About the speaker

Luigi Troiano is assistant professor and researcher at University of Sannio since 2006, where he is lecturing in Software Engineering and Intelligent Systems. He obtained Laurea (master degree) in IT Engineering at University of Naples “Federico II” in 2000, and Ph.D. in IT Engineering at University of Sannio in 2004. His research interests are mainly related to Computational Intelligence and Intelligent Systems, focusing on how to apply mathematical models and advanced algorithms to real-world problems. His competences refer to conception, data analysis, algorithm experimentation and validation, implementation of software systems and solutions, using the state of art of technologies. He earned an industrial experience working in Italy and abroad for multinational companies such as Pirelli Trelleborg (1999), Siemens ICN (2000-2001) and TotalFinaElf E&P Paris (2001-2002).

Big Data Science.
Any use for KAGRA?

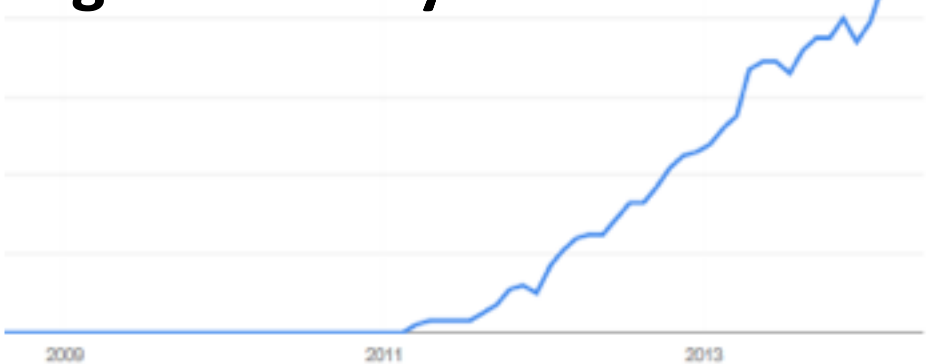
Luigi Troiano
University of Sannio, Italy

ICRR, University of Tokyo — May 23, 2014

Big Data by Google Trends

Big Data

Big Data Analytics



Big Data

“**Big Data** is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.”

Gartner Research

Big Data in Science

Some examples:

- The 1000 Genomes Project is aimed to find most genetic variants that have frequencies of at least 1% in the populations studied. The genome of each human being is 100 GB long.

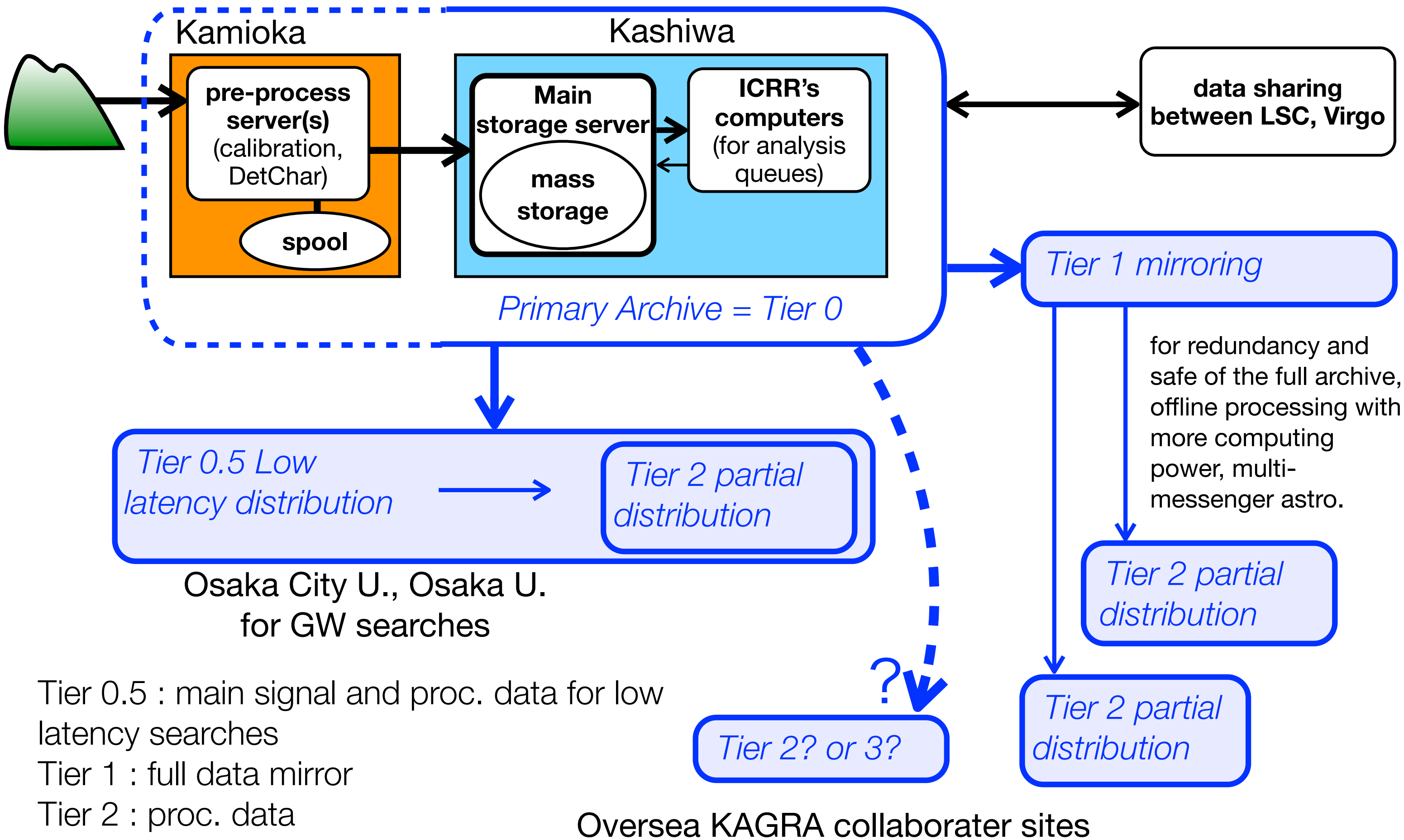
- Jack Gallant at UC Berkeley was able to recover what people were seeing by directly observing activity in their brains by using big data and statistical methods.

- The Large Hadron Collider (LHC) at CERN in Switzerland started to take data in 2009. The amount of data collected by CERN is about 25 PB a year.

And Kagra?

Tier	Site(s)	Purpose	Raw	Calibrated	Detecter Characteriza tion	Amount of data for 5yrs	event alarts
	Kamioka	DAQ	partial (spool)	partial (spool)	partial (spool)	200TB	partial
0	Kashiwa	Main Storage	○	○	○	5PB	(Not yet discussed)
0.5	Osaka City, Osaka	Low latency	NA or small amount	○	○	500TB	○
1	(undecided)	Full Mirroring	○	○	○	5PB	(Not yet discussed)
2	(Analysis-bases, undecided)	Offline searches	NA	○	○	500TB	NA
3	End users	Developm ent	NA	partial	partial	(Not yet discussed)	NA

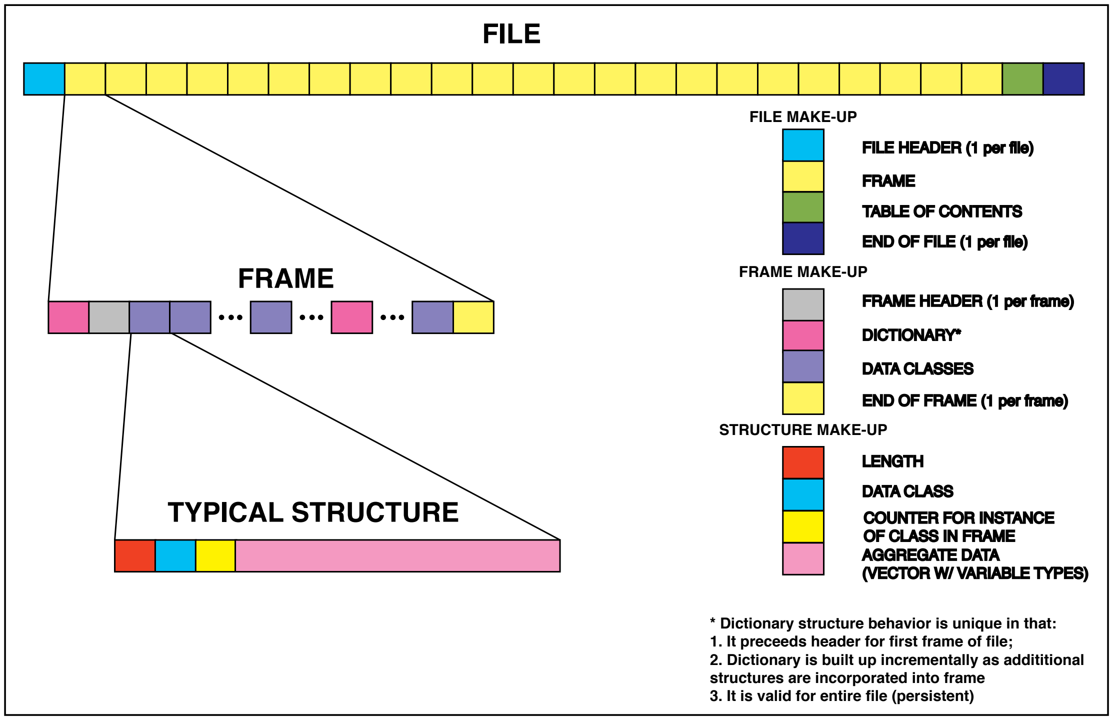
Kagra Processing Tiers

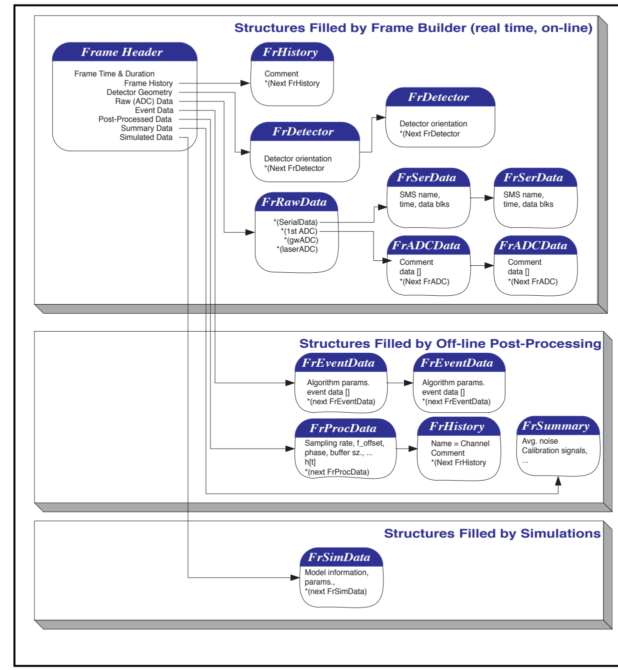


Tier 0.5 : main signal and proc. data for low latency searches
Tier 1 : full data mirror
Tier 2 : proc. data
Tier 3 : partial cache of proc. data

Frame Data Structures

Figure 1: Schematic representation of data organization within a file.



