Coping with Big Data Volume and Variety

Jiaheng Lu
University of Helsinki, Finland

Big Data: 4Vs

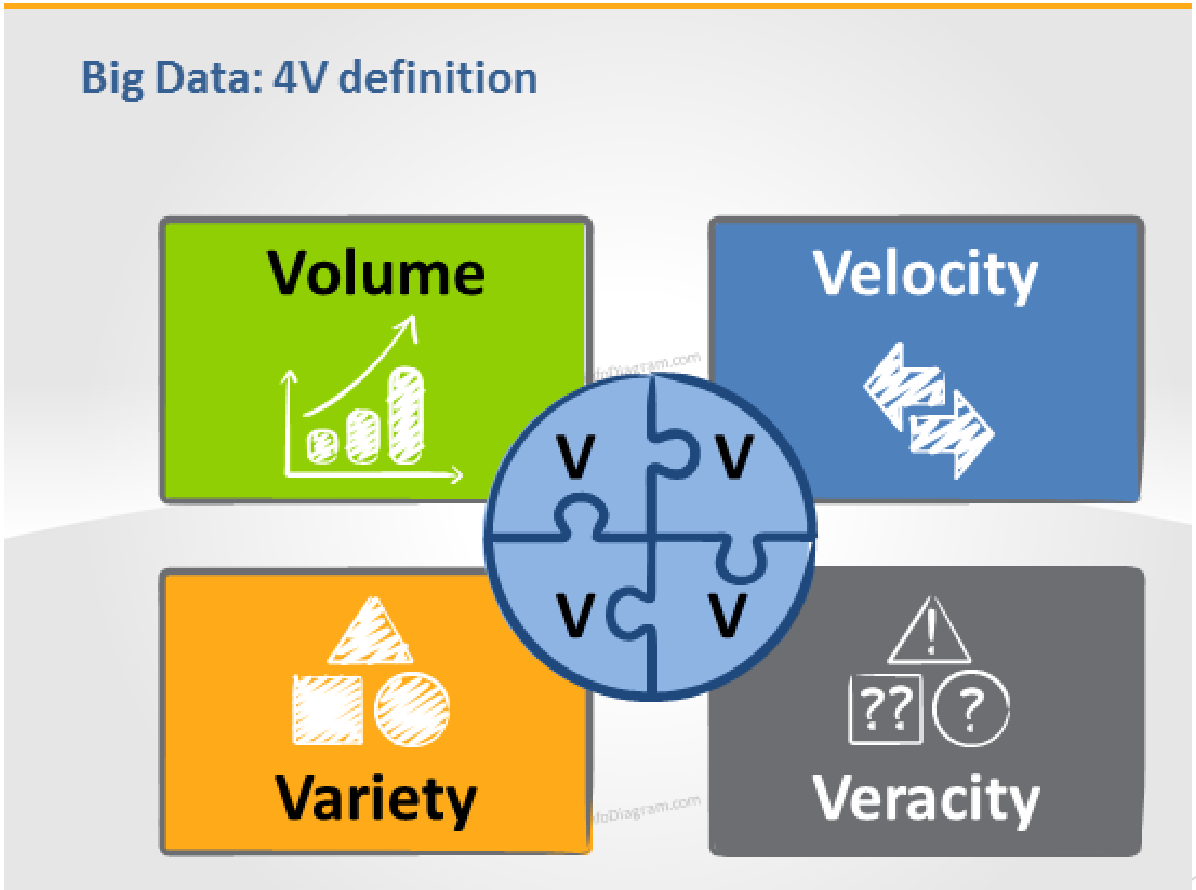


Photo downloaded from: https://blog.infodiagram.com/2014/04/visualizing-big-data-concepts-strong.html

