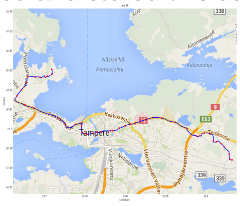
Tips and Tricks for Map-Reduce and Big Data Databases

Chalmers & Gothenburg Uni, 18.5.2017

Jyrki Nummenmaa
Faculty of Natural Sciences
University of Tampere, Finland

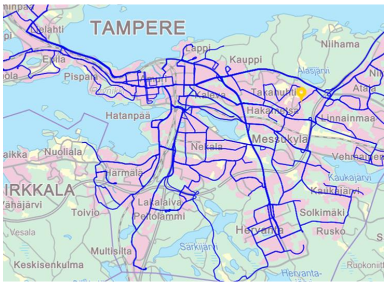
Tampere – Background info

- Tampere is a third largest city in Finland and Tampere region is the largest outside of capital area with a population a little over 500 000 (small in Chinese scale)
- There are about 150 buses in traffic during daytime
- Buses have GPS sensors. Locations are sent to background system
- Background system shares bus locations once a second through internet.
- We have collected and stored this data for analysis for over 2 years.



Tampere Bus Location Data

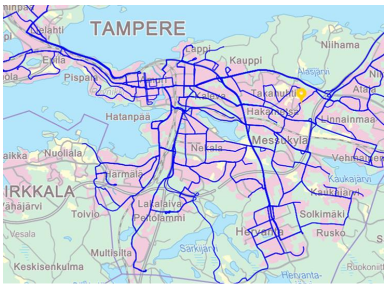
- Real-time data stream from Tampere public transportation bus fleet
  - > 100 vehicles
- in SIRI format
- Updates every second
- Includes e.g.
  - GPS location
  - Line number, direction and departure
  - Delay



UNIVERSITY OF TAMPERE

Tampere Bus Location Data

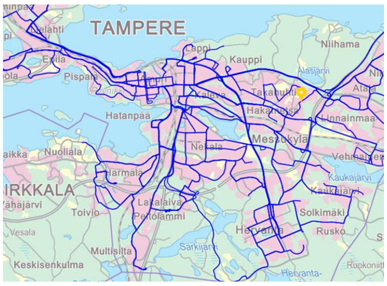
- Real-time data stream from Tampere public transportation bus fleet
  - > 100 vehicles
- in SIRI format
- Updates every second
- Includes e.g.
  - GPS location
  - Line number, direction and departure
  - Delay



UNIVERSITY OF TAMPERE

Data Quality Problems

- Service breaks with no data (not very often)
- Connection to bus lost or some other techical problem
  - Shows the same position
  - Last time the position was recorded is shown
- Buses that are not in any busline are included
- GPS accuracy

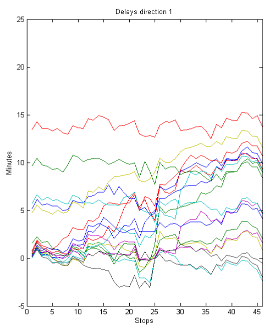


UNIVERSITY OF TAMPERE

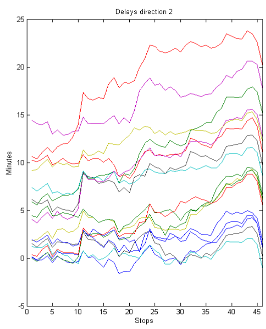
Bus delays

- Delay is a difference between timetable and arrival time
- Delay is calculated and included in the data (every second)
- For example, see below delay on Route 16 at bus stops during daytime hours
- If a bus starts late it will be late at the end
- Delay increases in the city center and some intersections

Direction 1



Direction 2



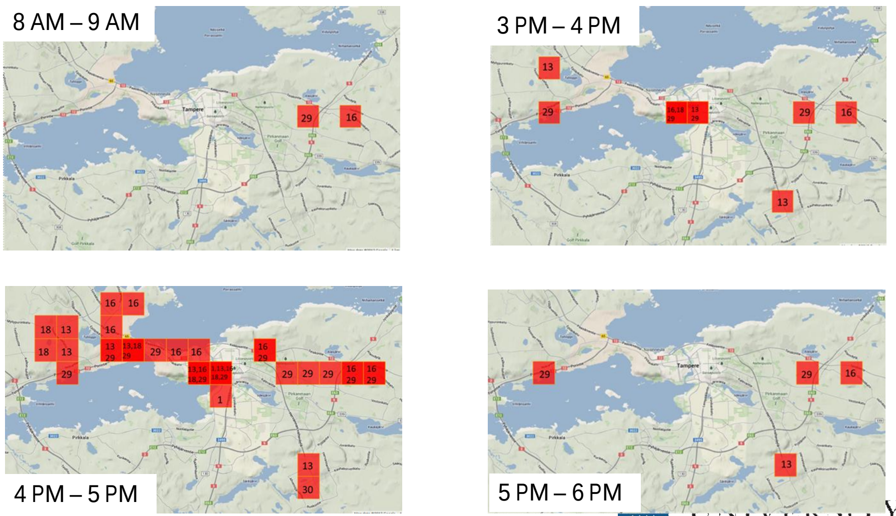
UNIVERSITY OF TAMPERE

Bus delay analysis

- From stored bus location data, analyze the traffic fluency
  - From all the observations with delay>5min, use frequent itemset mining to find the lines, locations and times of most regular delays
- Compare a large set of delayed journeys to not delayed journeys to find out the bottlenecks along the bus routes
  - Take the best and worst quartiles and compare to find out the bottlenecks

UNIVERSITY OF TAMPERE

Step 1: Where, when and on which lines do the delays typically occur?



UNIVERSITY OF TAMPERE

Route planner

- The best thing since public transportation was invented?
- You want to get from place A to place B – the route planner calculates the necessary connection, tells you when to leave, where to change the bus, etc.
- ”What could go wrong”?

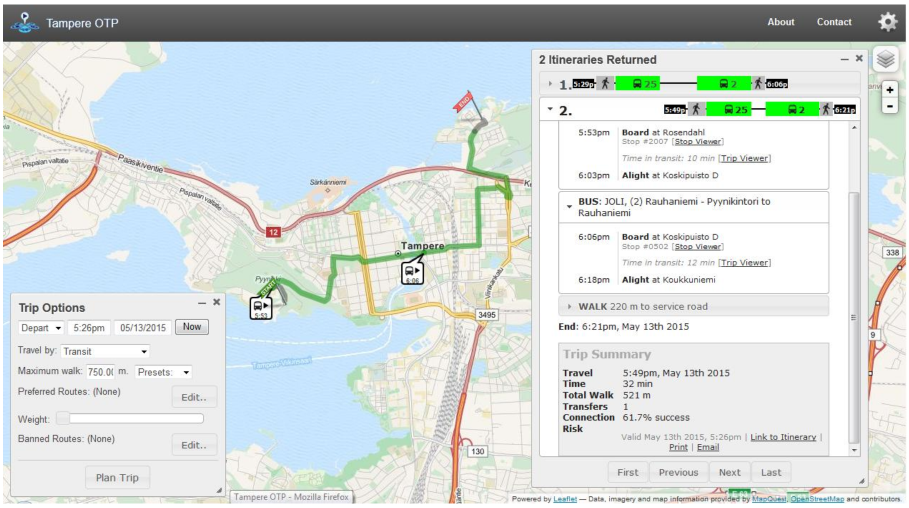
Route suggestions: Laulunmaa, Tampere - Runkokatu, Tampere Tuesday 31.12.2013