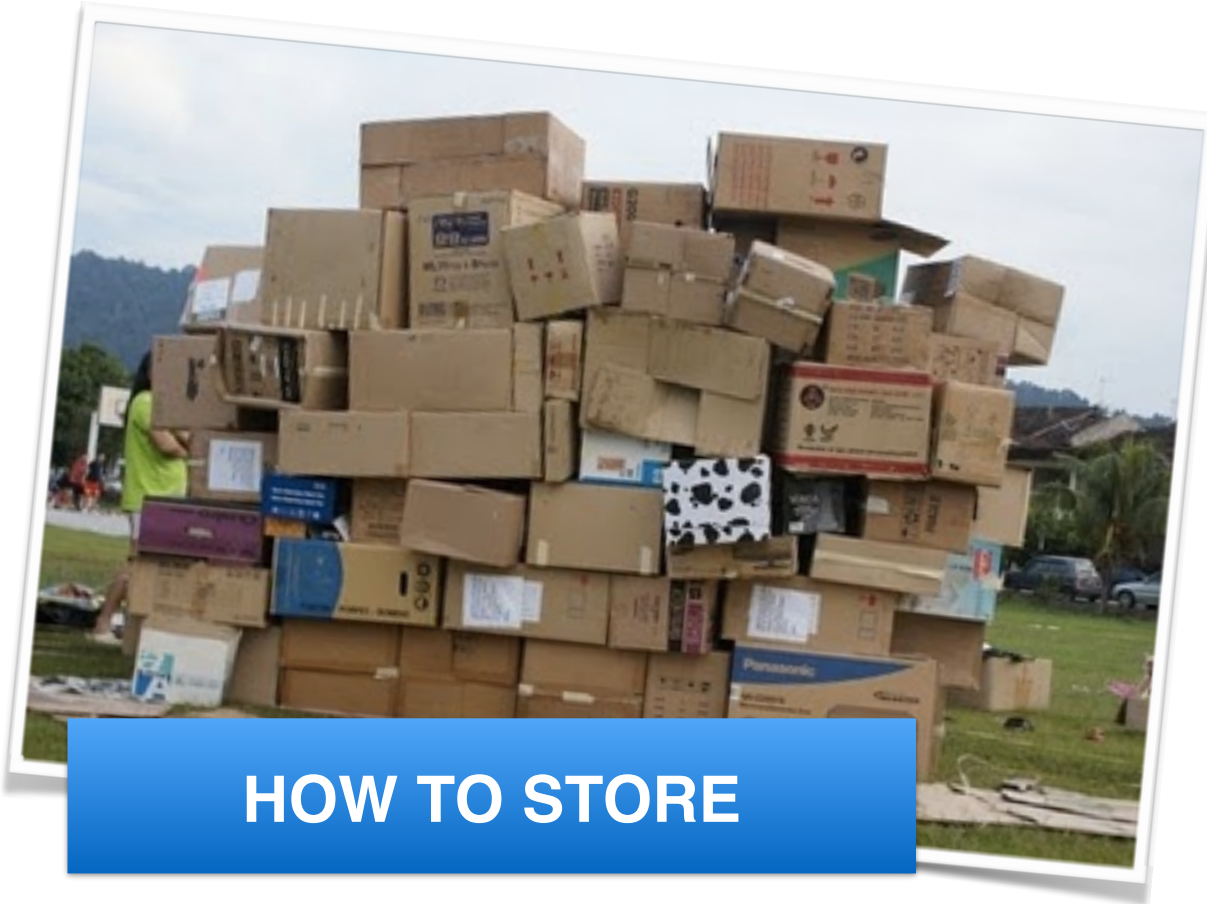


Big Data Challenges

HOW TO STORE

HOW TO RETRIEVE

HOW TO ANALYZE

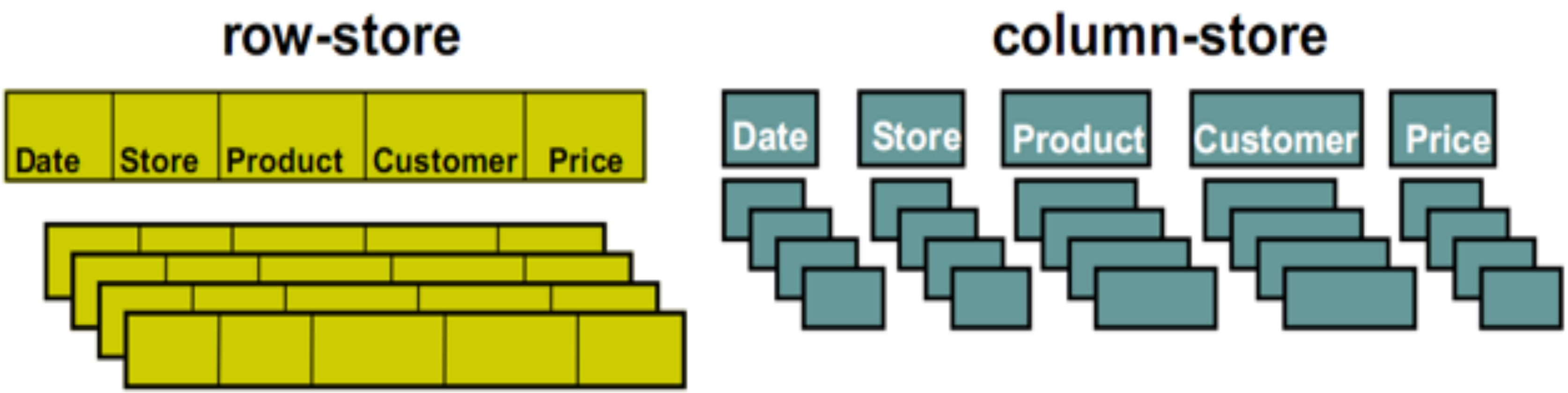


HOW TO STORE

Column-oriented DB

Column-oriented (a.k.a. vertical) databases store data with a focus on columns, instead of rows, allowing for huge data compression and very fast query times.

The downside to these databases is that they will generally only allow batch updates, having a much slower update time than traditional models.



Columns vs. Rows

- Column-oriented databases are suitable for read-mostly, read-intensive, large data repositories
  - OLAP, On-Line Analytical Processing
  - Big Data Analytics
- Row-oriented (conventional) databases are more suitable for accessing/update single transactions
  - OLTP, On-Line Transaction Processing
  - CRUD, Create/Read/Delete/Update activities

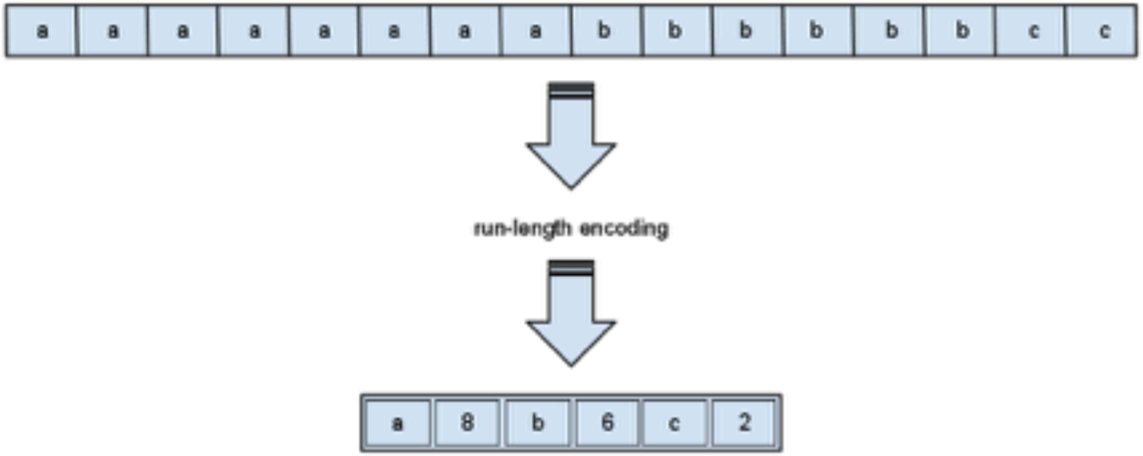
Row Store	Column Store
(+) Easy to add/modify a record	(+) Only need to read in relevant data
(-) Might read in unnecessary data	(-) Tuple writes require multiple accesses

Optimizing C-DBs

- Column-oriented databases make large use of the following optimizations:
  - Compression
  - Late Materialization
  - Block Iteration
  - Invisible Join

Compression

- Low information entropy (high data value locality) leads to High compression ratio
- If data is sorted on one column that column will be super-compressible in row store
- eg. Run Length Encoding



Late Materialization

- As result of queries we expect records
- So at some point of time multiple column must be combined
- One simple approach is to join the columns relevant for a particular query
- But further performance can be improve using late-materialization

Late Materialization

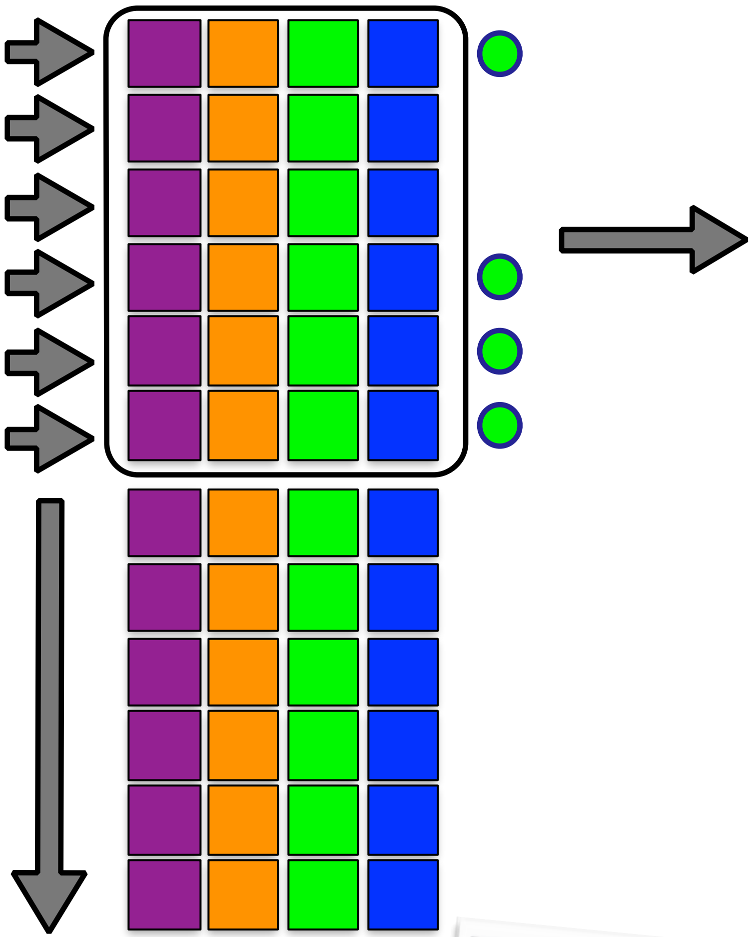
- Delay Tuple Construction
- Might avoid constructing it altogether
- Intermediate position lists might need to be constructed
- Eg: SELECT R.a FROM R WHERE R.c = 5 AND R.b = 10
  - Output of each predicate is a bit string
  - Perform Bitwise AND
  - Use final position list to extract R.a

Late Materialization

- Advantages
  - Unnecessary construction of tuple is avoided
  - Direct operation on compressed data
  - Cache performance is improved

Block Iteration

- Operators operate on blocks of tuples at once
- Iterate over blocks rather than tuples
- Like batch processing
- If column is fixed width, it can be operated as an array
- Minimizes per-tuple overhead
- Exploits potential for parallelism



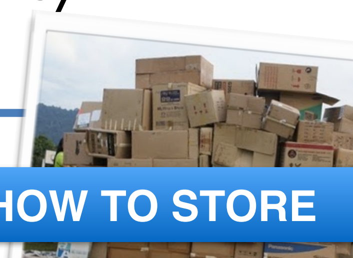
Invisible Join

- Invisible join is a late materialized join but minimize the values that need to be extracted out of order
- Invisible join
  - Rewrite joins into predicates on the foreign key columns in the fact table
  - These predicates evaluated either by hash-lookup
  - Or by between-predicate rewriting

Invisible Join

```
SELECT c.nation, s.nation, d.year,
       sum(lo.revenue) as revenue
FROM customer AS c, lineorder AS lo,
     supplier AS s, dwdate AS d
WHERE lo.custkey = c.custkey
  AND lo.suppkey = s.suppkey
  AND lo.orderdate = d.datekey
  AND c.region =  ASIA
  AND s.region =  ASIA
  AND d.year >= 1992 and d.year <= 1997
GROUP BY c.nation, s.nation, d.year
ORDER BY d.year asc, revenue desc;
```

Find Total revenue from Asian customers who purchase a product supplied by an Asian supplier between 1992 and 1997 grouped by nation of the customer, supplier and year of transaction



Invisible Join

STEP 1

Apply `region = 'Asia'` on Customer table

custkey	region	nation	...
1	Asia	China	...
2	Europe	France	...
3	Asia	India	...

→ Hash table with keys 1 and 3

Apply `region = 'Asia'` on Supplier table

suppkey	region	nation	...
1	Asia	Russia	...
2	Europe	Spain	...

→ Hash table with key 1

Apply `year in [1992,1997]` on Date table

dateid	year	...
01011997	1997	...
01021997	1997	...
01031997	1997	...