Hadoop and Spark platform optimization

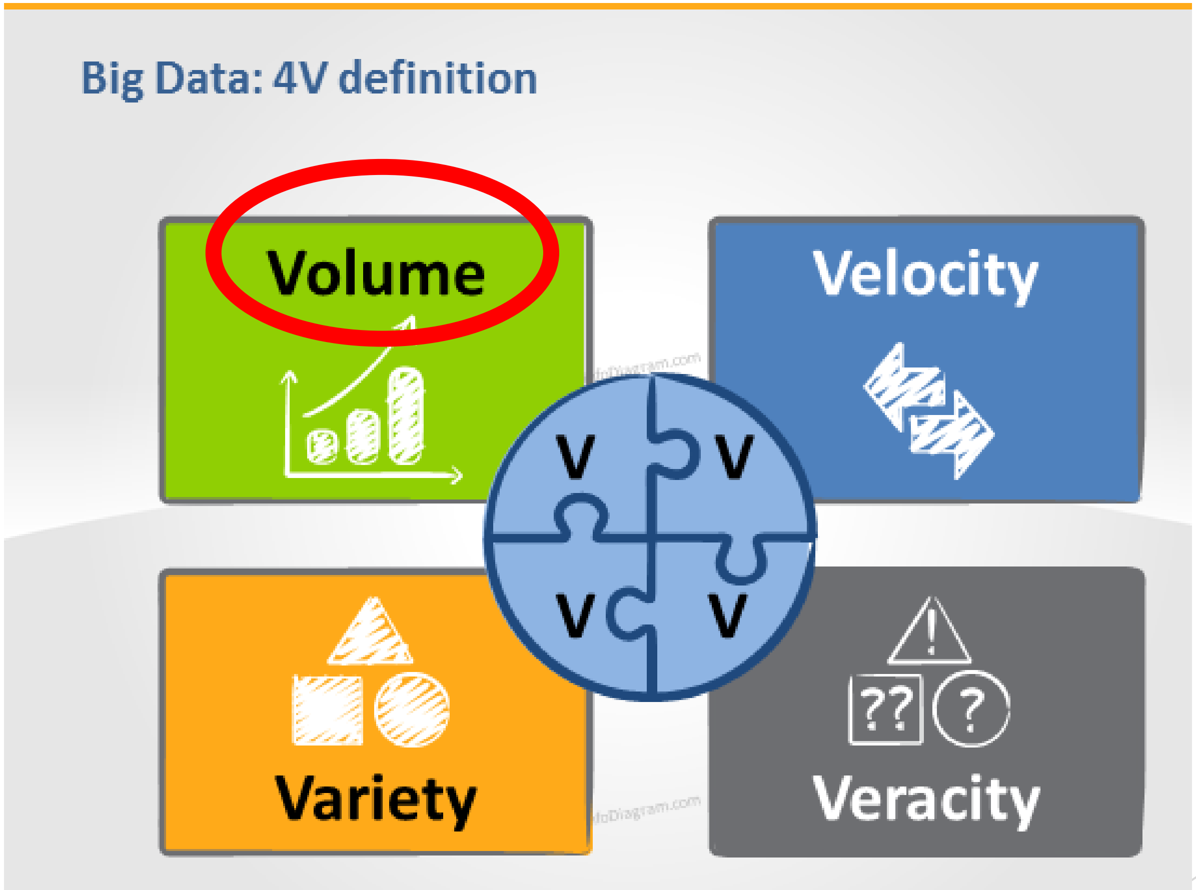


Photo downloaded from: https://blog.infodiagram.com/2014/04/visualizing-big-data-concepts-strong.html

Multi-model databases: quantum framework and category theory

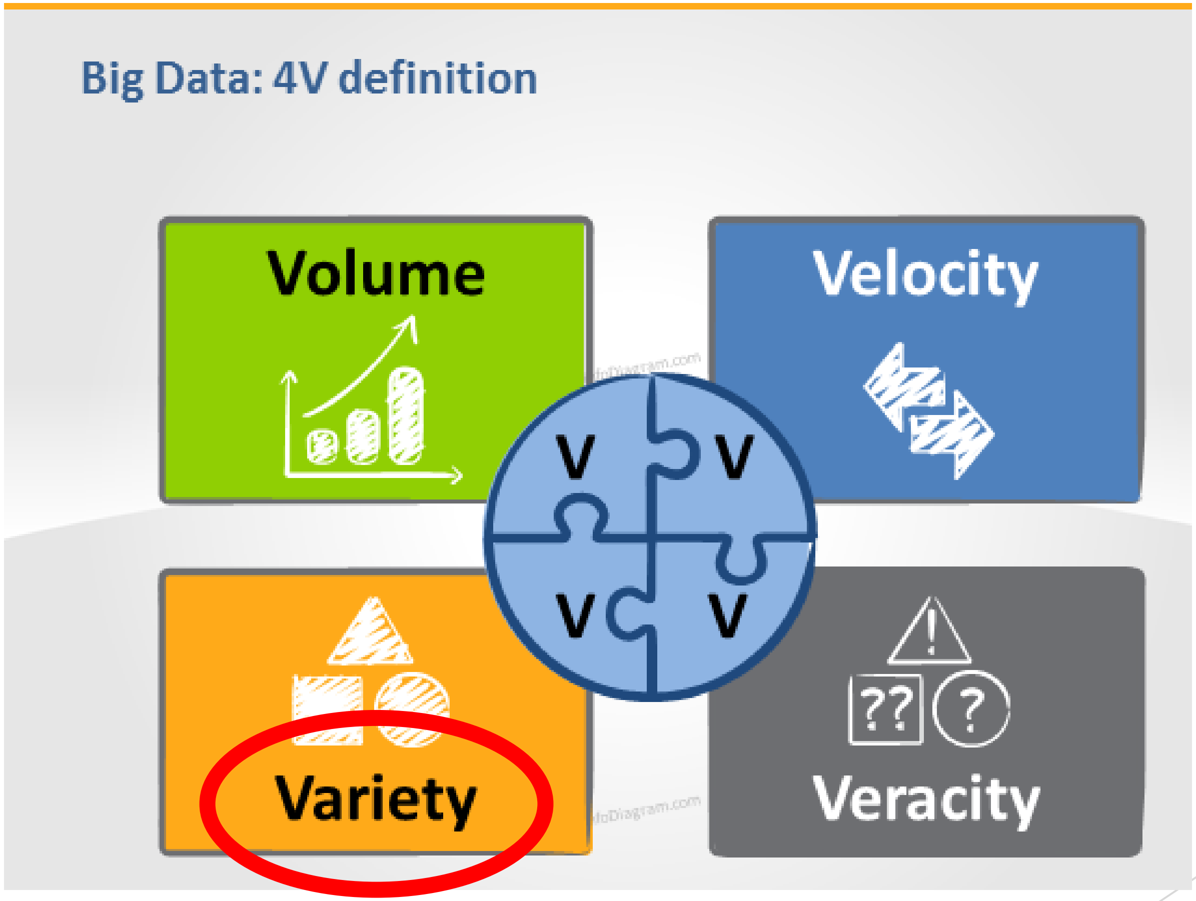


Photo downloaded from: https://blog.infodiagram.com/2014/04/visualizing-big-data-concepts-strong.html