Later

Dep.	Route	Arrival	Duration	Total walking
15:14	0.1 km – Laulunmaa 15:15 (30) 15:28 – 0.1 km – Koskipuisto 15:40 (18) 16:07 – 0.1 km	16:09	54 min	0.3 km
15:04	0.1 km – Laulunmaa 15:05 (30) 15:18 – 0.1 km – Koskipuisto 15:30 (18) 15:57 – 0.1 km	15:59	54 min	0.3 km

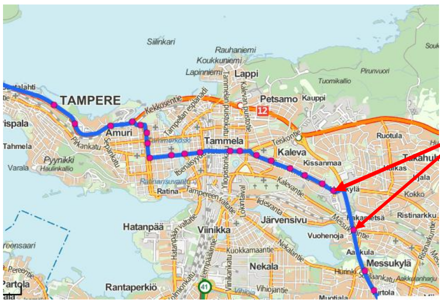
UNIVERSITY OF TAMPERE

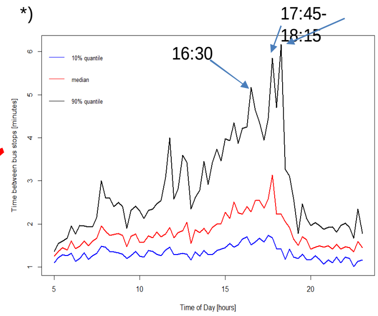
On-line prediction service



UNIVERSITY OF TAMPERE

"Prisma" junction, left turn, bus line No 3





*) Computed from a sample of 5744 observations on 76 working days / winter 2014-2015, with 15 minute time resolution

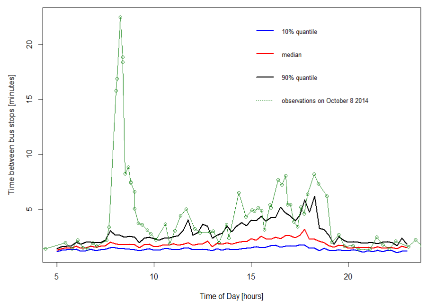
80 % of the traveling times fit between the blue and black lines.

Half of the traveling times are below and half above the red line.

UNIVERSITY OF TAMPERE

Exceptional case

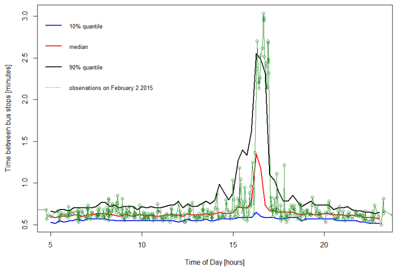
- Accident in the junction on October 8th 2014 at about 8AM jammed the traffic
- The traveling time in the junction raised to about 10-fold compared to a normal morning traffic
- (In addition, we can see that the traffic signal settings are probably different at the time when the model was built compared to October 8th => the model must be continuosly updated)



UNIVERSITY OF TAMPERE

Daily peak “Pispalan valtatie”

*)



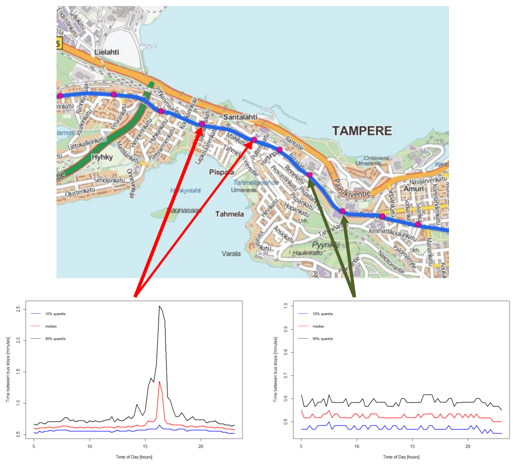
*) Computed from 28002 observations on 76 days in winter 2014-2015 with 15 minute time resolution

Green: Monday 2nd February 2015

UNIVERSITY OF TAMPERE

Monitoring Tampere traffic

- All the ~2000 between-busstops-segments in Tampere can be **automatically profiled** using bus history data to get the ”normal profile”
  - E.g. with 30 min resolution, the model is a table of ~60000 rows and ~5-10 columns of numerical information
  - Fits easily in main memory
- From the normal profile, we can find the **interesting links** that contain some regular peaks
- All the profiles can be used in real time to **detect exceptional traffic situations**



Interesting profile

Not interesting profile

UNIVERSITY OF TAMPERE

Computational challenges

- Fast analysis for bus changes and traffic status using latest data
- This analysis needs models that have to be computed using the collected data
- Relatively high volume of incoming data
- We need *scalable* solutions
  - To work in a city of any size
  - Supporting both fast on-line analysis on the latest data, and massive background processing

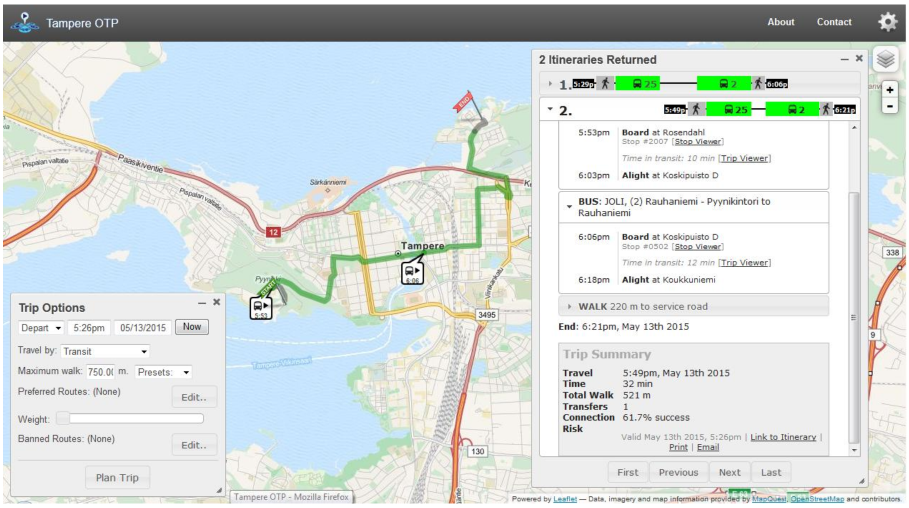
UNIVERSITY OF TAMPERE

Traffic data and databases

- We started storing the data in an SQL database but soon found out it was not a good solution. At least the way we did it.
- Now we store data in HDFS in files, and in Hbase.
  - Both can be used for MapReduce
  - We compute statistical models etc using MapReduce.
  - Mapping can be done based on e.g. geographical area etc.
- Hbase is used to access the latest data.
  - If primary key starts with timestamp, then the data is physically ordered by time – once the start of the data has been found, it is very fast to access the latest data.