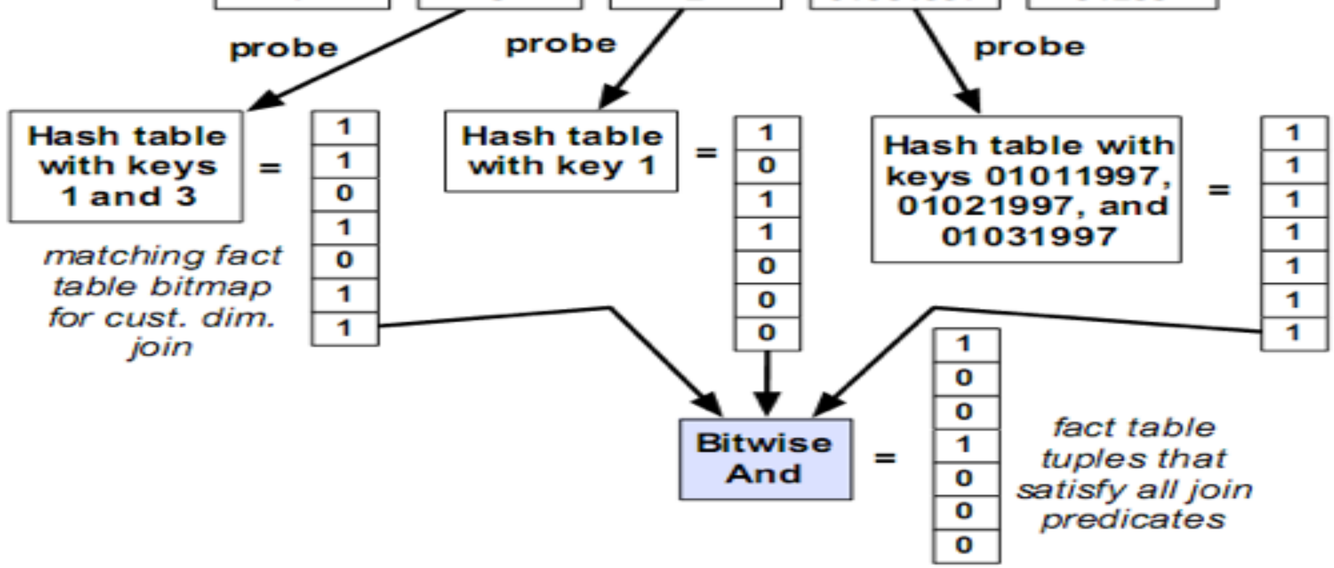
→ Hash table with keys 01011997, 01021997, and 01031997

Invisible Join

STEP 2

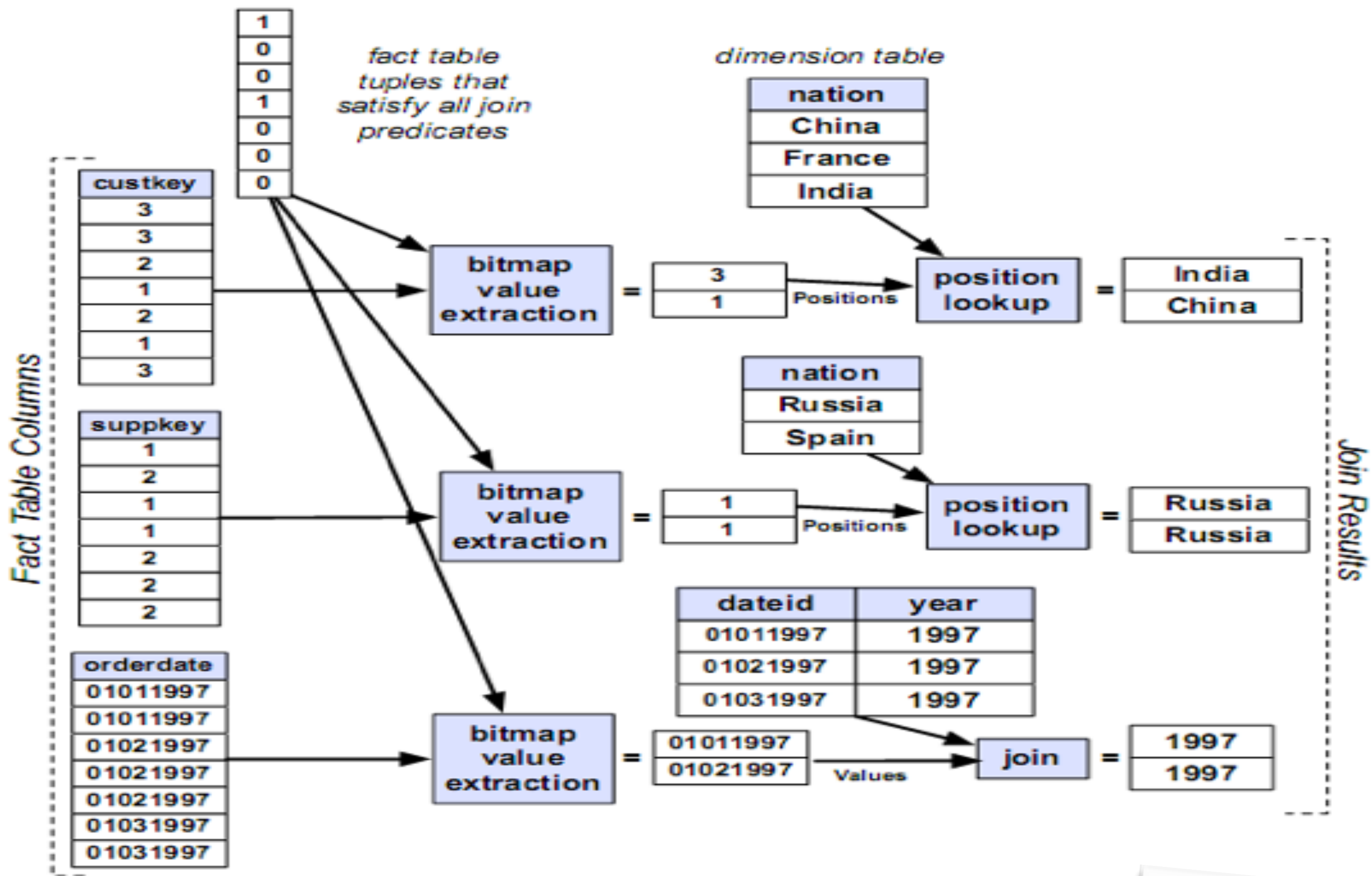
Fact Table

orderkey	custkey	suppkey	orderdate	revenue
1	3	1	01011997	43256
2	3	2	01011997	33333
3	2	1	01021997	12121
4	1	1	01021997	23233
5	2	2	01021997	45456
6	1	2	01031997	43251
7	3	2	01031997	34235



Invisible Join

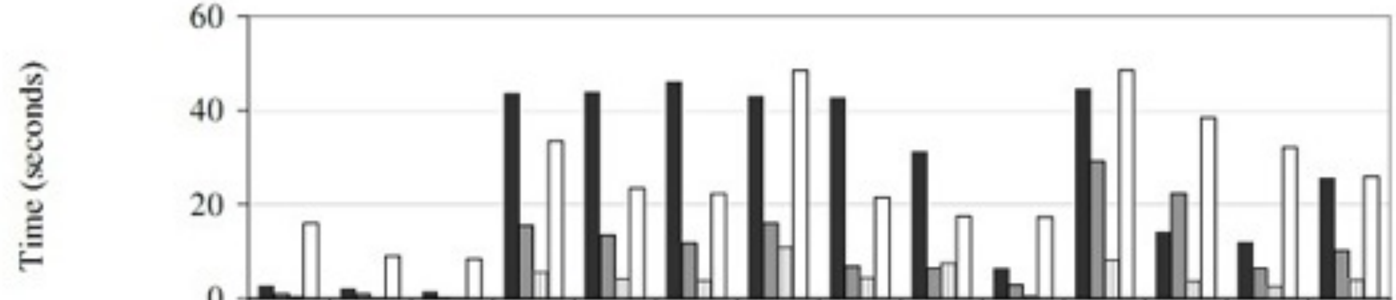
STEP 3



C-DBMS (some)

- Apache Accumulo (Open Source)
- **Apache Cassandra** (Open Source)
- **Apache HBase** (Open Source)
- Calpont InfiniDB (Commercial)
- **C-Store** (Discontinued)
- Druid (Open Source)
- **MonetDB** (Open Source)
- RCFile (Open Source)
- Sybase IQ (Commercial)
- **Vertica** (Commercial branch of C-Store)

Comparison



	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2	3.3	3.4	4.1	4.2	4.3	AVG
■ RS	2.7	2.0	1.5	43.8	44.1	46.0	43.0	42.8	31.2	6.5	44.4	14.1	12.2	25.7
RS (MV)	1.0	1.0	0.2	15.5	13.5	11.8	16.1	6.9	6.4	3.0	29.2	22.4	6.4	10.2
CS	0.4	0.1	0.1	5.7	4.2	3.9	11.0	4.4	7.6	0.6	8.2	3.7	2.6	4.0
□ CS (Row-MV)	16.0	9.1	8.4	33.5	23.5	22.3	48.5	21.5	17.6	17.4	48.6	38.4	32.1	25.9

Baseline performance of C-Store "CS" and System X "RS", compared with materialized view cases on the same systems.

- RS: Conventional Data Base System (Not Mentioned)
- CS: Base C-Store case
- RS (MV): System X with optimal collection of MVs
- CS (Row-MV): Column store constructed from RS(MV)

HOW TO RETRIEVE

Problem

How to retrieve

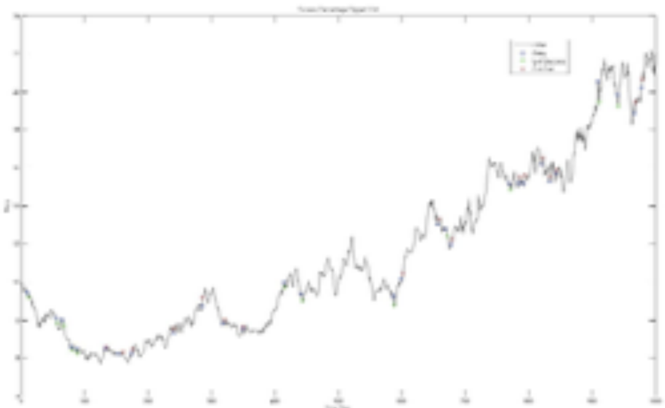
the information we need

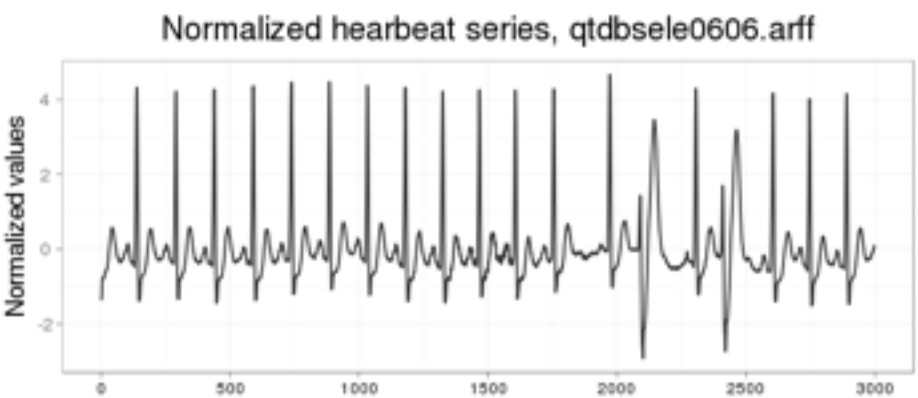
(efficiently)

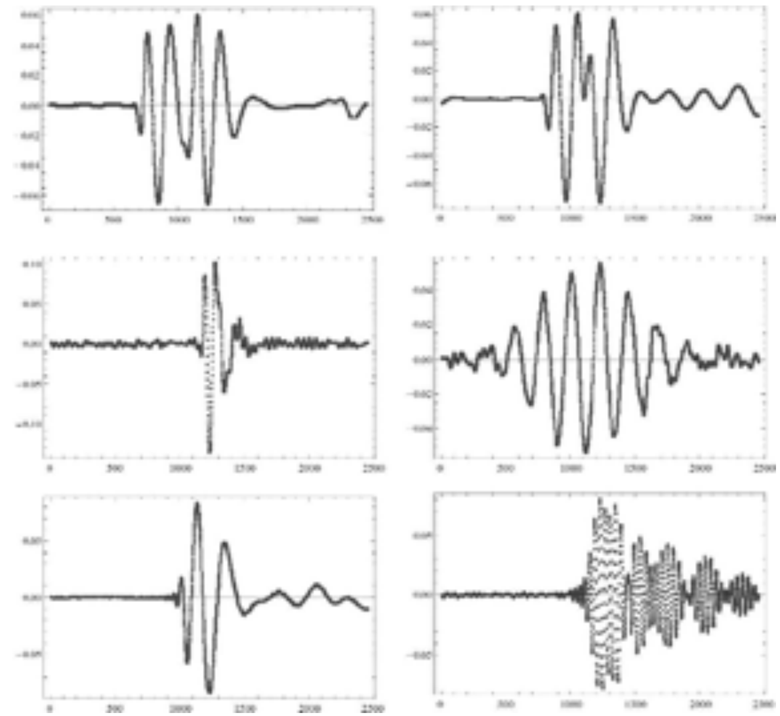
It depends on domain

Similarity (Equality)