Outline

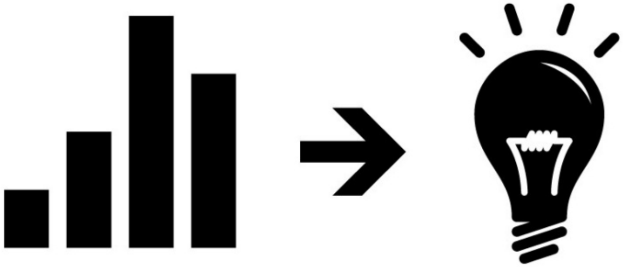
- Big data platform optimization (10 mins)
  - Motivation
  - Two main principles and approaches
  - Experimental results
- Multi-model databases (10 mins)
  - Overview
  - Quantum framework
  - Category theory

Optimizing parameters in Hadoop and Spark platforms

Analysis in the Big Data Era

Data Analysis **Decision making**

Massive data → **Insights** → **Saving and Revenue**



- Key to success = Timely and Cost-effective analysis

Analysis in the Big Data Era

- Popular big data platform：Hadoop and Spark
- Burden on users
  - Responsible for provisioning and configuration
  - Usually lack expertise to tune the system

Analysis in the Big Data Era

- Popular big data platform：Hadoop and Spark
- Burden on users
  - Responsible for provisioning and configuration
  - Usually lack expertise to tune the system

As a data scientist, I do not know how to improve the efficiency of my job?

Analysis in the Big Data Era

- Popular big data platform： Hadoop and Spark
- Burden on users: provision and tuning
- Effect of system-tuning for jobs

	Tuned vs. Default
Running time	Often 10x
System resource utilization	Often 10x
Others	Well tuned jobs may avoid failures like OOM, out of disk, job time out, etc.

Good performance after tuning

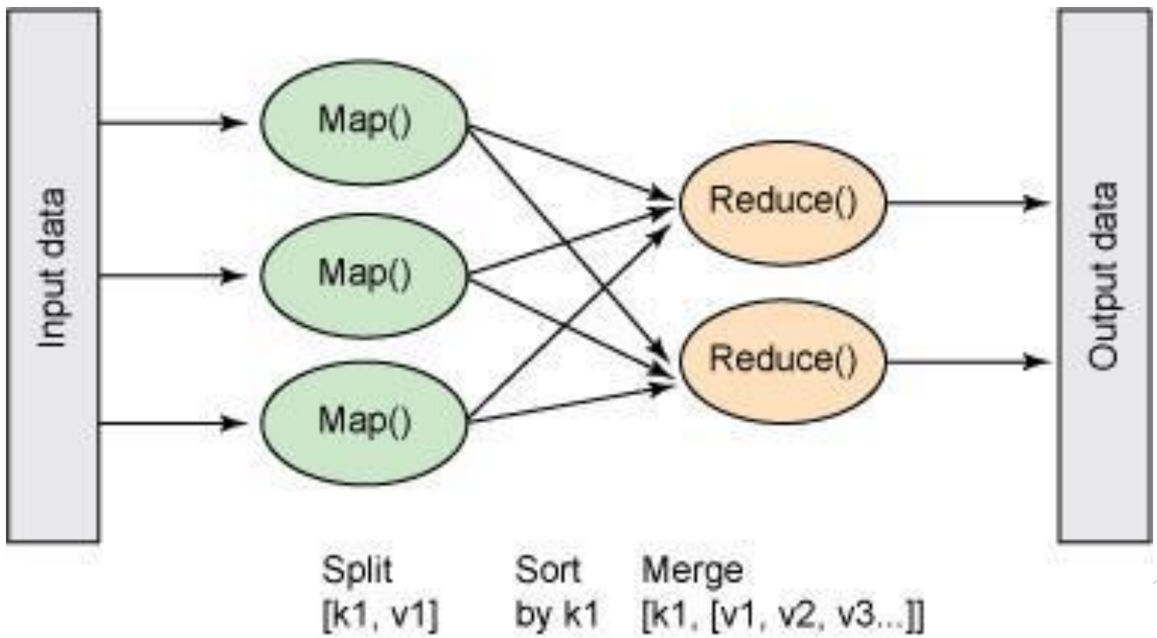
Automatic job optimization toolkit

- NOT our goal: Change the codes of the system to improve the efficiency
- Our goal: Configure the parameters to achieve good performance

Our system is easy to be used in the existing Hadoop and Spark system

Problem definition

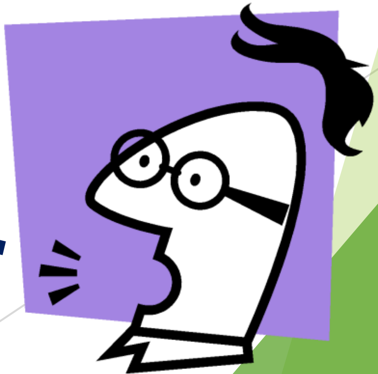
- Given a MapReduce or Spark job with input data and running cluster, we find the setting of parameters that optimize the execution time of the job. (i.e. minimize the job execution time)



Challenges: too many parameters!