UNIVERSITY OF TAMPERE

On-line prediction service



UNIVERSITY OF TAMPERE

Compute parameters: MapReduce

Each night, for yesterday (note that we search data by timestamp):

1. Map: by key (line, direction, origin, destination, departureTime, data) - additionally only arrivalTime and position are needed from the raw data (~800 Mb per day, we use 60 days, but every night just the last day's data is calculated – previous data does not change.)
2. Reduce: Find the information of the bus stop, and for each stop, find for each bus the arrival and departure time (a bit of calculations with the coordinates are involved.
3. Store the resulting stopCode, arrivalTime, departureTime data (~5 Mb)

Now for the latest 60 weekdays (~300 Mb or arrival and departure data)

1. Map: by (line,direction,origin,destination,originDepartureTime,stopCode)
2. Reduce: compute the distributions
3. Results are saved for on-line prediction (~10 Mb)

UNIVERSITY OF TAMPERE

MapReduce principles

- Map processes are completely independent of each other, once they are started.
- Map results are combined in the Reduce step.
- After that you can do subsequent MapReduce rounds.
- In the previous example, this was done because of different data sets
  - First MapReduce yesterday's data.
  - Then merge it with the 59 days before yesterday, but on another granularity level.
- Let's check a couple of more cases…

UNIVERSITY OF TAMPERE

Frequent Sequence Mining

- Sequences can be thought of as strings.
  - Special case of Frequent Itemset Mining.
- There are different mining tasks, most likely is to check for ”common patterns” in strings.
  - E.g. ”AB” appears in 50% of {”ABC”,”AAB”,”ACB”,”CCC”}
  - And in 75% if we allow *gaps* of length 1.
- Support of pattern p is the number of strings containing p divided by the number of all strings.
- We want to find top-k of those patterns (e.g. 20 with biggest support).
- How to MapReduce?

UNIVERSITY OF TAMPERE

Frequent Sequence Mining / 2

- It is possible to distribute the computation of support into MapReduce tasks. This makes sense, if the data set is huge.
- However, usual ”generate-and-test” strategies generate candidate patterns and use some *pruning rules.*
  - A typical pruning rule follows the a priori principle. A sequence cannot have a bigger support than any of its subsequences.
  - While generating the candidates you can maintain the top-k candidates and right off reject the sequences whose support is not enough, and all of their supersequences.
  - But to compute the support I need the whole dataset!
  - What can I do?

UNIVERSITY OF TAMPERE

Frequent Sequence Mining / 3

- If the whole dataset fits in memory, it is possible to distribute the ”candidate space” between the MapReduce processes and they can completely manage their candidate space (sequences starting with ’AB’ on one server, ’AC’ on another, etc...
- Usually the candidate space is so big that you can employ a sufficient amount of servers with this strategy.
- If the whole dataset *does not* fit in memory, then each support computation round is parallel / distributed.

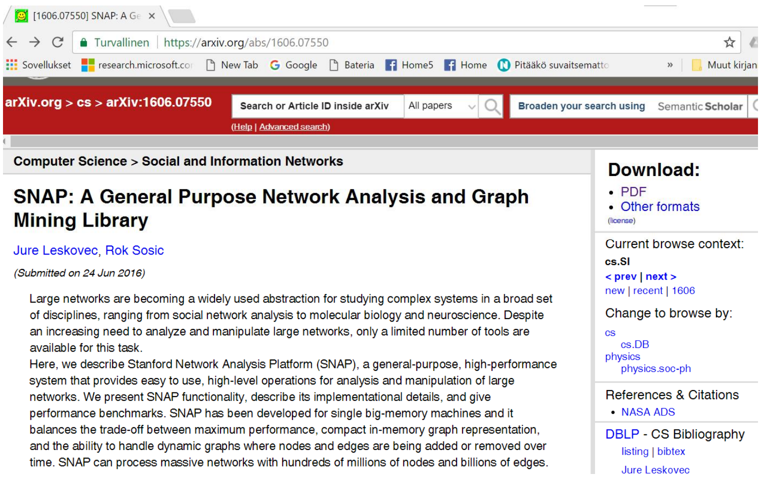
UNIVERSITY OF TAMPERE

Graph Algorithms

- Many graph algorithms are ”walking” around the graph in such a way that we cannot split the graph into suitable slices for MapReduce.
- This means that we are even worse of than in Frequent Sequence Mining – it seems incredibly complicated to write algorithms that operate on ”slices” of a graph.

UNIVERSITY OF TAMPERE

Stanford to rescue



[1606.07550] SNAP: A Ge

Turvallinen https://arxiv.org/abs/1606.07550

Sovellukset research.microsoft.co New Tab Google Bateria Home5 Home Pitääkö suvaitsematto » Muut kirjan

arXiv.org > cs > arXiv:1606.07550

Search or Article ID inside arXiv | All papers | Broaden your search using Semantic Scholar

(Help | Advanced search)

Computer Science > Social and Information Networks

SNAP: A General Purpose Network Analysis and Graph Mining Library

Jure Leskovec, Rok Sosic

(Submitted on 24 Jun 2016)

Large networks are becoming a widely used abstraction for studying complex systems in a broad set of disciplines, ranging from social network analysis to molecular biology and neuroscience. Despite an increasing need to analyze and manipulate large networks, only a limited number of tools are available for this task.

Here, we describe Stanford Network Analysis Platform (SNAP), a general-purpose, high-performance system that provides easy to use, high-level operations for analysis and manipulation of large networks. We present SNAP functionality, describe its implementational details, and give performance benchmarks. SNAP has been developed for single big-memory machines and it balances the trade-off between maximum performance, compact in-memory graph representation, and the ability to handle dynamic graphs where nodes and edges are being added or removed over time. SNAP can process massive networks with hundreds of millions of nodes and billions of edges.

Download:
- PDF
- Other formats

(license)

Current browse context:
cs.SI
< prev | next >
new | recent | 1606

Change to browse by:
cs
cs.DB
physics
physics.soc-ph

References & Citations
- NASA ADS

DBLP - CS Bibliography
listing | bibtex
Jure Leskovec

UNIVERSITY OF TAMPERE

Stanford maintains an interesting Large Network Dataset Collection

From Facebook, Wikipedia, Amazon reviews, etc.
Distribution of graph sizes in the library according to the article

Size (no of edges)	No of graphs
< 0.1 M	18
0.1M – 1M	24
1M – 10M	17
10M – 100M	7
100M – 1B	4
>1B	1

UNIVERSITY OF TAMPERE

SNAP RAM memory consumption

- Node: 54.4 bytes (obviously average)
- Edge 8.3 bytes (obviously average)
- For 1024 GB RAM, SNAP can represent grpahs with 123.5 billion edges.
- According to the network library statistics on previous slide, this seems sufficient in practice.