Time Series



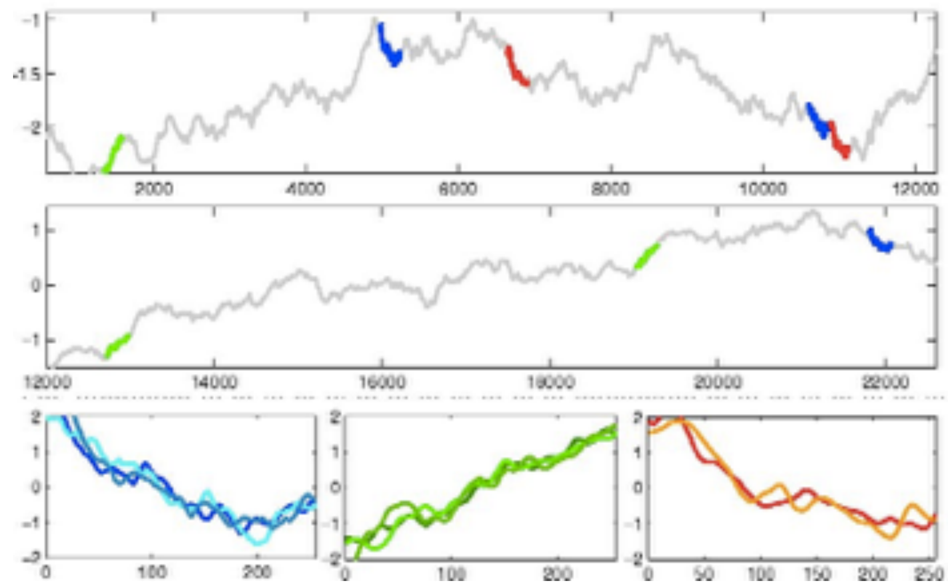

Normalized hearbeat series, qtdbsele0606.arff
Normalized values
4
2
0
-2
0
500
1000
1500
2000
2500
3000




HOW TO RETRIEVE

Time Series Retrieval

- Goal: to find a frame of data similar to that we use for querying



Nuno Castro, Paulo J. Azevedo
Significant motifs in time series.
Statistical Analysis and Data Mining 5(1): 35-53 (2012)

- Similarity measure
- Time Series Indexing

Similarity Measures

- A similarity measure is based on distance D between two series, so that
  - D(A,B) = D(B,A)
    Symmetry
  - D(A,A) = 0
    Constancy of Self-Similarity
  - D(A,B) >= 0
    Positivity
  - D(A,B) ≤ D(A,C) + D(B,C)
    Triangular Inequality
- The time series A,B can be directly represented within the time domain of within another space by some transform
  - Fourier
  - Wavelet
  - Gabor
  - PCA
  - Fuzzy Transform
  - etc.

Euclidean Similarity

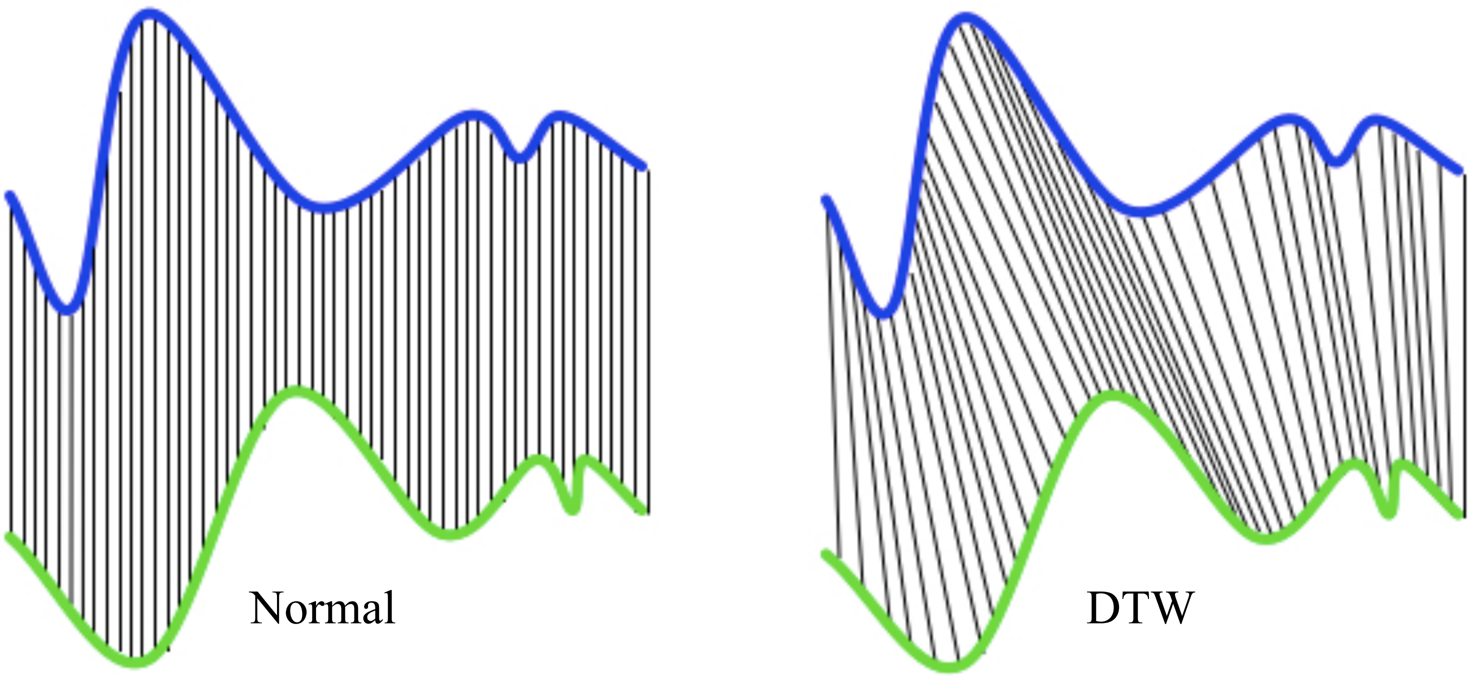
- Given two series:
  - A = x1,x2,…,xn
  - B = y1,y2,…,yn

$$\boxed{L_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}}$$

p=1 Manhattan distance
p=2 Euclidean distance
…
p=Infinity Maximum

Dynamic Time Warping (DTW)

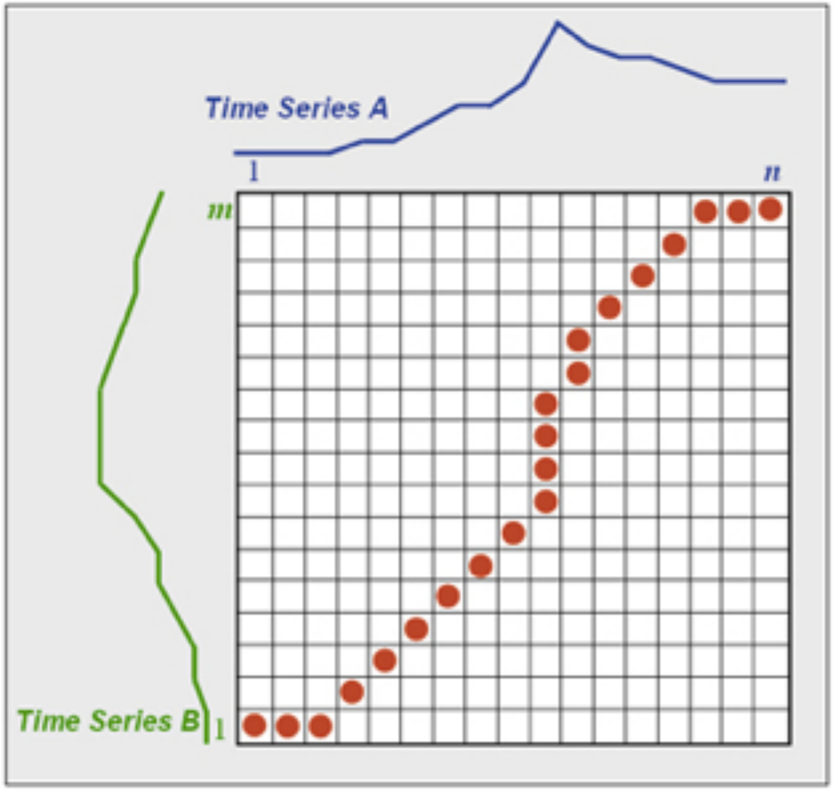
- Time series may vary in time and speed
- Dynamic Time Warping help the sequence re-alignment for simil

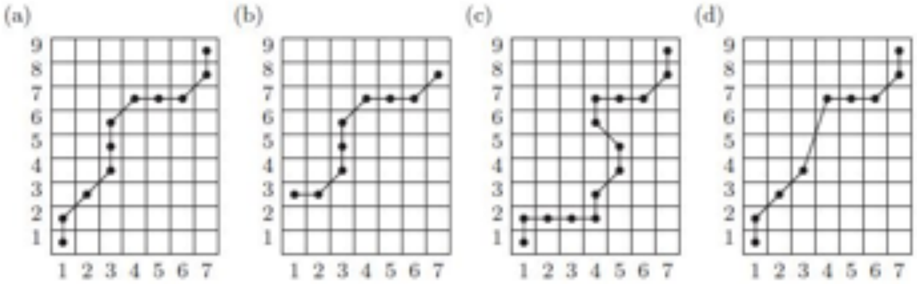


Sakoe, H. and Chiba, S., Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech and Signal Processing, 26(1) pp. 43– 49, 1978, ISSN: 0096-3518

Warping Path

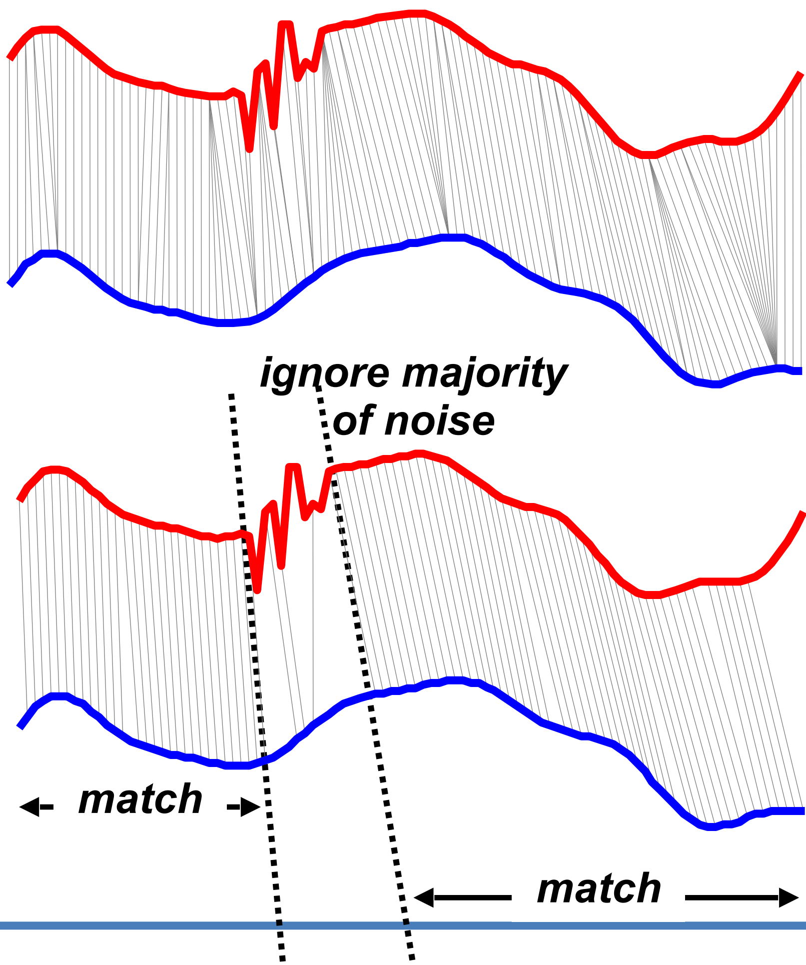
- The path which produces a sequence re-allignment.
- It is an optimization problem
- It can be solved efficiently by dynamic programming technique (O(n^2))





Longest Common Sub-Sequence (LCSS)

LCSS is more resilient to noise than DTW.