Symbol	Parameter	Groups
N_m	mapred.map.tasks	Map input split
N_r	mapred.reduce.tasks	Reduce output
s_{min}	mapred.min.split.size	Map input split
s_{max}	mapred.max.split.size	Map input split
B_m	io.sort.mb	Map output buffer
r_{rec}	io.sort.record.percent	Map output buffer
c	mapred.compress.map.output	Map output compr.
N_{copy}	mapred.reduce.parallel.copies	Reduce copy
N_{sf}	io.sort.factor	Reduce input
B_r	mapred.job.reduce.input.buffer.percent	Reduce input
S_{OB}	dfs.block.size	Reduce output
$N_{ms}(i)$	mapred.tasktracker.map.tasks.maximum	Set by HAC
$N_{rs}(i)$	mapred.tasktracker.reduce.tasks.maximum	Set by HAC

There are more than **190 parameters** in Hadoop!



Two key ideas in job optimizer

1.Reduce search space!

190 → 13 → 4

KEEP
CALM
BECAUSE
LESS IS MORE

13 parameters we tune

Symbol	Parameter	Groups
N_m	mapred.map.tasks	Map input split
N_r	mapred.reduce.tasks	Reduce output
$s_{\min}$	mapred.min.split.size	Map input split
$s_{\max}$	mapred.max.split.size	Map input split
B_m	io.sort.mb	Map output buffer
r_{rec}	io.sort.record.percent	Map output buffer
c	mapred.compress.map.output	Map output compr.
N_{copy}	mapred.reduce.parallel. copies	Reduce copy
N_{sf}	io.sort.factor	Reduce input
B_r	mapred.job.reduce .input.buffer.percent	Reduce input
S_{OB}	dfs.block.size	Reduce output
$N_{ms}(i)$	mapred.tasktracker .map.tasks.maximum	Set by HAC
$N_{rs}(i)$	mapred.tasktracker .reduce.tasks.maximum	Set by HAC

Four key factors

- We identify four key factors (parameters) to model a MapReduce job execution
  - The number of Map task waves *m*. (number of Map tasks)
  - The number of Reduce task waves r. (number of Reduce tasks)
  - The Map output compression option c. (true or false)
  - The copy speed in the Shuffle phase v (number of parallel copiers)

Cost model

- **Producer:** the time to produce the Map outputs in m waves

$$\mathcal{T}_{producer} = t_{map}(\mathcal{D}, c) + t_{schedule}(m) + t_{cs}(\mathcal{D}, c, v, r)$$

- **Transporter:** the non-overlapped time to transport Map outputs to the Reduce side

$$\mathcal{T}_{trasporter} = \min\left(\left(\frac{m-1}{mrv} \cdot D_s - \frac{m-1}{m} \cdot \mathcal{T}_{producer} - t_{lrw}\left(\frac{m-1}{mr} \cdot D_s\right)\right), 0\right) + \frac{(2mr - m - r + 1)}{mrv} \cdot D_s + t_{lrw}(D_s))$$

- **Consumer:** the time to produce Reduce outputs in r waves

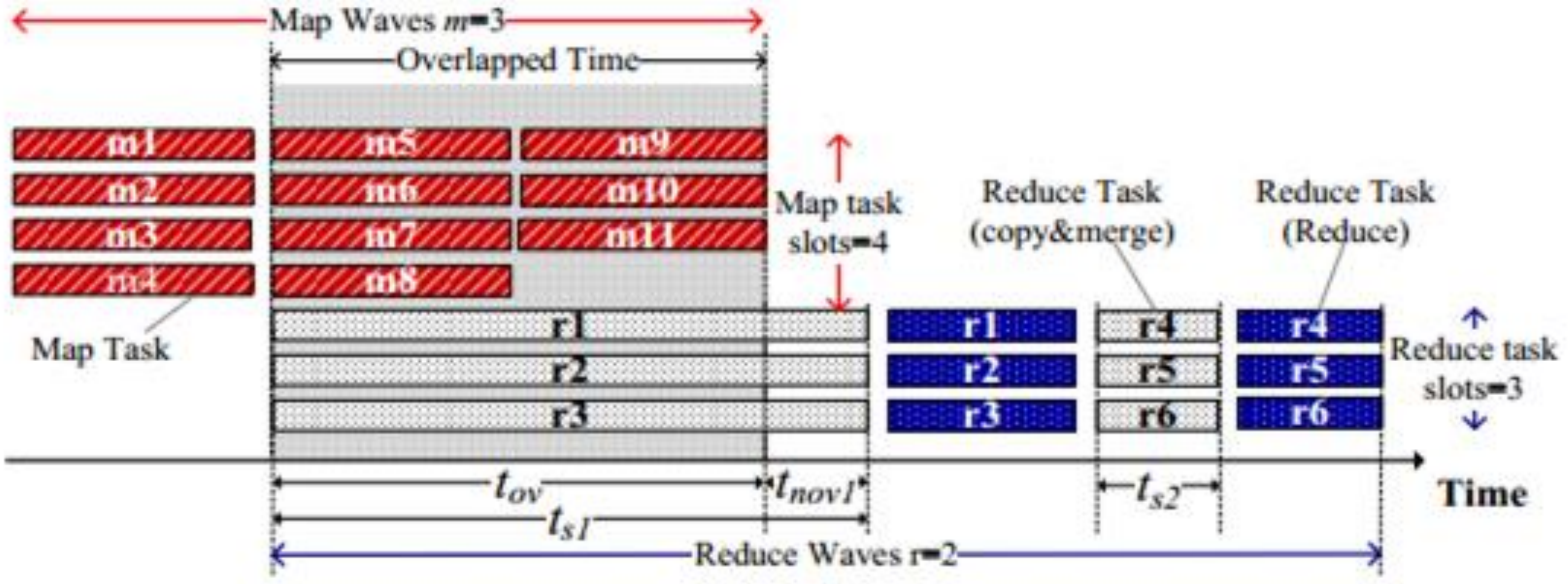
$$\mathcal{T}_{consumer} = t_{reduce}(\mathcal{D}, c) + t_{schedule}(r) - t_{lrw}(B_r) \cdot r$$