UNIVERSITY OF TAMPERE

MapReduce and Graphs

- There are still computations that require various rounds and benefit from MapReduce
- Consider a "centrality" of node v based on how many shortest paths between a pair of nodes goes via v.
- What can be done with MapReduce?
  - For a shortest path between (v1,v2) you need the whole graph.
  - BUT you can distribute the input parameter space (node pairs) into different computations that get these parameters and the whole graph.

UNIVERSITY OF TAMPERE

Sampling

- Sometimes you may not or simply cannot scan through all of the data – at least not initially.
- Sampling can be used to estimate from the data
  - Remember that there is no clean big data.
  - There can always be an error.
  - Basically data mining implies you estimate things from the data.

UNIVERSITY OF TAMPERE

Execution parameters

- Consider e.g. Spark, which can be used to run MapReduce in main memory.
- Programmers' job is easy: Write a Map function and a Reduce function.
- The system takes care of the rest – really?
- Spark executes in ”containers” – let's see an example configuration for a small-scale system:

UNIVERSITY OF TAMPERE

Smal system "back home"

- 1 node. 24 CPU cores and 60 GB RAM
- 5 nodes -> 120 CPU cores and 300 GB RAM
- Spark configuration for 1 container: 2 cores CPU and 4GB RAM
- Total: 60 containers and 240 GB RAM
    - Some RAM needed for other things…
- When executing your work you specify the no of executors, and executor-cores & RAM
- *Check your data size!* (Spark prefers 500 MB)
- *Check the tasks (executors)! Too large -> high overhead.*
- Spark does not tune this for you.

UNIVERSITY OF TAMPERE

Big Data Databases –
A couple of points

UNIVERSITY
OF TAMPERE

Social media applications

- Lots of users
- Lots of data
- Each user has needs data from ”neighbours”
- Simple transactions
- Reliability not such a priority as e.g. in banking
- Approximate analysis should usually be ok
  - Topic trends, interest mining (for e.g. marketing purposes)

UNIVERSITY
OF TAMPERE

Internet shopping

- Lots of users
- Lots of data
- Users mostly just browse the items and maybe collect items into a local ”shopping cart”.
- Only when the users make a purchase, they change some database content
  - What if the shop does not have the products, because two customers try to buy them at the same time?
- Data mining and analysis for advertisments, suggestions, etc.

UNIVERSITY OF TAMPERE

What are databases?

- Reliable data storage
  - Tolerates failures
  - Protects your data
  - Little down-time
- Multiple concurrent users
- High-level query language
  - Interactive access and access from programs
  - However somewhat limited with complex JN1 and special data types
- Large amounts of data
- Classic choice: databases based on the idea of simple table (relational databases)

UNIVERSITY OF TAMPERE

JN1 Maybe mention that databases tend to go in for uniform structure, rather than ad hoc structures of programming applications
Jyrki Nummenmaa; 17.5.2017

Big Data databases are needed…

- …because our data is distributed in servers in different locations!
  - No. Distributed databases are nothing new. They were available ages ago and you can read about them in old database textbooks etc.
- …because there is so much data these days that we need to store!
  - No. There are traditional database products developed into Big Data scale and you can put your petabytes of data there.

UNIVERSITY OF TAMPERE

Big Data databases are needed…

- …because our data types are not like in the traditional database systems!
  - There are NoSQL (no SQL / not only SQL) databases that are not really for distributed large data sets.
- …because you want to program your own Big Data computations and not use the database query & computations facilities!
  - Yes. But note that some database products like Teradata give an option to integrate MapReduce & R into your data as well.

UNIVERSITY OF TAMPERE

Big Data databases are used…

- …because we do not want to pay very expensive licenses!
  - Definately. Now there are open source products that work really nicely with Big Data on Hadoop platform.
- …because the tables are growing ”wider” (more and more attributes)!
  - This is a part of the modern development anyway, and needs a different approach from the traditional one. However the standard database producs can handle vast amounts of attributes.

UNIVERSITY OF TAMPERE

Big Data databases are used…

- …because you have reasonably cheap hardware and want to benefit from Hadoop failure management!
  - Yes, this could be good motivation.
- …because you are willing to compromise the traditional database ”values” for scalable Big Data solutions!
  - You may be able to do that with traditional database systems as well.

UNIVERSITY OF TAMPERE

Distributed databases

- More data can be managed and it can be JN9 placed in computers in different places.
- This provides performance, but with the cost of JN10 increased coordination between the database instances on different computers.
- In particular, the updates need to be coordinated to ensure that they are managed consistently across the distributed database.
- There will be a lot about distributed updates in the next lecture.
- But not so much of distributed commit protocols.

UNIVERSITY OF TAMPERE

Distributed Transactions

- In a distributed transaction there is a set of subtransactions T1,...,Tk, which are executed on sites S1,...,Sk.
- Each subtransaction manages local data. The particular problems of managing distributed transactions vs. centralised (local) transactions come from two sources:
  - Data may be replicated to several sites. Lock management of the replicated data is a particular problem (can be done by voting).
  - Regardless of whether the data is replicated or not, there is a need to control the fate of the distributed transaction using a distributed commit protocol.

UNIVERSITY OF TAMPERE

JN9 There are no advantages in terms of quantity of data. If an organization has a distributed database, it will be because it was forced to acceppt the solution. E.g., two companies with different database technologies merged.
Jyrki Nummenmaa; 17.5.2017

JN10 I doubt whether there are any performance benefits. An organization tolerates a distriuted database because integrating the databases would be a nightmare.
Jyrki Nummenmaa; 17.5.2017

Distributed Transaction

- A set of participating processes with local sub-transactions, distributed to a set of sites, perform a set of actions.
- *All or none* of the updates or related operations should be performed.
- Process autonomy - any process can unilaterally decide to abort the transaction.

UNIVERSITY OF TAMPERE

Distributed Commit

- At the end of the transaction, it must be found out, whether it is feasible to make the proposed changes on all participating processes.
- This is done by a voting protocol called distributed commit protocol.
- Without failures, voting would be extremely simple.

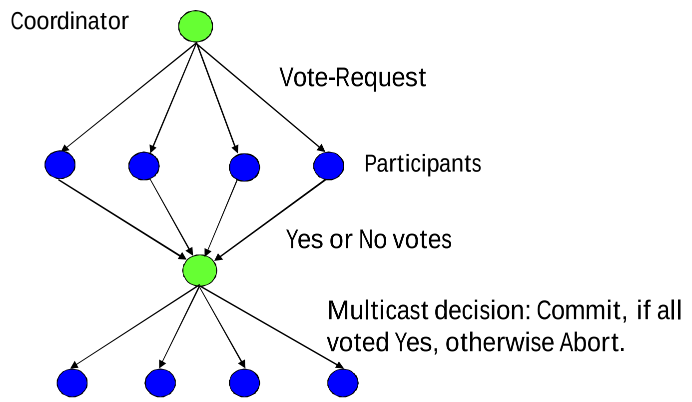
UNIVERSITY OF TAMPERE

Failure

- Hardware failure
- Software crash
- User switched off a computer
- Active attack
- Network/message delivery failure
- Denial-of-service attack
- Typically, these failures are partial.

