: *Proc. 12th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing*. 2012, pp. 588–594.

[2] Ibrahim Kamel and Christos Faloutsos. "Hilbert R-tree: An Improved R-tree Using Fractals". In: *Proc. 20th Int. Conf. on Very Large Data Bases*. 1994, pp. 500–509. ISBN: 1-55860-153-8.

R-tree

- Classical data structure for geometric data
- Nodes have a Minimum Bounding Rectangle key
- Key contains the key of all child nodes
- Typically high-fanout, 1 leaf node per data element
- Many variants

PDC-tree

- R-tree-like structure which replaces MBRs with MDSs
- Overlap-minimizing split algorithm
- Supernodes
- Scales to many more dimensions than R-trees
- Multi-thread support with minimal locking

Hilbert R-tree

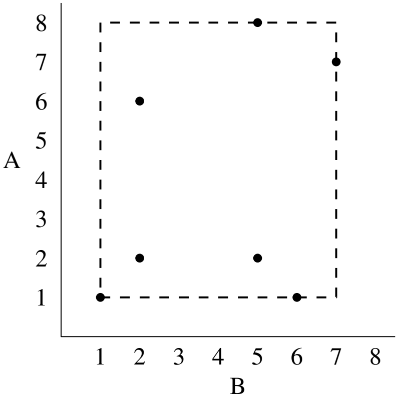
- R-tree that uses Hilbert order for insertion
- Avoids geometric calculation during insertion
- Improves insertion throughput considerably
- Locality preserving properties of Hilbert mapping maintains good query performance

Hierarchical IDs

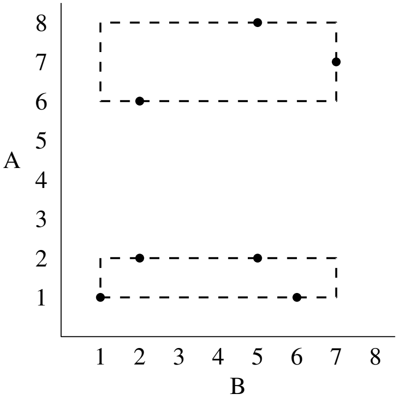
- IDs are stored in integers
- Self-contained ID contains index at all levels
- Improves DC-tree scheme by avoiding dictionary lookups
- IDs can be viewed at a higher level with simple bit masking

Dimension	Level 1	Level 2	Level 3	Level 4

Minimum Describing Subsets



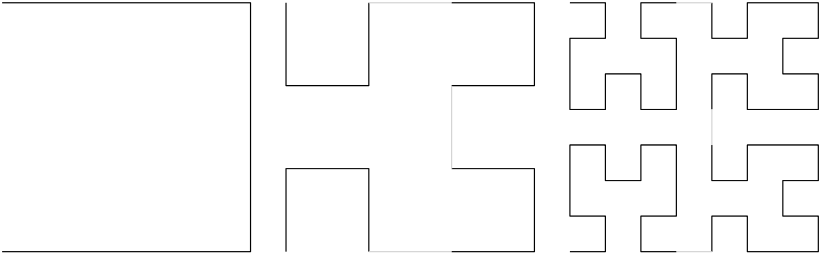
MBR $[A1, A8]$,
$[B1, B7]$



MDS $\{A1, A2, A6, A7, A8\}$,
$\{B1, B2, B5, B6, B7\}$

The Hilbert Curve

- Fractal space-filling curve
- Locality preserving



The first three iterations of a 2D Hilbert curve construction.

Hierarchical Hilbert Mapping

- Using Hilbert order requires mapping hierarchical IDs
- Mapped IDs are at the bottom level of dimension hierarchies
- Dimension hierarchies may have uneven distribution
- Naïve solution may not work well since directory node keys are at higher levels

Mapping Schemes

	Dim	Level 1	Level 2	Level 3	Level 4
ID	01	1	11	111	1111
	10	11	1	1	11
Direct	01	01	0011	0111	1111
	10	11	0001	0001	0011
Dimensionless	00	01	0011	0111	1111
	00	11	0001	0001	0011

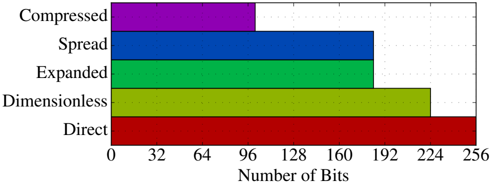
	Pad	Level 1	Level 2	Level 3	Level 4
Spread	00000	01	11	111	1111
	00000	11	01	001	0011
Expanded	00000	10	11	111	1111
	00000	11	10	100	1100

Compressed Hilbert Mapping

The compressed mapping removes all unused bits and does not preserve hierarchical structure across dimensions.