Two key ideas in job optimizer

2. Keep everything busy!

- CPU: map, reduce and compression
- I/O: sort and merge
- Network: shuffle

Keep map and shuffle parallel



MRTunner approach:

Given a new job for tuning

↓

Retrieve the profile data

↓

Search for the four key parameters

↓

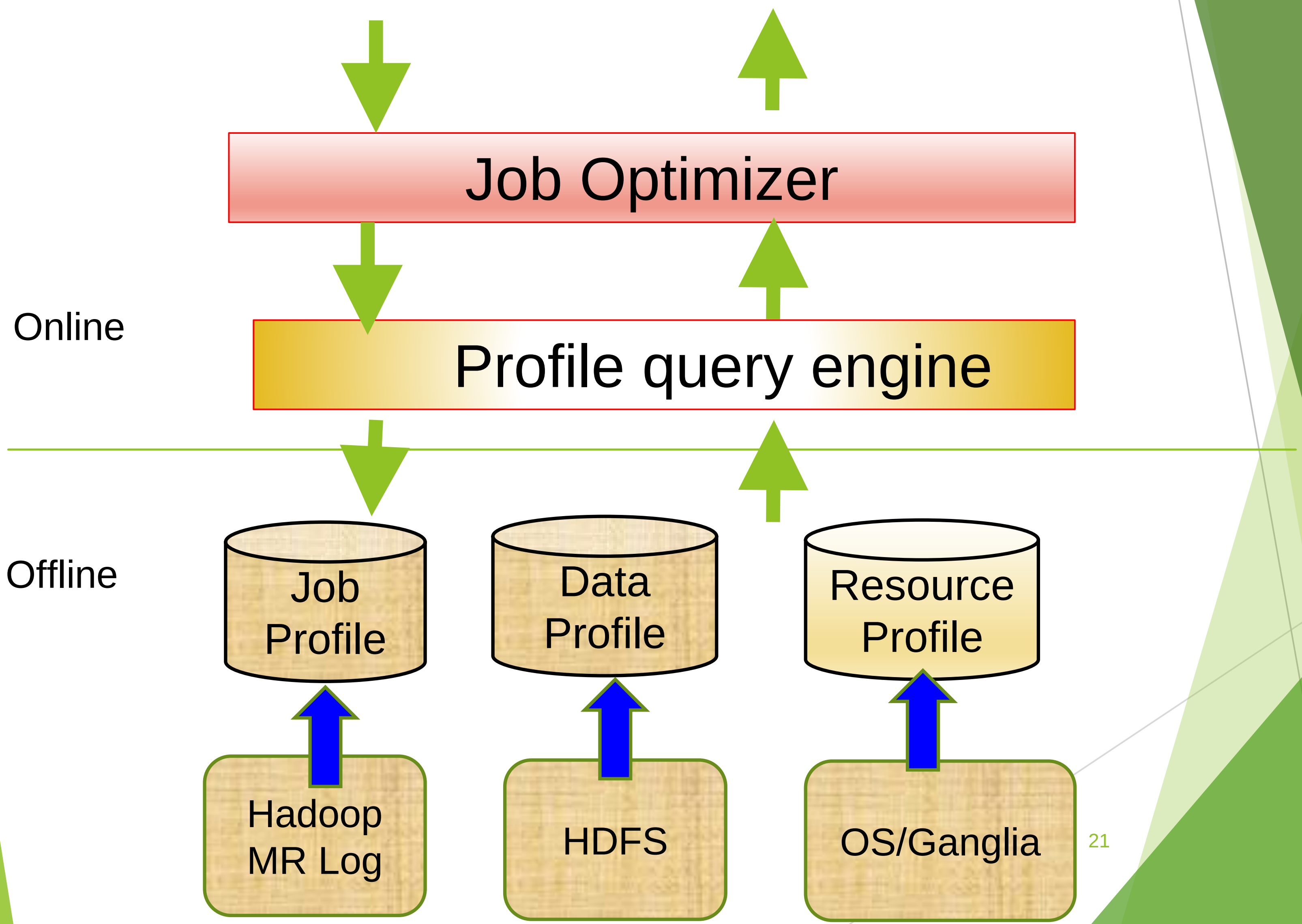
Compute the 13 parameters

↓

Configure and run

Good performance after tuning

Architecture：



Profile data

- Job profile
  - Selectivity of Map input/output
  - Selectivity of Reduce input/output
  - Ratio of Map output compression
- Data profile
  - Data Size
  - Distribution of input key/value
- System profile
  - Number of machines
  - Network throughput
  - Compression/Decompression throughput
  - Overhead to create a map or reduce task

Experimental evaluation

- Performance Comparison
  - **Hadoop-X (Commercial Hadoop):**
  - **Starfish**: Parameters advised by Starfish
  - **MRTuner:** Parameters advised by MRTuner