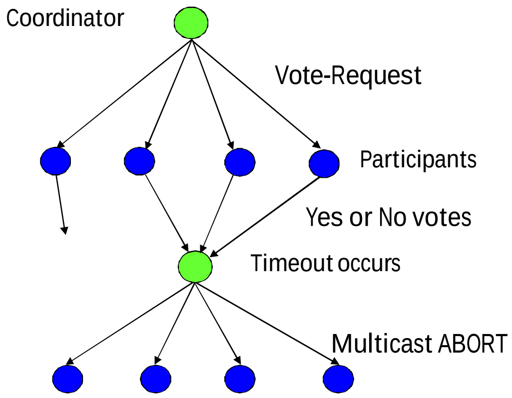
UNIVERSITY OF TAMPERE

2PC for Distributed Commit



UNIVERSITY OF TAMPERE

2PC - a timeout occurs



UNIVERSITY OF TAMPERE

ACID properties in Big Data databases

- Typically, transactions are simple. So:
  - Atomicity is not a concern.
  - Concistency checking is complicated and ”dirty reads” may occur.
  - Simple transactions help in isolation maintenance.
  - Durability is required.
    - But not as critical as in e.g. banking.

UNIVERSITY OF TAMPERE

Physical arrangements

- BigTable is Google's Big Data database model.
- Rows maintained in sorted by primary key order
  - Applications can use this property for efficient row scans
- Columns grouped into column families
  - Column key = *family:qualifier*
  - Column families provide locality hints
  - Unbounded number of columns

UNIVERSITY OF TAMPERE

BigTable Applications and HBASE

- BigTable can be used as input and output for MapReduce
- Applications: Google's web crawl, Google Earth, Google Analytics
- HBase is the open source Hadoop implementation of Bigtable.
  - Runs on HDFS (which provides replication)
  - Scales up to 1000s of servers and PBs of data.

UNIVERSITY OF TAMPERE

HBase is ..

- A distributed data store that can scale horizontally to 1,000s of commodity servers and petabytes of indexed storage.
- Offers persistency
- Can deal with distributed, sparse data
- Designed to operate on top of the Hadoop distributed file system (HDFS) or Kosmos File System (KFS, aka Cloudstore) for scalability, fault tolerance, and high availability.

UNIVERSITY OF TAMPERE

Hbase uses multiversioning

- Multiversioning is the alternative for locking in concurrency control
  - Transactions can write new data
  - Transaction can read the "right" data based on its own timestamp and timestamps of the data versions.
- A value is identified by tuple (Table,RowKey,ColFamily,Column,Timestamp)
- By default 3 versions are kept, but this can be configured.

UNIVERSITY OF TAMPERE

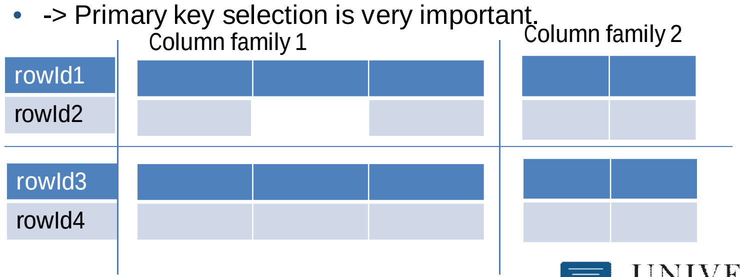
Hbase uses multiversioning

- Multiversioning is the alternative for locking in concurrency control
  - Transactions can write new data
  - Transaction can read the “right” data based on its own timestamp and data versions’ timestamps.
  - No real transaction model in Hbase (just get&put)
- A value is identified by tuple (Table,RowKey,ColFamily,Column,Timestamp)
- By default 3 versions are kept, configurable.

UNIVERSITY OF TAMPERE

Hbase data organization

- Data is organized into stores by column families
- Value only exists, if stored (implicit NULLs)
- Data is physically ordered by row key.
- Can also be partitioned by row key.
- Data is stored as bytes (Hbase point of view)
- No secondary indexing built-in.
  - -> Primary key selection is very important.



UNIVERSITY OF TAMPERE

Queries vs. scanning of data

- No query language included.
- Data read from Hbase can be fed to MapReduce
  - Good integration for this.
  - MapReduce output can also go to HBase.
- Fast scanning of data between row key interval.
  - After locating the right primary key, it is possible to read the column family data fast, using this physical order.
- By default read gives the last data by timestamp.

UNIVERSITY OF TAMPERE

When HBase

- Lot of writes but no updates to previously written data.
- It is feasible to use the primary key for direct access and MapReduce for other searching.
  - No need for more sophisticated query language.
- Used by Facebook, twitter, Yahoo! etc.

UNIVERSITY OF TAMPERE

Let’s go back to last week’s cartoon!

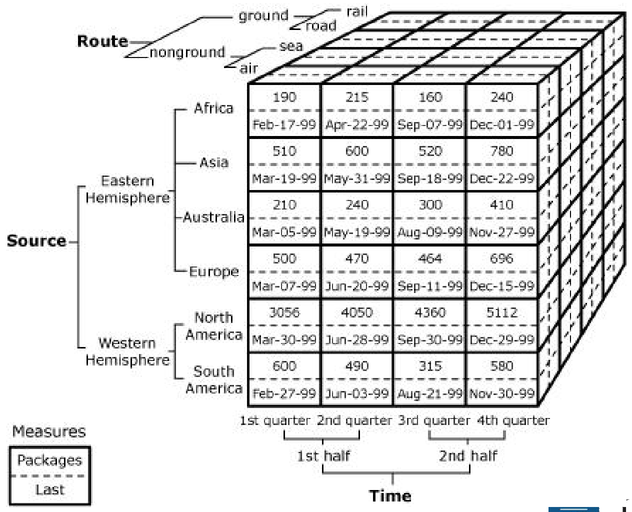
UNIVERSITY OF TAMPERE

How could end-users explore the database without their own distributed MapReduce written in Erlang?

UNIVERSITY OF TAMPERE

What is dimensional analysis and OLAP?