	Pad	Levels
Compressed	000000	1111111111
	000000	0011111111

Hilbert Bits

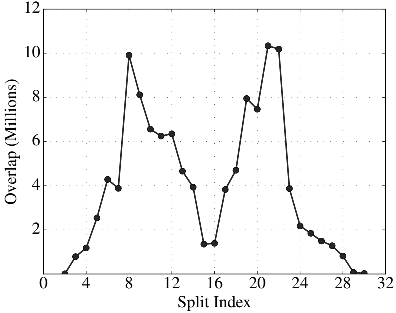


Number of bits used for various Hilbert mappings.

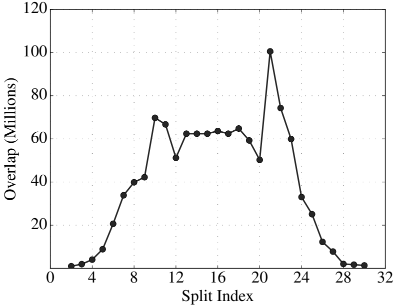
Node Splitting

- Order of child nodes is fixed due to Hilbert ordering
- PDC-tree split algorithm not applicable
- Hilbert R-tree balanced split may result in high overlap
- Overlap is much more expensive than imbalance for aggregation
- Solution: choose split index based (primarily) on overlap in linear time
- Create supernode if no good split index is found

Split Overlaps



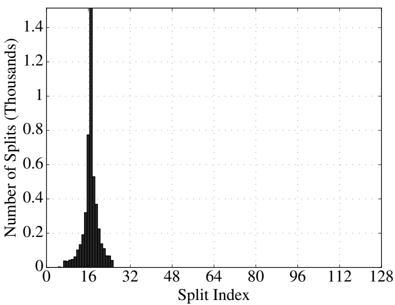
An obvious split, but with overlap.



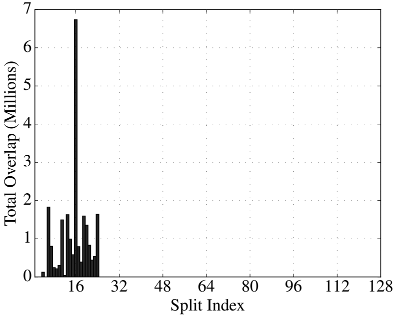
A dificult node to split.

Overlap at each split point in observed directory nodes.

Split Frequency



Distribution of split positions

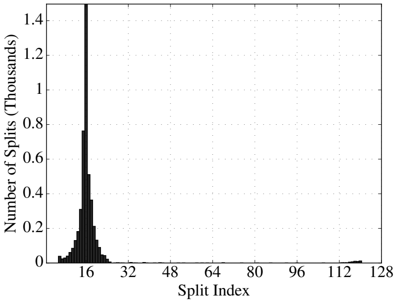


Total resulting overlap

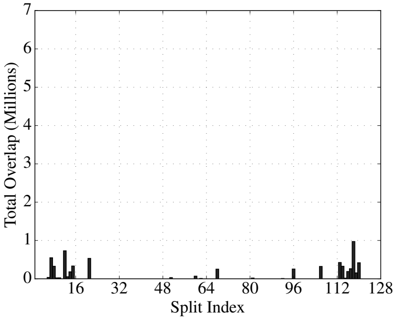
Split index frequency and overlap with fixed maximum fanout.

Supernode Split Frequency

- Supernodes are created if no good split index is found
- Due to multi-threading, if maximum size is reached, force split



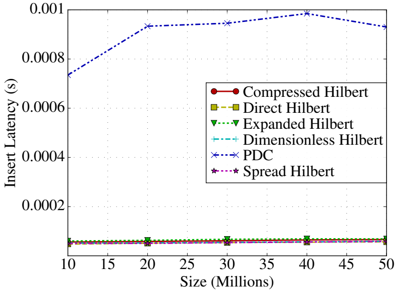
Distribution of split positions



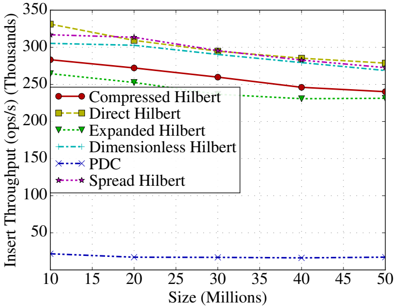
Total resulting overlap

Split index frequency and overlap with supernodes.