Disadvantages of DTW:

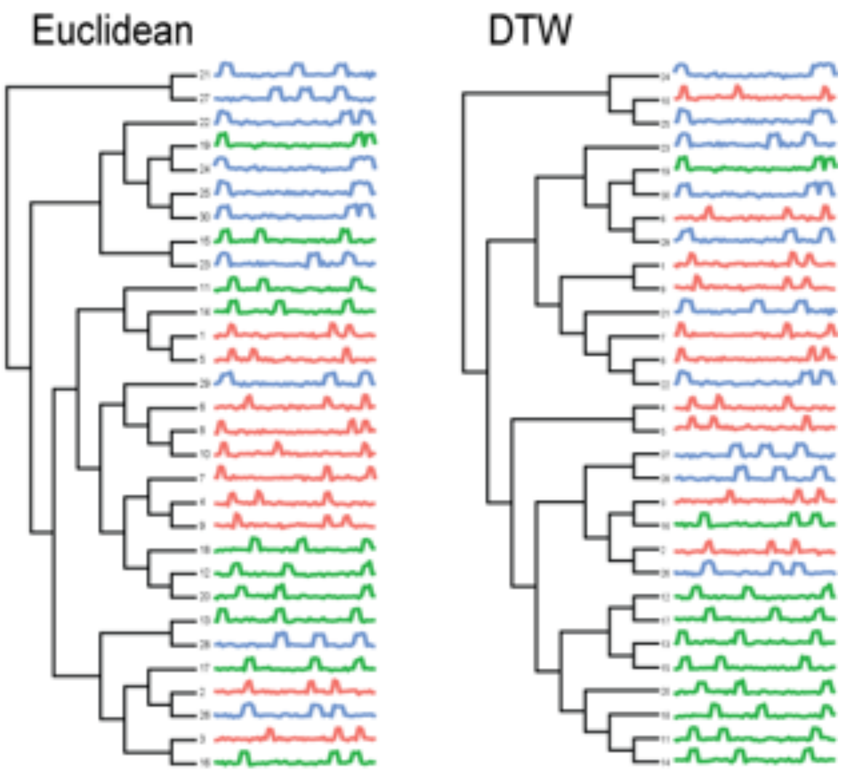
A. ***All points are matched***
B. ***Outliers can distort distance***
C. ***One-to-many mapping***

Advantages of LCSS:

A. ***Outlying values not matched***
B. ***Distance/Similarity distorted less***
C. ***Constraints in time & space***

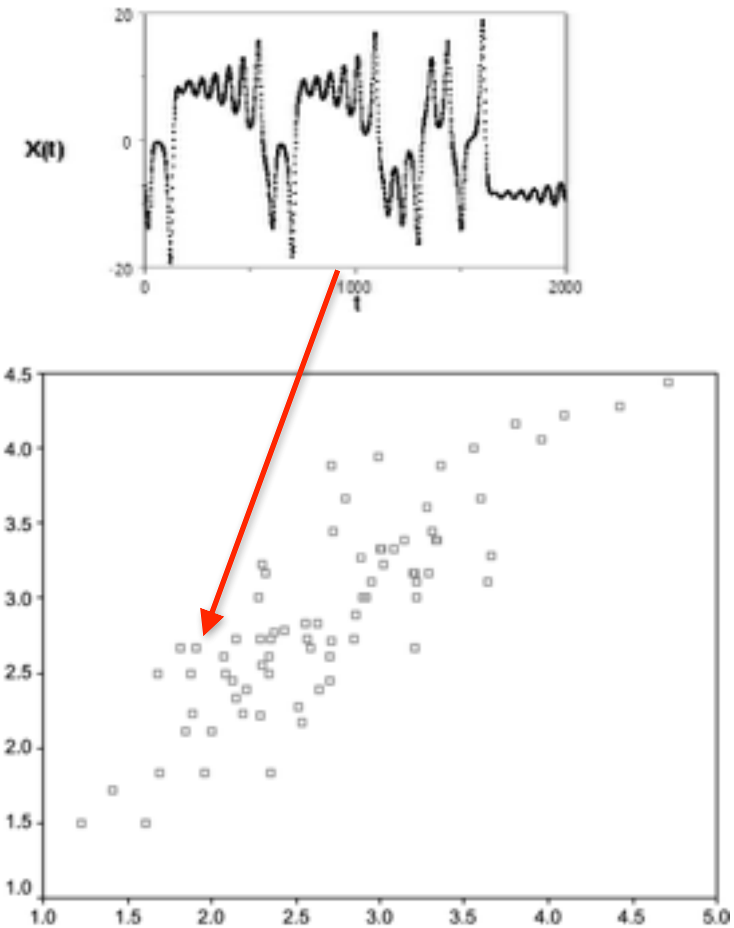
Hierarchical Clustering

- After we segmented the series (e.g., by LCSS), they are ready to be indexed
- We can build an index of segments by Hierarchical Clustering
- The index is used to retrieve faster the segments of interest



Feature Space

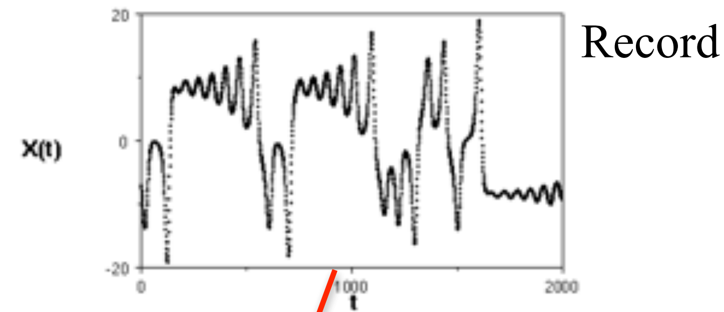
- Each time series becomes a point in a low-dimensional space, by means of some transform:
  - Fourier
  - Wavelet
  - …

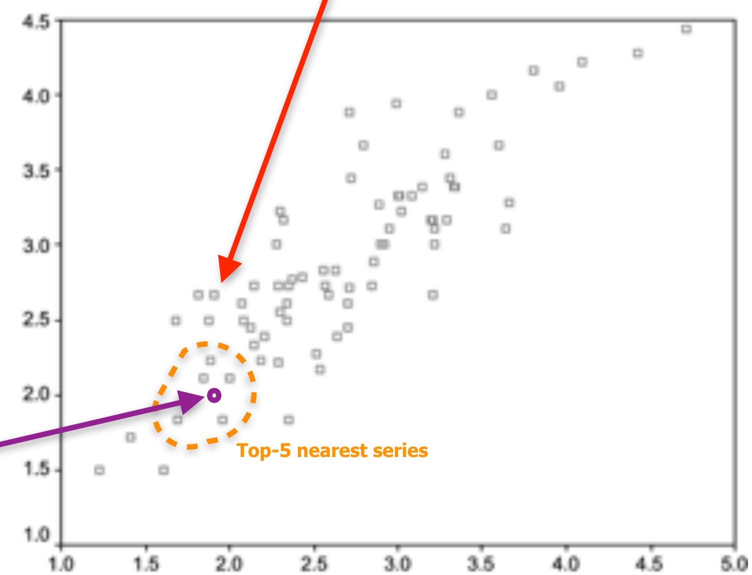


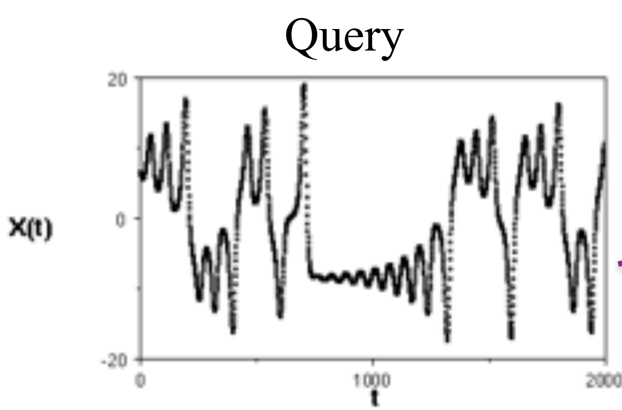
- (Dis-)Similarity is measured as distance between points in the feature space

Retrieval in Feature Space

- We can find the top K most similar time series
  - K-NN Algorithm
- Retrieval on large datasets can be improved by indexing the space
  - R-Tree, appropriate for spatial data