UNIVERSITY OF TAMPERE



UNIVERSITY OF TAMPERE

Dimensional Approach

- Separation between quantitative and qualitative attributes
  - Measure
    - Is a measure of facts (events)
    - Is aggregatable
    - Is a quantitative attribute
  - Dimension
    - Is a logical entity (Customer, Product, Time, …)
    - Each dimension has qualitative attributes describing the entity
- Representation as a multidimensional matrix

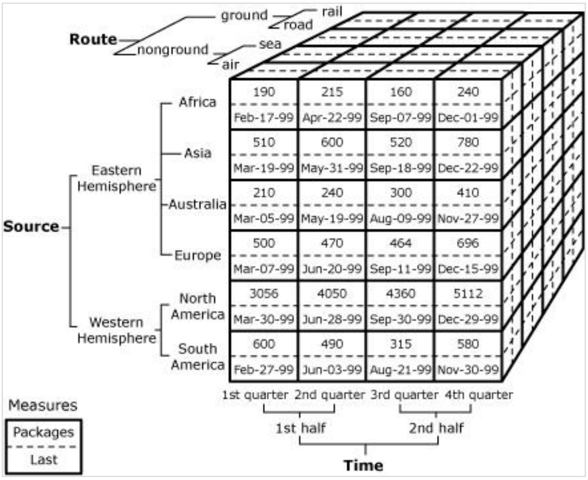
UNIVERSITY OF TAMPERE

Formalizing the dimensional model

1. A dimension schema D_i ($1 \le i \le n$) is a sequence $< A^i_1, A^i_2, \ldots, A^i_x >$ ($x = |D_i|$) of attributes, called levels. (one attribute per level, see 4)
2. $D =< D_1, D_2, \ldots, D_n >$ is an ordered set of dimension schemata.
3. $M = \{M_1, M_2, \ldots, M_q\}$ is a set of measure attributes.
4. The members of the collection $\{D_1, D_2, \ldots, D_n, M\}$ are pairwise disjoint (Because of this, one attribute per level does not limit generality)

UNIVERSITY OF TAMPERE

Dimensions and measures?



UNIVERSITY OF TAMPERE

Example

- Example 1. Let (D,M) be a summarization schema in which D =< D_1,D_2> and M = {M_1}. We let D_1= {A^1_1,A^1_2} and D_2= {A^2_1,A^2_2,A^2_3}. Let r = {t_1, t_2, . . . , t_5} be the following (flat) relation over the summarization schema (D, M).

	• A1.1	• A1.2	• A2.1	• A2.2	• A2.3	• M
• t1	b1	c1	d1	e1	f1	10001
• t2	b1	c2	d1	e1	f1	10020
• t3	b2	c3	d1	e1	f2	10300
• t4	b2	c3	d2	e1	f3	14000

UNIVERSITY OF TAMPERE

Formalizing the dimensional model

Definition 2 (Set of domain values). Let r be a relation over a summarization schema, (D, M). For $1 \le i \le |D_i|$, $1 \le j \le n$, $DOM_j^i(r)$ denotes the set of domain values in the A^i_j column of r.

Example 2. Thus, in our running example, we have the following values in Dimension 1

Dimension D_1
$DOM_1^1(r) = \{b1, b2\}$ $DOM_2^1(r) = \{c1,c2, c3\}$

UNIVERSITY OF TAMPERE

Set of domain values

Definition 2 (Set of domain values). Let r be a relation over a summariza- tion schema, (D, M). For $1 \le i \le |D_i|$, $1 \le j \le n$, $DOM_j^i(r)$ denotes the set of domain values in the A^i_j column of r.

Example 2. Thus, in our running example, we have the following values:

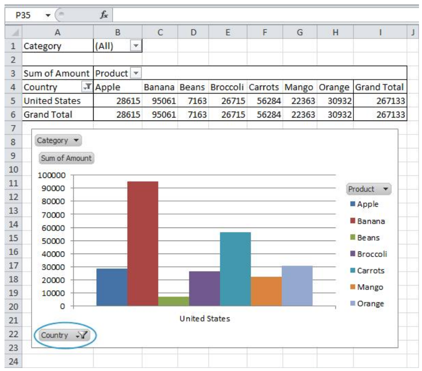
Dimension D_1
$DOM_1^1(r) = \{b1, b2\}$, $DOM_2^1(r) = \{c1, c2, c3\}$

Dimension D_2
$DOM_3^2(r) = \{f1, f2, f3\}$ $DOM_1^2(r) = \{d1, d2\}$ $DOM_2(r) = \{e1, e2\}$

UNIVERSITY OF TAMPERE

How do end-users query ?

- There is a large amount of users who can use such tools as Excel pivot tables, which can utilize (and create small) dimensional structures



UNIVERSITY OF TAMPERE

Simple summarization

SELECT DaysToManufacture,
AVG(StandardCost) AS AverageCost
FROM Production.Product
GROUP BY DaysToManufacture;

DaysToManufacture	**AverageCost**
0	5.0885
1	223.88
2	359.1082
4	949.4105

UNIVERSITY OF TAMPERE

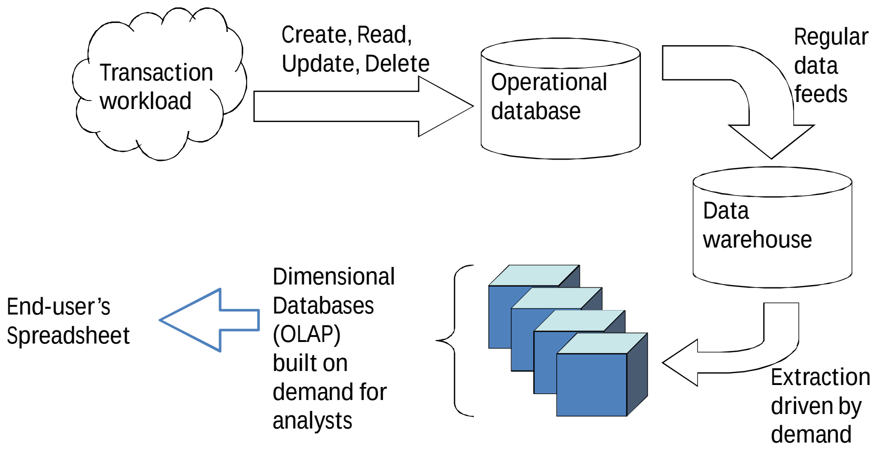
Example pivoted in SQL

```sql
SELECT 'AverageCost' AS Cost_Sorted_By_Production_Days,
[0], [1], [2], [3], [4]
FROM
(SELECT DaysToManufacture, StandardCost
    FROM Production.Product) AS SourceTable
PIVOT
(
AVG(StandardCost)
FOR DaysToManufacture IN ([0], [1], [2], [3], [4])
) AS PivotTable;
```

Cost_Sorted_By_Production_Days	0	1	2	3	4
AverageCost	5.0885	223.88	359.1082	NULL	949.4105

UNIVERSITY OF TAMPERE

Traditional data flow for OLAP cubes (dimensional structures)



UNIVERSITY OF TAMPERE

Big Data era

- End users like to use the same tools
- Results are wanted fast
- New data is coming in fast
- There are lots of attributes, dimensional structures would be huge, and may contain vast amounts of values that are never looked at
- It is becoming increasingly non-practical to pre-compute the dimensional structure (such as OLAP cube)

UNIVERSITY OF TAMPERE

“Curse of dimensionality”

- “[...] no feasible data cube can be constructed with [...] over 100 dimensions and [...] 10^6 tuples”

Li, X., Han, J. and Gonzalez, H., High-dimensional OLAP: a minimal cubing approach. *Proceedings of the 30th VLDB Conference*, Toronto, Canada, 2004, pp. 528-539.