Insertion Performance



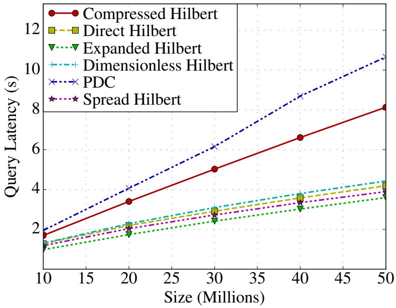
Insert Latency

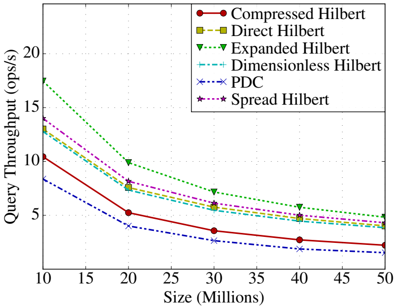


Insert Throughput

Performance with a stream of inserts.

Query Performance

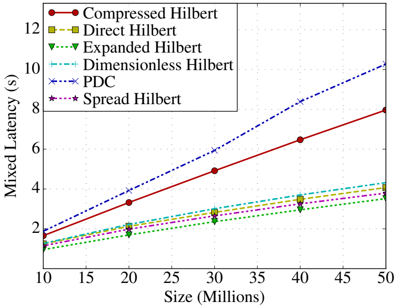


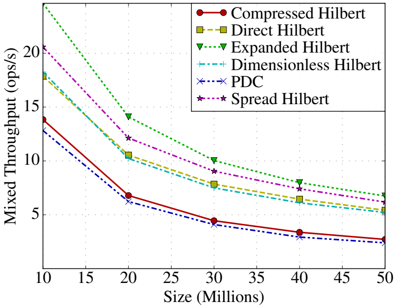


Query Latency Query Throughput

Performance for a stream of queries.

Mixed Performance



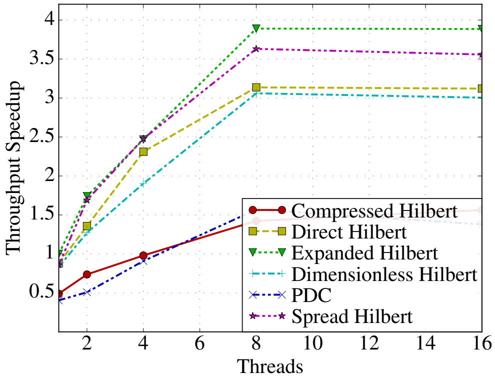


Mixed Latency

Mixed Throughput

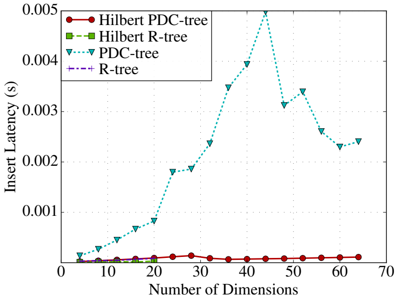
Performance for a mixed stream of 50% inserts and 50% queries.

Speedup

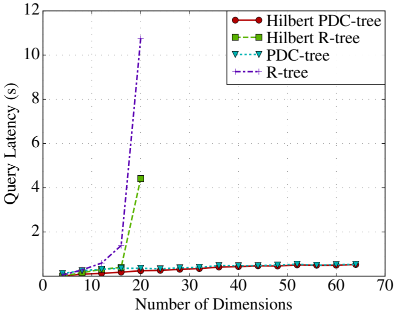


Speedup for a mixed stream of inserts and aggregate queries.

Many Dimensions



Insert latency



Query latency

Latency as number of dimensions is increased.

Benefits and Use Cases

- The Hilbert PDC-tree is a data structure for real-time aggregate queries on high-velocity data
- Key benefits:
  - Much higher ingestion throughput
  - Scales well to many hierarchical dimensions
- Used as the foundation of VOLAP
  - A fully distributed system to support the same data model
  - Distributes many Hilbert PDC trees across any number of worker nodes
  - Server nodes coordinate and provide a similar insertion/query model to the tree itself

Thank you