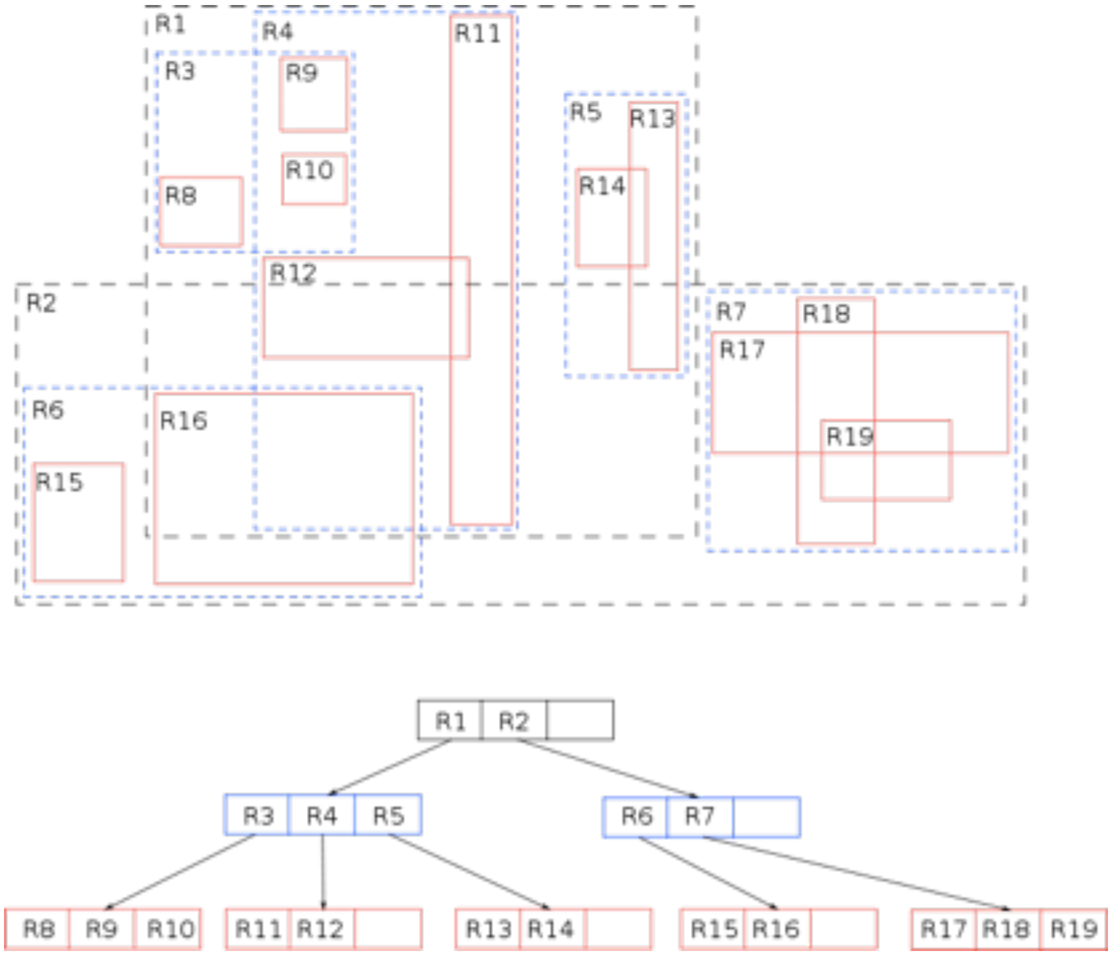


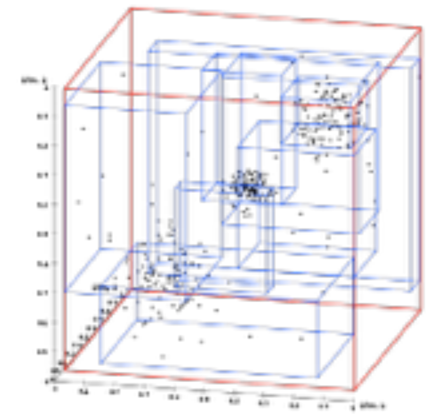


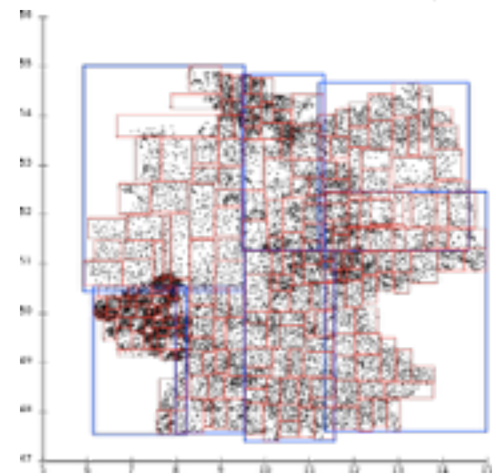


R-Tree Examples

Source: Wikipedia







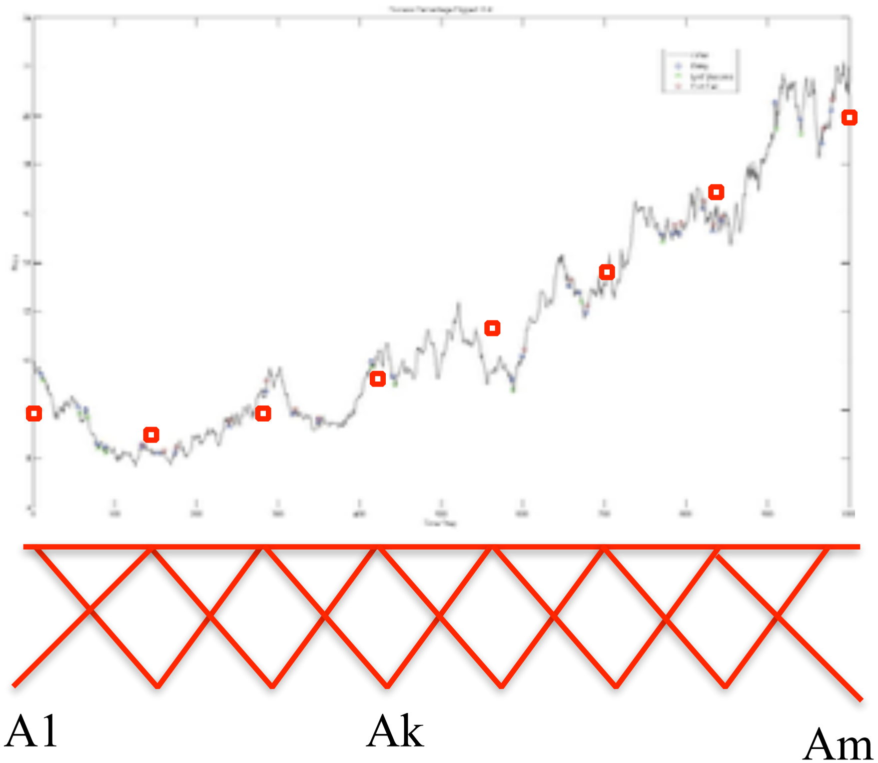
Fuzzy Transform

- It maps a time series at different time granularity

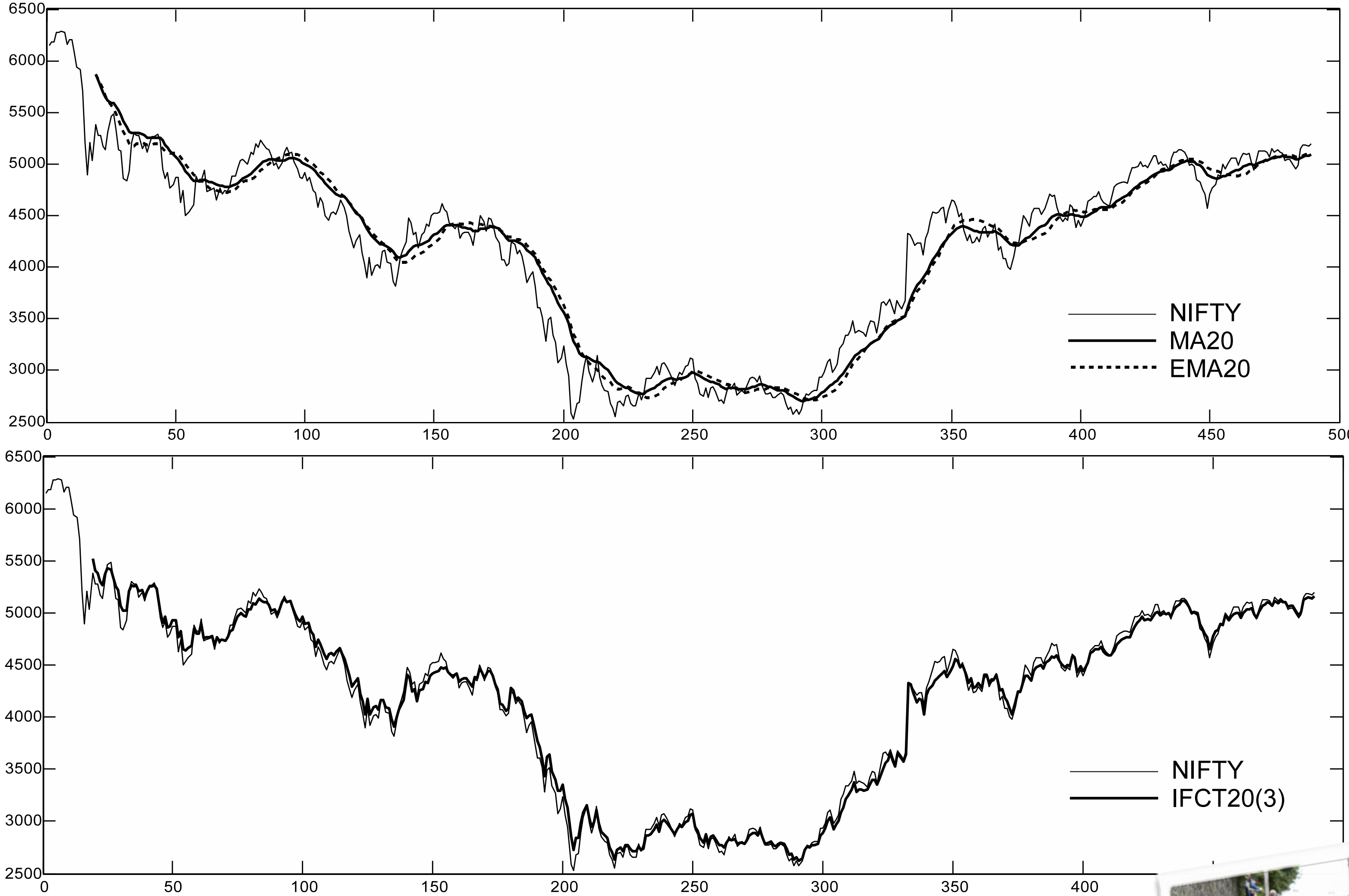
$$F_k = \sum_{i=1}^{n} x(i) A_k(i)$$

- Inverse Transform

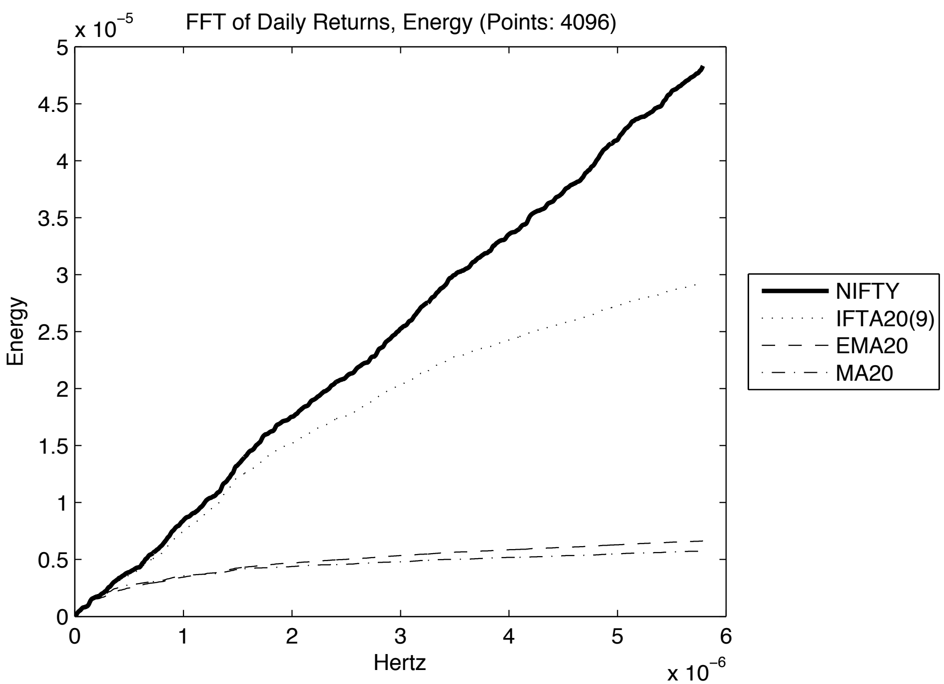
$$\hat{f}(i) = \frac{\sum_{k=1}^{m} F_k A_k(i)}{\sum_{k=1}^{m} A_k(i)}$$

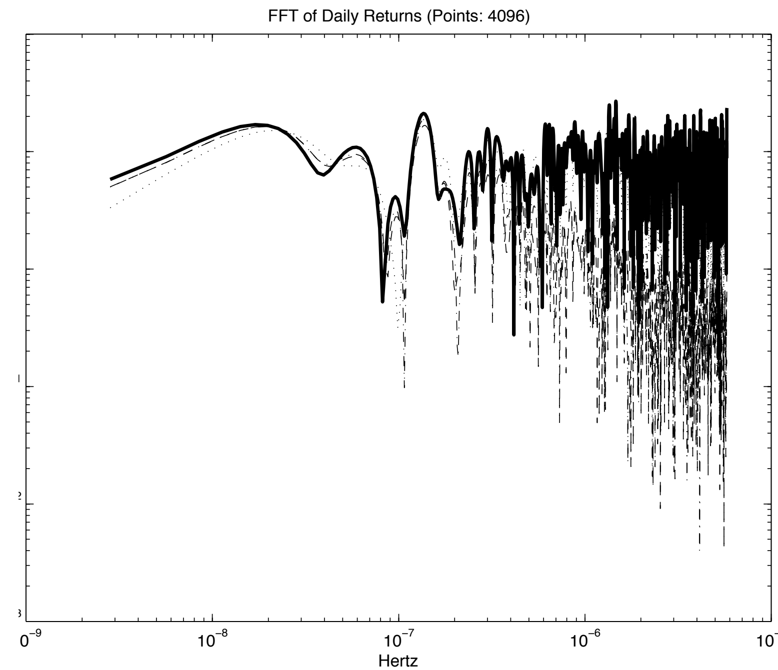


Reconstruction



Frequency Behavior





HOW TO ANALYZE

ICRR, University of Tokyo — May 23, 2014

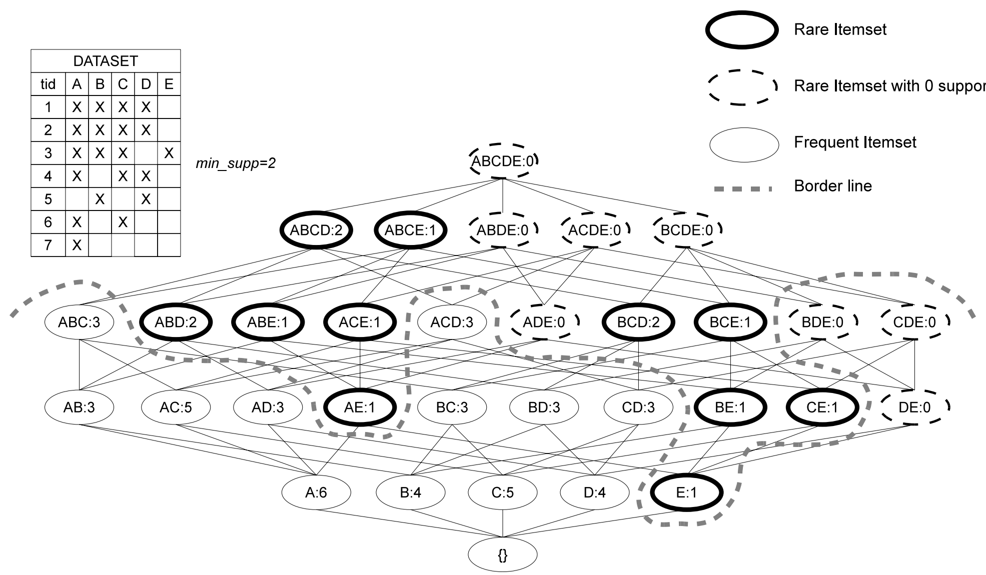
Analysis

Many many … many techniques

- Classification
  - Neural Networks
  - Kernel Machines (e.g., SVN)
  - Bayesian Methods
  - …
- Association Discovery
  - Frequent (and rare) itemsets
  - Association rules
  - Regression models
  - …
- Model fitting
- …