- Workloads
  - Terasort
  - N-gram
  - Pagerank

Effectiveness of MRTuner Job Optimizer

Running time of jobs

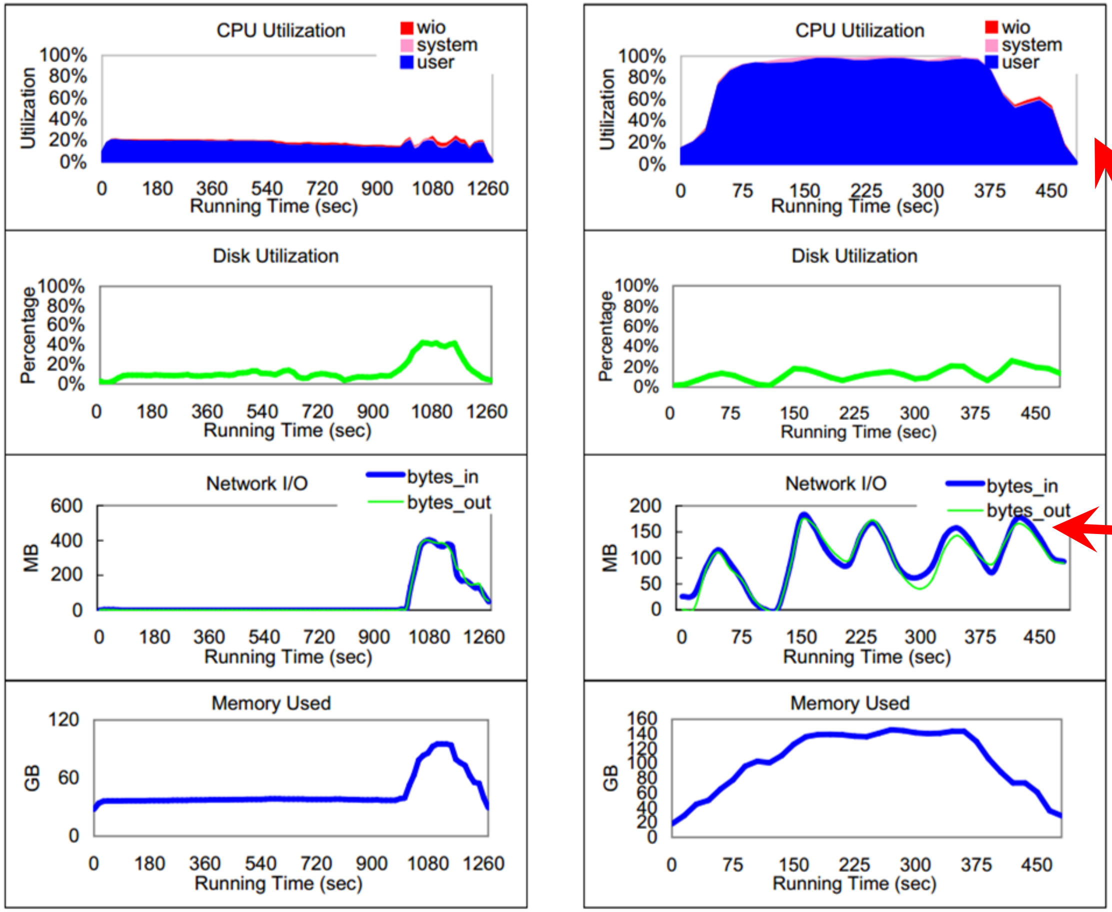
Commercial Hadoop-X

JobName	ID	Clu-ster	Input (GB)	Hadoop -X(sec)	MRTuner (sec)	Speed -up
Terasort	TS-1	$\mathcal{A}$	10	469	278	1.7
Terasort	TS-2	$\mathcal{A}$	50	2109	1122	1.87
Terasort	TS-3	$\mathcal{B}$	200	767	295	2.60
Terasort	TS-4	$\mathcal{B}$	1000	6274	2192	2.86
N-Gram	NG-1	$\mathcal{A}$	0.18	4364	192	22.7
N-Gram	NG-2	$\mathcal{A}$	0.7	N/A	661	∞
N-Gram	NG-3	$\mathcal{A}$	1.4	N/A	1064	∞
N-Gram	NG-4	$\mathcal{B}$	1.4	1100	249	4.41
N-Gram	NG-5	$\mathcal{B}$	2.8	1292	452	2.86
N-Gram	NG-6	$\mathcal{B}$	5.6	1630	930	1.75
PR(Trans.)	PR-1	$\mathcal{A}$	3.23	962	446	2.2
PR(Deg.)	PR-2	$\mathcal{A}$	Inter	49	41	1.2
PR(Iter.)	PR-3	$\mathcal{A}$	Inter	933	639	1.5
PR(Trans.)	PR-4	$\mathcal{B}$	3.23	148	65	2.28
PR(Deg.)	PR-5	$\mathcal{B}$	Inter	24	22	1.09
PR(Iter.)	PR-6	$\mathcal{B}$	Inter	190	82	2.32

For N-gram job, MRTuner obtains more than 20x speedup than Hadoop-X

Comparison between Hadoop-X and MRTuner (N-gram)