- This was described as the “curse of dimensionality”

Gibas, M., Canahaute, G., AND Ferhatosmanoglu, H. 2008. Online index recommendations for high dimensional databases using query workloads. *IEEE Trans. on Knowl. and Data Eng.* 20, *2*, 246–260.

- Lazy evaluation avoids such a curse.

UNIVERSITY OF TAMPERE

Cube vs. No cube

- CUBE:
- Acceptable performance with small number of dimensions
- Good for complex "navigational" style dimensional queries.
- Pre-aggregation may be needed for performance.
- Examples: Cognos Powerplay, Oracle OLAP Option, Microsoft Analysis Services, Essbase
- NO CUBE:
- Can cope with a large number of dimensions.
- Only suited to relatively simple aggregation expressions.
- Advantageous for unpredictable ad hoc queries.
- Examples: SAP Business Objects, Oracle BI Suite, Microsoft Analysis Services.

UNIVERSITY OF TAMPERE

Statistical approach

- General method:
  1. Find distribution functions to fit the measure attributes
  2. Capture the intra relationship among attributes.
  3. Use probability density functions of joint distributions to find approximate answers to aggregation functions.
- Problem: suppose you find some interesting-looking results:
  1. How do you estimate the error margin?
  2. How do you (quickly) find the exact answer for the items of information in which you are particularly interested?
  - *Solution: use our technique!*

UNIVERSITY OF TAMPERE

User Input

1. The subset of the dimensions over which aggregation is required.
2. For each such dimension, the name of the attribute which corresponds to the required level in the aggregation hierarchy.
3. For each such attribute, a domain member to be used for selection (later called "slicer").
4. An expression in which the operands are aggregation functions (e.g. MIN, MAX, AVG) with measure attributes as arguments.

- Inputs for our technique can be obtained "semi-automatically" using Pivot Tables.

UNIVERSITY OF TAMPERE

Rollup vector

Let $D =< D_1, D_2, . . . , D_n >$ be dimension schemata.
A *rollup vector* for D is an ordered n-tuple of integers, $L =< L_1,L_2,...,L_n >$, where for $i = 1,2,...,n, 0 \leq L_i \leq |D_i|$.

The interpretation of the values of the elements of L is as follows.
– $L_i = 0$ denotes that the user's analysis does not require an axis corresponding to dimension schema D_i.
– $1 \leq Li \leq |D_i|$ denotes that the user's analysis requires an axis corresponding to dimension schema D_i.

If $L_i = l$, and r is the relation of interest, then the axis that corresponds to dimension schema D_i shall be labelled by the elements of $DOM_l^i(r)$, (members of the attribute column A^i_l)

UNIVERSITY OF TAMPERE

Selection on cube values

Slicer vectors are used to select on different levels of dimension hierarchies.

Permissible values in the slicer vector are constrained by the values in the rollup vector. The slicer for a given dimension must be a single member of the level for that dimension which is specified in the rollup vector.

UNIVERSITY OF TAMPERE

Selection on cube values

Given a relation, r, a dimension, D_i, $1 \leq i \leq n$, and a level number, l, $1 \leq l \leq |D_i|$, let A^i_l denote the attribute associated with level l of dimension D_i and let $x \in DOM^i_l(r)$.

The set of tuples in a relation r, which have the value x in the attribute column denoted by i and l is:

tuples(i,l,x) =

r, if $l = 0$ $t \in r | t[A^i_l] = x$, if $l \neq 0$

UNIVERSITY OF TAMPERE

Selection on cube values

	• A1.1	• A1.2	• A2.1	• A2.2	• A2.3	• M
• t1	b1	c1	d1	e1	f1	10001
• t2	b1	c2	d1	e1	f1	10020
• t3	b2	c3	d1	e1	f2	10300
• t4	b2	c3	d2	e1	f3	14000

The tuples() sets for our running example is as follows: tuples(1,1,b1) = {t1,t2} tuples(1,1,c3) = {t3,t4} tuples(2,2,e2) = {t4} tuples(1,1,b2) = {t3,t4} tuples(2,1,d1) = {t1,t2,t3} tuples(2,2,e1) = {t1,t2,t3} tuples(1,1,c1) = {t1} tuples(2,1,d2) = {t4} tuples(2,3,f2) = {t3} tuples(1,1,c2) = {t2} tuples(2,3,f1) = {t1,t2} tuples(2,3,f3) = {t4}

UNIVERSITY OF TAMPERE

Slicer vector

Given the summary state s = (r, L), L =< L_1, L_2,...,L_n >, a slicer vector for s is an ordered set < x_1,x_2,...,x_n > where for i = 1,2,...,n, if in the rollup vector, $L_i = 0$, then x_i = NULL; otherwise let L_i =l>0, then $x_i \in DOM_l^i(r)$.

Example 6. Given the rollup vector, < 1, 2 >, a valid slicer vector is < b1, e1 >.

UNIVERSITY OF TAMPERE

Slicer vector

Given a summary state, s = (r, L), and a slicer vector, we can find the collection of tuples() sets which contain the measure values needed by the summary function in order to compute the value for the cell identified by the slicer vector.

Example 7. In our running example, the rollup vector, < 1, 2 >, and the slicer vector, < b1, e1 >, give tuples(1, 1, b1) = $\{t_1, t_2\}$ and tuples(2, 2, e1) = $\{t_1, t_2, t_3\}$.

UNIVERSITY
OF TAMPERE

Summary Rowset

The rollup and slicer vectors identify the cell in the summarization structure for which a summarization operation is to be performed. In our implementation, the actual data for this summarization operation is in the original relation.

Let s = (r,L) be a summary state and let

x =< $x_1,x_2,...,x_n$ > be a slicer vector for s. The set

tuples(i, L_i, x_i) is the *summary rowset* for s and x .

UNIVERSITY
OF TAMPERE

Theorem 1

Let $s = (r,D,< L_1,L_2,...,L_n >,f,M_1)$ be a summary state, and
$x =< x_1,x_2,...,x_n >$ a slicer vector for s.

Then a row $t \in r$ must be included in an aggregation operation for the cell identified by x if and only if

$t \in \cap$ tuples(i, L_i, x_i).

UNIVERSITY OF TAMPERE

Example using Theorem 1

Using the data in our running example, with the summary state and the slicer vector from Example 6, suppose we want to apply the SUM aggregation operator.

Using Theorem 1, the summary rowset is T = tuples(1, 1, b1) ∩ tuples(2,2,e1) = $\{t_1,t_2\}$. The sum for this cell is given by the expression t_i[M].

From the data, t_1[M] = 10001 and t_2[M] = 10020. The sum is 20021.