Complexity

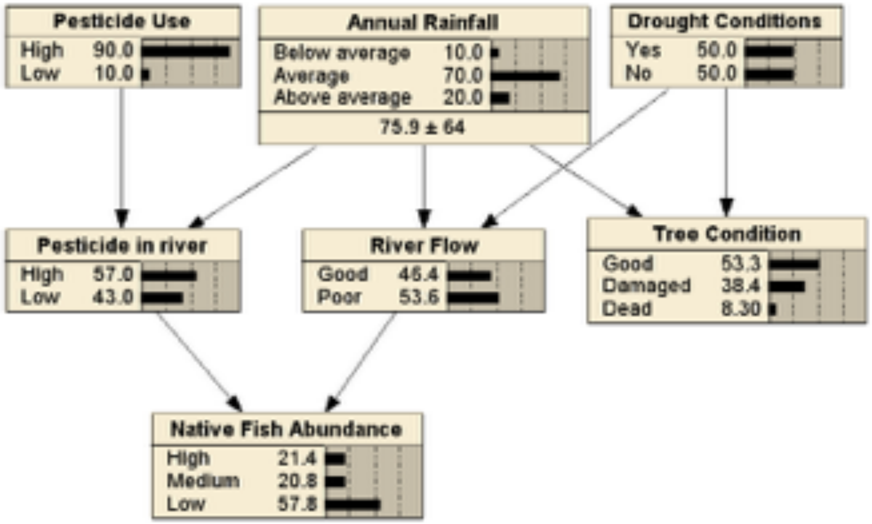
- The most of problems and algorithms are **NP-hard** (some NP-complete)
  - Generally difficult on medium-sized datasets
  - Almost infeasible on large datasets
- An example:
  - The search of frequent (rare) itemsets is combinatorial



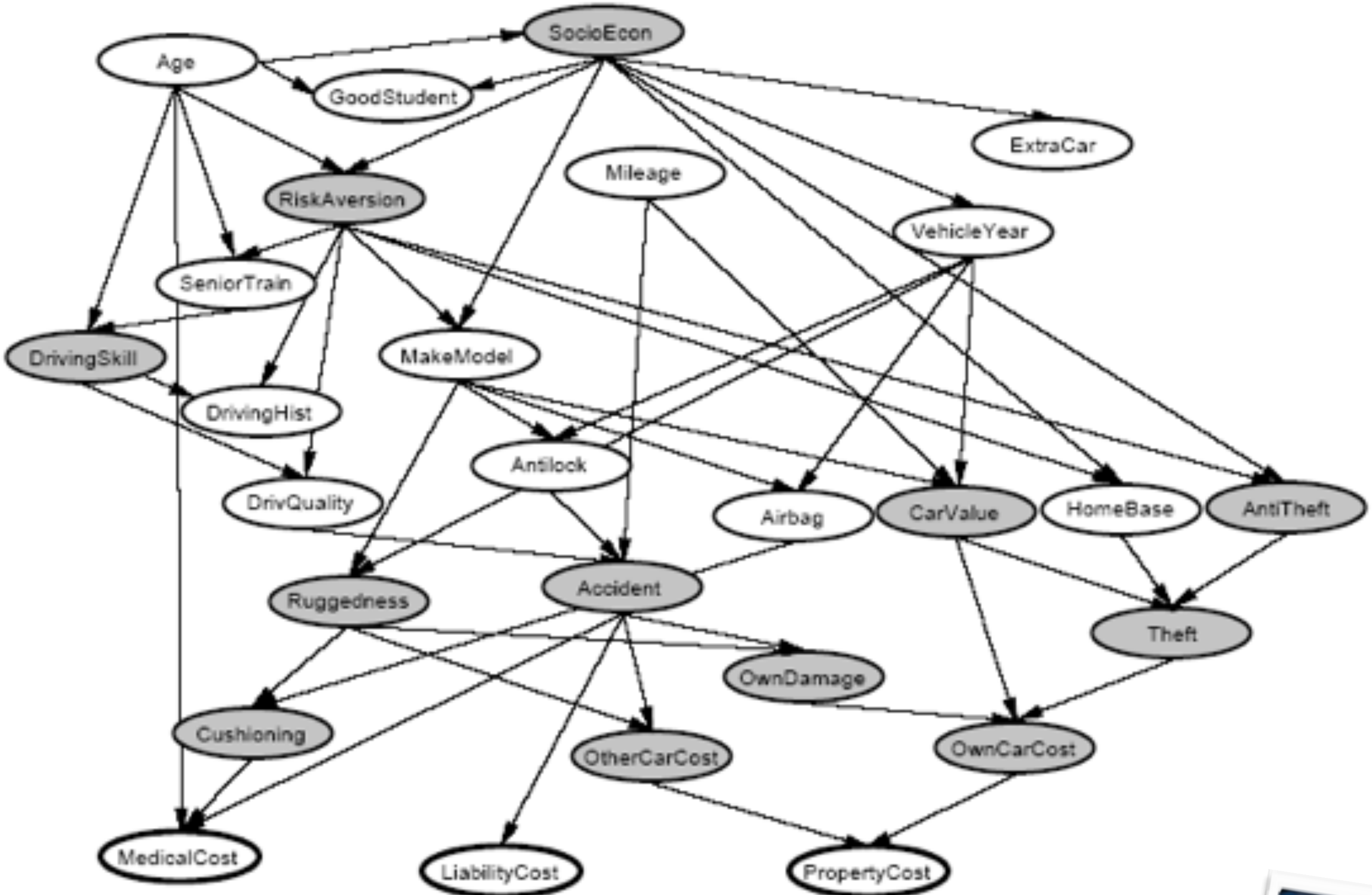
DATASET					
tid	A	B	C	D	E
1	X	X	X	X	
2	X	X	X	X	
3	X	X	X		X
4	X		X	X	
5		X		X	
6	X		X		
7	X				

Bayesian Networks

- Graphical models
- Nodes are random variables (discrete or continuous)
- Edge represent conditional dependencies between data
- Each node has associated CPT, Conditional probability table
- Acyclic
  - But time can be introduced by Dynamic Bayesian Networks (DBN)
- Structure and parameters can be learnt
  - by data
  - by experts
  - or by both (mixed or incremental)
- Robust method to compute a-posteriori probability

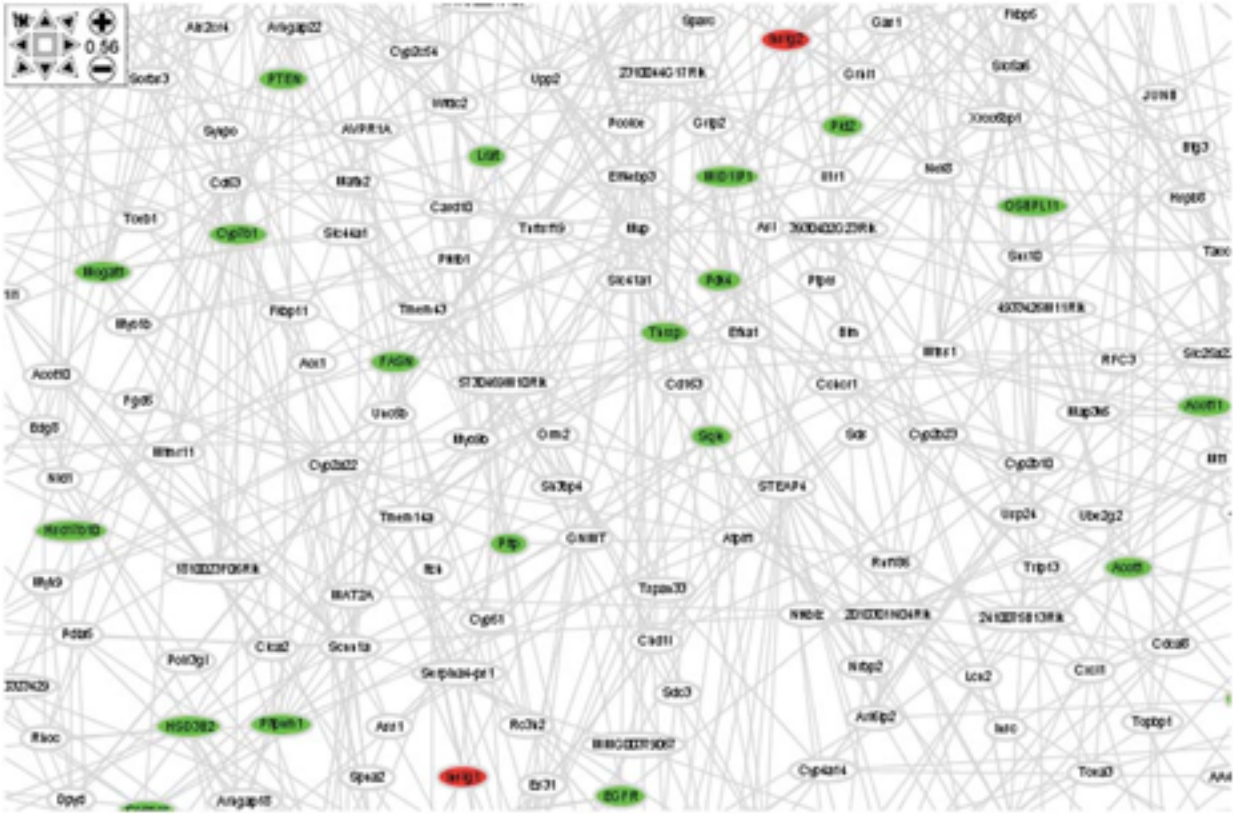


They can be larger …



A Bayesian network establishing relations between events on the automobile insurance domain.

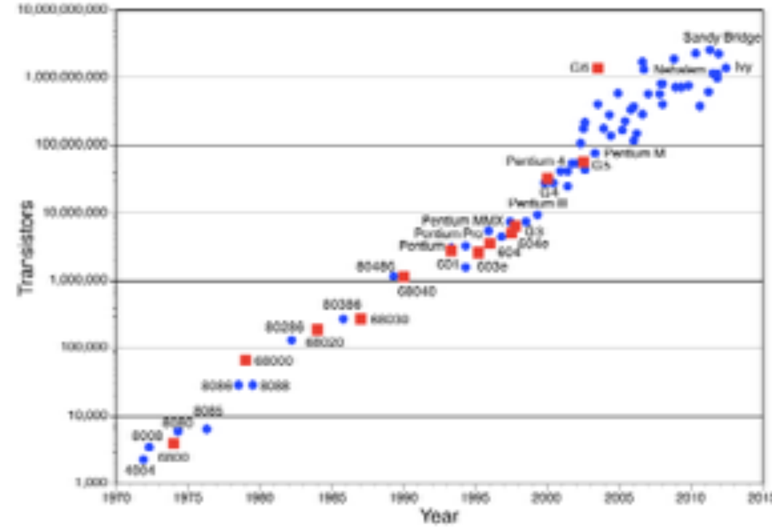
… and larger …

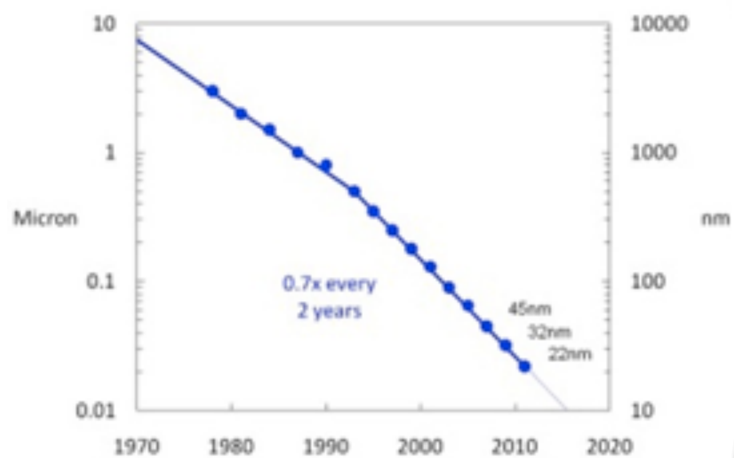


A portion of the core subnetwork, derived from the liver transcriptional subnetworks representative of gene expression signatures of the mouse models of the candidate genes. (Nature Genetics, 2009)