Cluster-wide Resource Usage from Ganglia

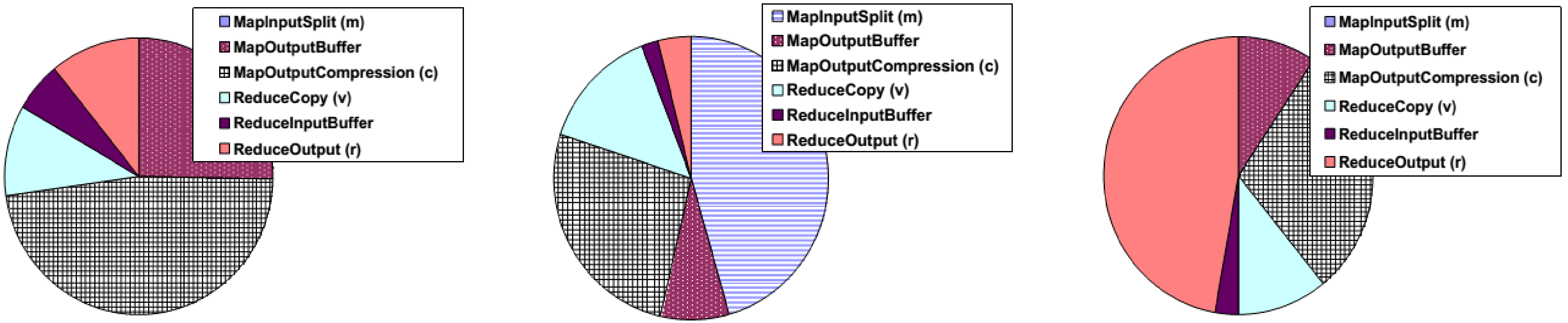


Hadoop-X

MRTuner

CPU and Network utilizations are higher.

Impact of Parameters on Selected Jobs



(a) TS-3 (b) NG-5 (c) PR-4

Compression is important

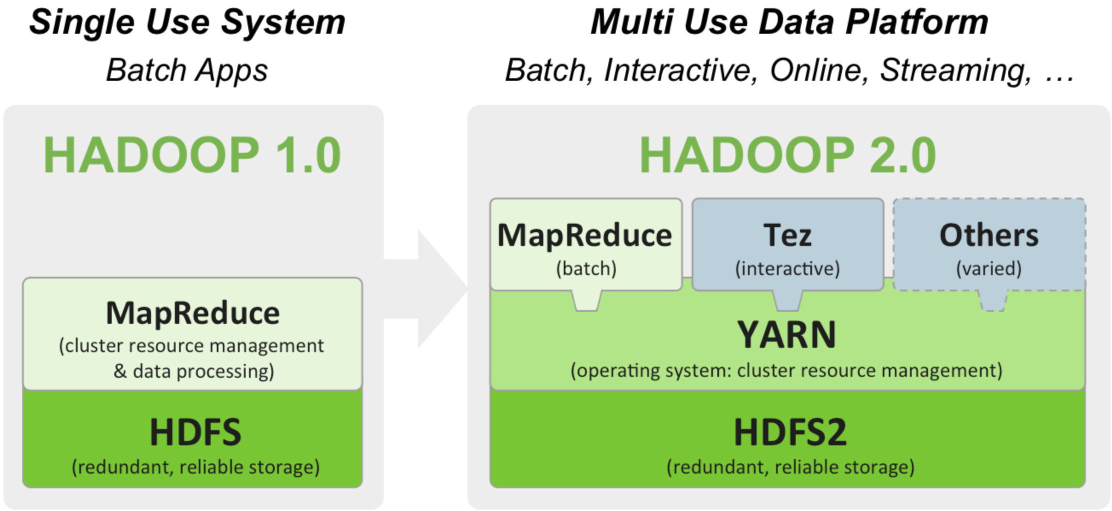
Map task # is important

Reduce task # is important

MRTuner tuned time: t1
Results after changing parameters to Hadoop-X setting: t2
The impact: (t2-t1)/t1, then normalize all the impacts

Ongoing research topics

- Efficient job optimization on YARN and Spark
- Tune for container size and executor size