UNIVERSITY OF TAMPERE

Using Theorem 1

Thus, given a denormalised [10] table (which contains both the fact data and the dimension data), a rollup vector and a slicer vector, we can use Theorem 1 to compute a summarised value anywhere in the summarization structure directly, without having to create the entire cube.

UNIVERSITY OF TAMPERE

Implementation

- The user:
  - Loads the data model into Excel (with a sample of the data)
  - Creates a pivot table with the slicers and filters
  - Clicks on the required pivot table cell
- Our software
  - Captures the cube formula for the cell,
  - Generates the SQL query which efficiently computes the aggregation for the cell using the whole data
  - Executes the query and returns the result

UNIVERSITY OF TAMPERE

SQL code to generate SQL

```
ALTER PROCEDURE [dbo].[sp_Build_and_Exec_Query]
   @numberOfDimensions INT,     @measureAttributeName NVARCHAR(100),
@aggregationFunctionName
NVARCHAR(100)
AS  BEGIN
   DECLARE @sql NVARCHAR(MAX),  @attributeName NVARCHAR(MAX),       @dim INT,
@val NVARCHAR(8);
    SET @sql = N'SELECT ' + @aggregationFunctionName   + N' ( '  +
@measureAttributeName
                                                                    + N' ) FROM tblFlat
WHERE ';
   SET @dim = 1;
   WHILE @dim <= @numberOfDimensions     BEGIN
      SET @attributeName = getAttributeName(@dim);
      SELECT @val = [val] FROM dbo.MemberVector WHERE dimNumber = @dim;
      IF @dim > 1             SET @sql = @sql + N' AND ';
      SET @sql = @sql +  + @attributeName + N' = ''' + @val + '''';
      SET @dim = @dim + 1;
   END
   EXECUTE sp_executesql @sql;
END
```

UNIVERSITY OF TAMPERE

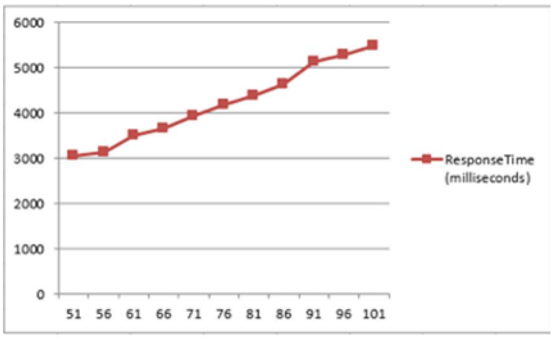
Example of generated SQL

```
SELECT SUM(M1)
FROM T
WHERE
D1L2 = 'D01L2M01' AND D2L1 = 'D02L1M1'
                        AND D3L2 = 'D03L2M01'
                         . . .
                        AND D100L1 = 'D100L1M1'
                        AND D101L2 = 'D101L2M0';
```

UNIVERSITY OF TAMPERE

Performance

Time to aggregate a single sell plotted against the number of dimensions



UNIVERSITY OF TAMPERE

Performance vs. database size

Rows & Response time

500,000 1674

600,000 1841

700,000 2032

800,000 1994

900,000 2137

1,000,000 2086

UNIVERSITY OF TAMPERE

Disjointness of summary sets

Lemma 1. Disjointness of summary rowsets. Let s = (r,L) be a summary state, L =< L_1,L_2,...,L_n >, let x and x′ be different slicer vectors for s and let T and T′ be the corresponding summary rowsets for x and x′. Now, $T \cap T' = \emptyset$.

A particular consequence of Lemma 1 is that the calculation of a summary value for different cells does not involve redundant computation.

UNIVERSITY OF TAMPERE

Null and non-null cells

Given a summarization instance, in general the vast majority of summary cells will be NULL. A summary cell will be NULL if the tuple set associated with cell's slicer vector is empty.

The number of non-NULL cells in the summarization instance cannot exceed the number of rows in the de-normalised relation (and it could be a lot less than this number of tuples).

By contrast, the number of summary cells depends on the number of axes and the number of values to index each axis, and the number of members. For example, if there are one hundred axes each with 10 values in the summarization instance then there will be 10^{100} summary cells.

UNIVERSITY OF TAMPERE

Unique slicer vectors for tuples

Lemma. Let s = (r,< L_1,L_2,...,L_n >) be a summarization instance. For each row, t ∈ r, t is in the tuple set of exactly one slicer vector, and that slicer vector can be formed from the values in t.

For any tuple in the relation, there is a unique slicer vector and for any slicer vec- tor there is a unique summary cell. Consequently, a summarization instance defines a set-theoretic partition on the set of tuples in the denormalised relation. Given a sum- marization instance, Algorithm 1 computes exactly the non-empty elements of that partition.

UNIVERSITY OF TAMPERE

…continued

At the end of this process, we have the partition of the set of tuples. With each element of the partition, we have the associated slicer vector. Thus we have identified every non-NULL summary cell associated with the summarization instance.

Now, when the user wants to find the contents of a summary cell, we determine the slicer vector, say x for that summary cell, then we find the matching partition whose members are precisely the tuples, t, for which slicerV ector(s, t) = x.

If no partition is found, then this summary cell is NULL.

UNIVERSITY OF TAMPERE

Complexity of computing the tuple sets

Theorem. The tuple sets associated with all non-NULL summary cells in a summarization instance can be computed in time O(nw), where n is the number of dimensions and w is the number of tuples in the denormalised relation.

The proof is based on an algorithm iterating through the tuples of the denormalised relation and for each tuple, constructing an n-place vector. For each such vector, it must check by using a hash function whether or not it has already generated an identical vector for an earlier tuple. There can be no more than w such vectors.

UNIVERSITY OF TAMPERE

Number of possible rollup vectors

Lemma. Given a summarization schema (< D_1 , ..., D_n >, M), the number of possible rollup vectors is exponential to n.

Proof. Consider a case, where each D_i =< A^i_1 >. Then in the rollup vector the corresponding L_i may be either 0 or 1, thus giving n^2 different rollup vectors. In case that the dimensions contain more attributes, there will be even more different rollup vectors.

UNIVERSITY OF TAMPERE

Complexity of computing the tuple sets

Theorem. The tuple sets associated with all non-NULL summary cells in a summarization instance can be computed in time O(nw), where n is the number of dimensions and w is the number of tuples in the denormalised relation.

The proof is based on an algorithm iterating through the tuples of the denormalised relation and for each tuple, constructing an n-place vector. For each such vector, it must check by using a hash function whether or not it has already generated an identical vector for an earlier tuple. There can be no more than w such vectors.

UNIVERSITY OF TAMPERE

Conclusions

- Virtually unlimited amount of dimensions can be managed
- The present technique collects the tuples from SQL database or databases
- The technique parallelizes in a straightforward way, but that is yet to be done to test the technique using Spark or Hadoop MapReduce

UNIVERSITY OF TAMPERE

Finally it is over…

- Questions?

UNIVERSITY OF TAMPERE