How to deal with complexity

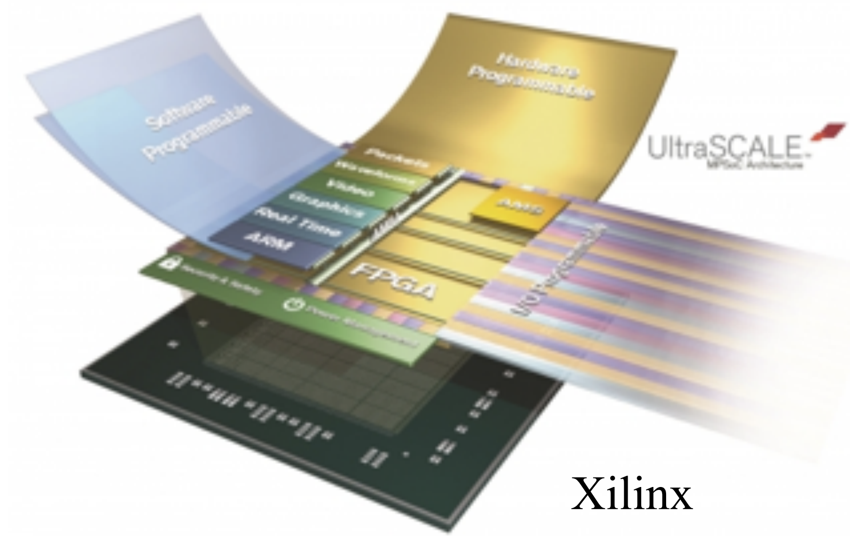
- We will need more and more computational power.
  - More Moore:
    - + integration scale
    - + clock frequency
  - More than Moore:
    - System on Chip (SoC)
    - FPGA



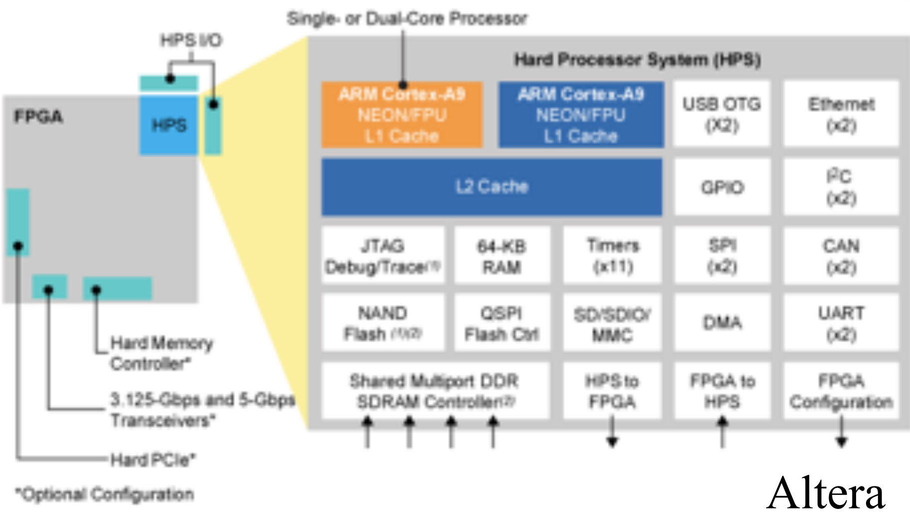


FPGA in SoC

- Systems-on-Chips
  - Faster
  - Less power
- FPGA
  - Programmable Hardware



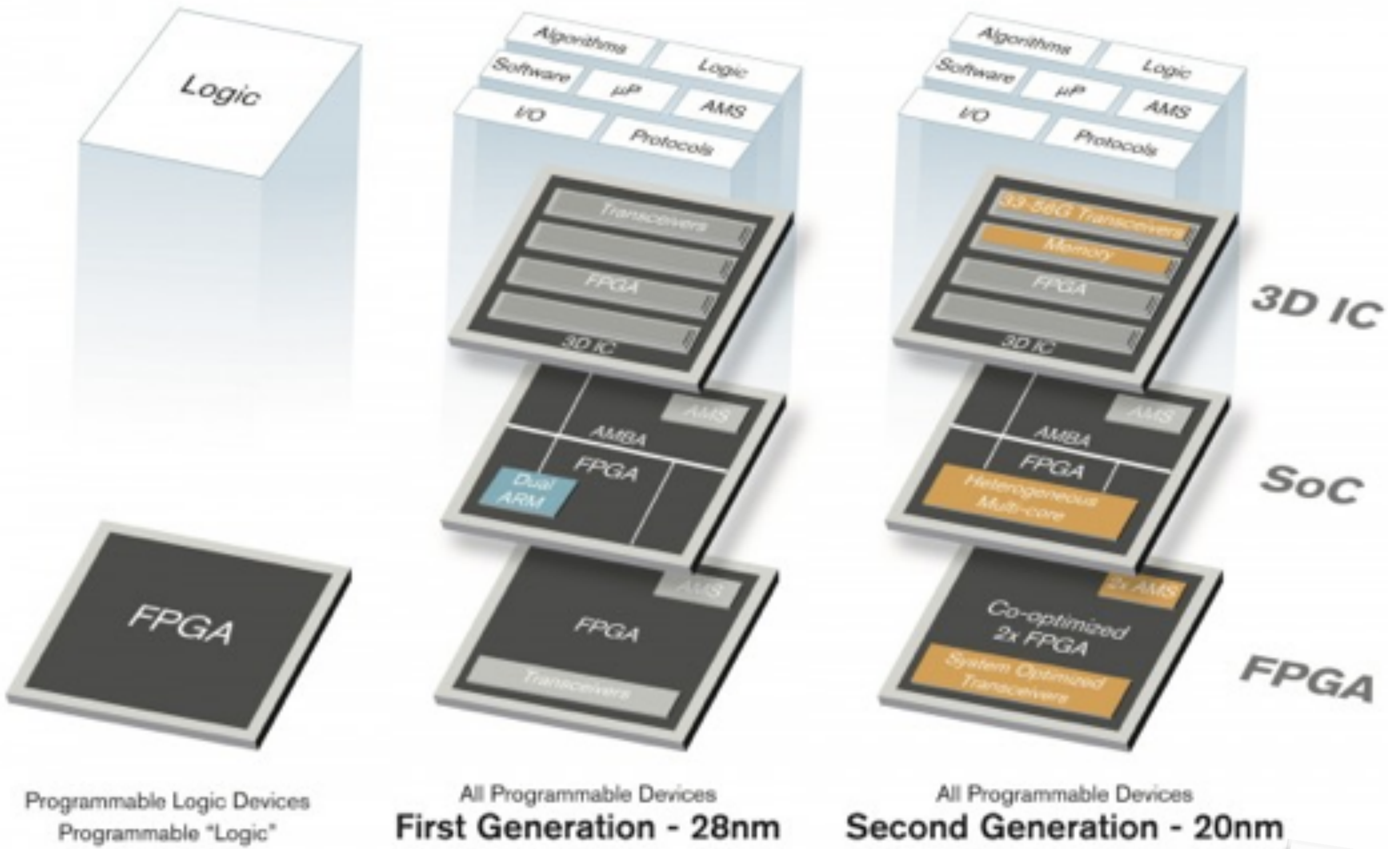
Xilinx



Altera

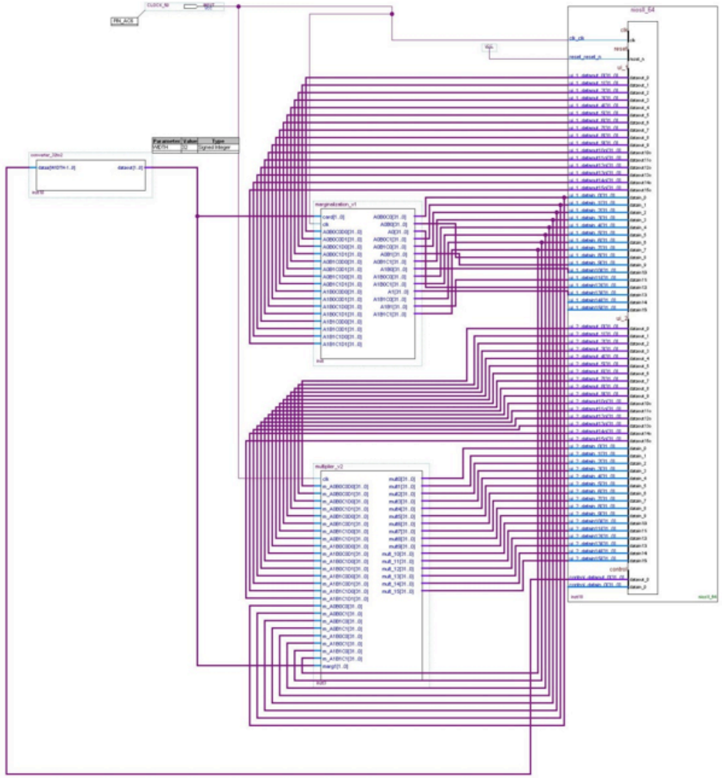
- Silicon Convergence
- Reconfigurable Computing
  - Make the device you need, when you need.

3D IC 20nm (Xilinx)

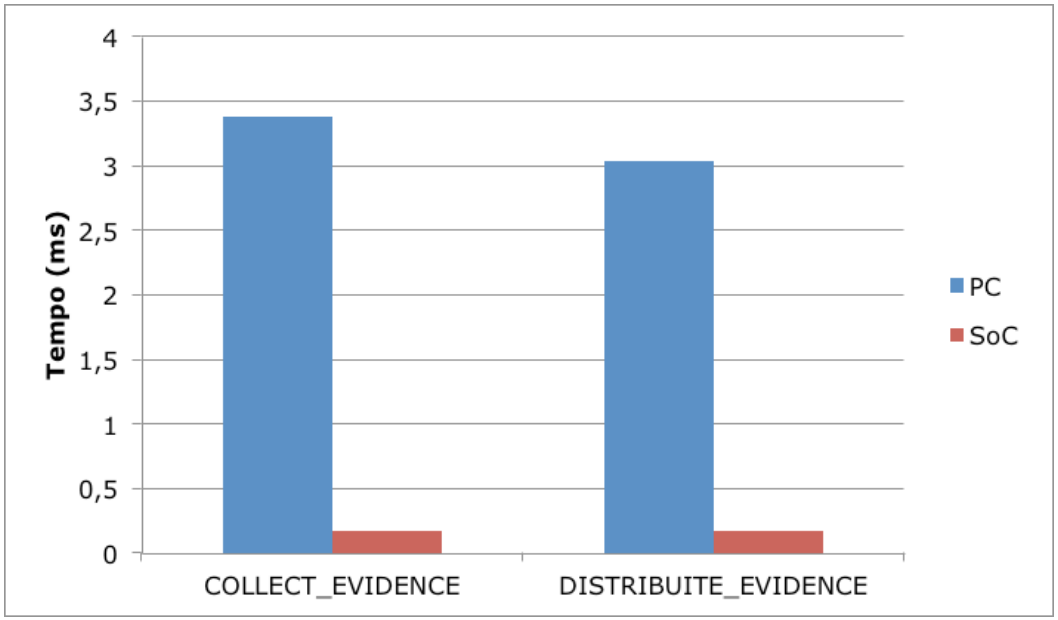


A prototype at our Lab

- A Bayesian coprocessor based on SoC in FPGA
- Based on Altera Stratix IV chipset and Nios Architecture
  - Memory on chip
  - Bayesian device (Memory Mapped)
- Two levels
  - Evidence propagation is controlled via software by Nios processor
  - Clique computation is perfumed by hardware



Experimental Results



	PC	SoC
COLLECT_EVIDENCE	3.3765 ms	0.1731 ms
DISTRIBUITE_EVIDENCE	3.0313 ms	0.1742 ms
FULL STEP	**6.4078 ms**	**0.3473 ms**

Conclusions

- Big Data is a "Big" trend in IT
- Data are easy to collect, but to benefit of them we need to change the way we manage them
- Advances in how data are stored, retrieved and analyzed pose the basis for a technology shift
- We need to rethink the relationship between software and hardware
- KAGRA, being one of the major experiment in Physics for the following years, might take advantage of some of these advances

Big Data Science. Any use for KAGRA?

Luigi Troiano
University of Sannio
troiano@unisannio.it

ありがとう
ございます。