Multi-model databases: Quantum framework and category theory

Multi-model databases

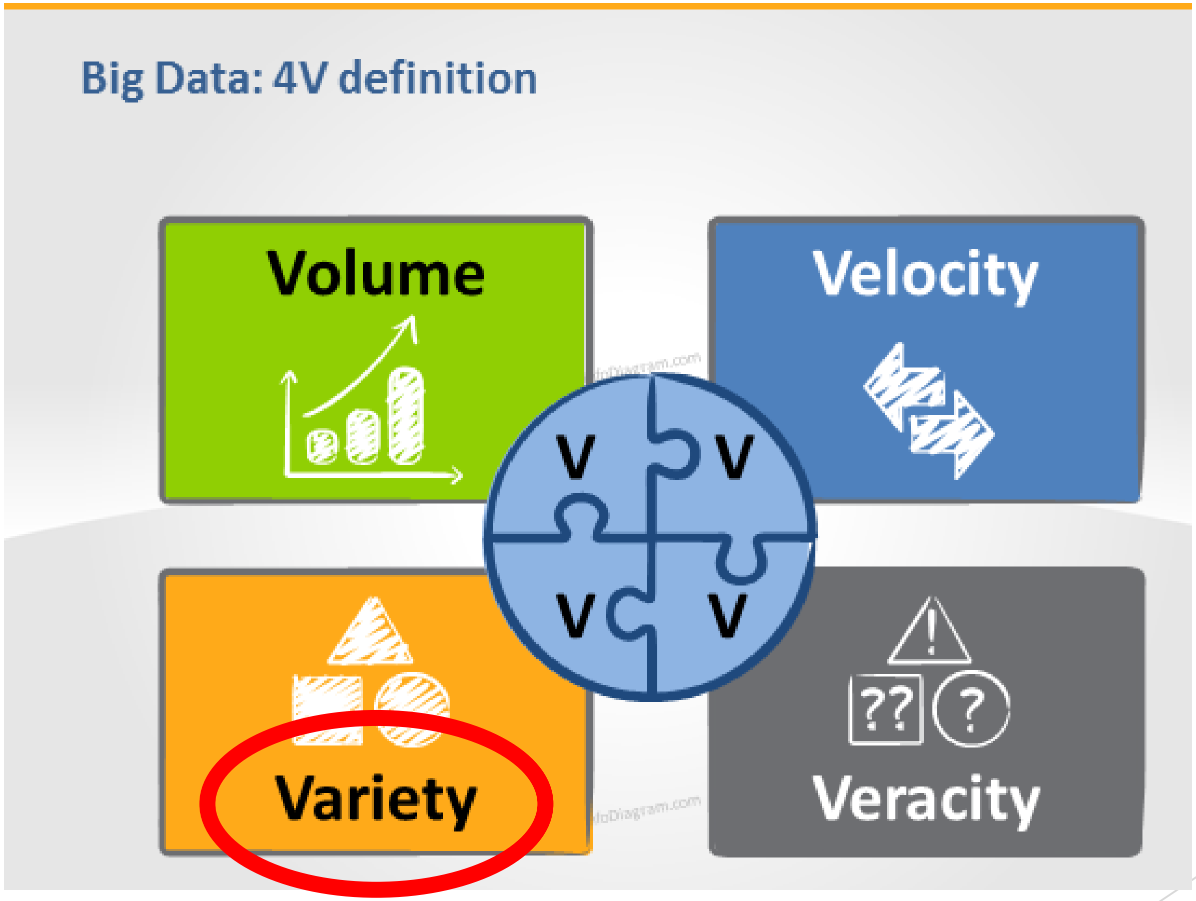
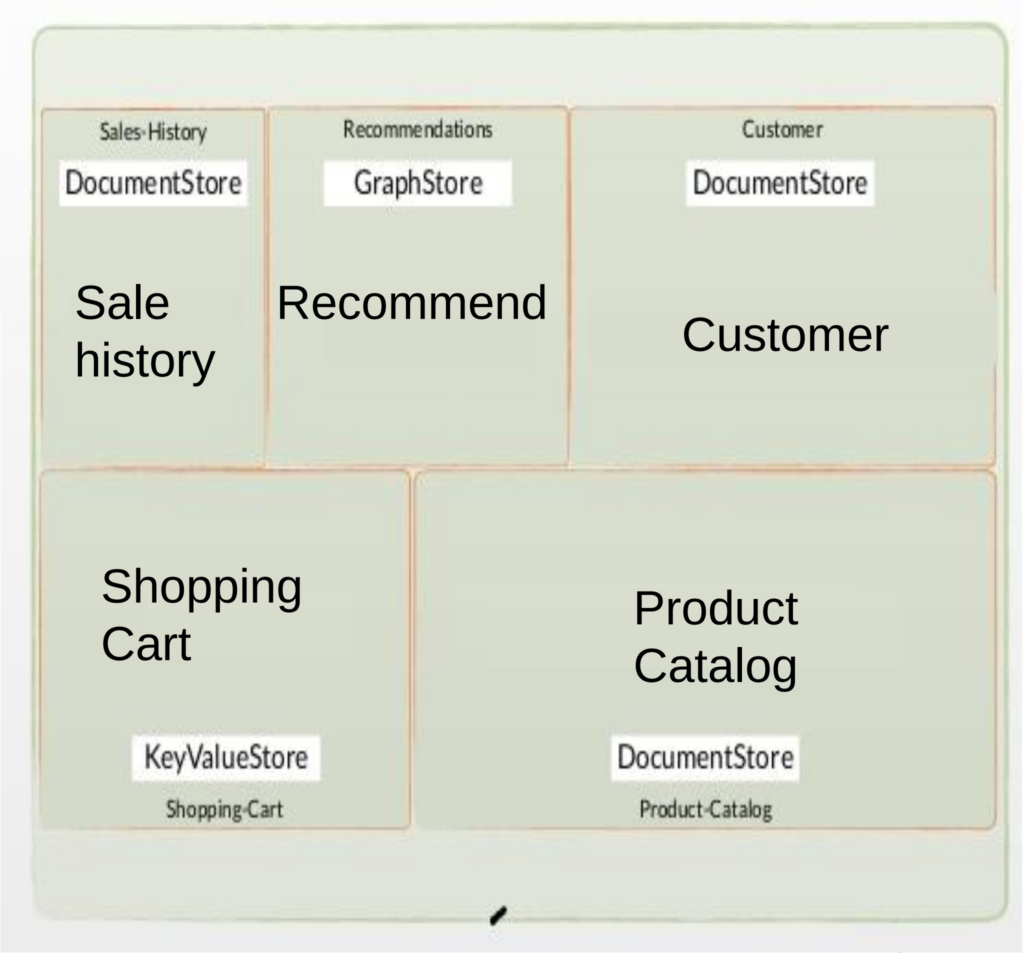


Photo downloaded from: https://blog.infodiagram.com/2014/04/visualizing-big-data-concepts-strong.html

Motivation: one application to include multi-model data

An E-commerce example with multi-model data



NoSQL database types

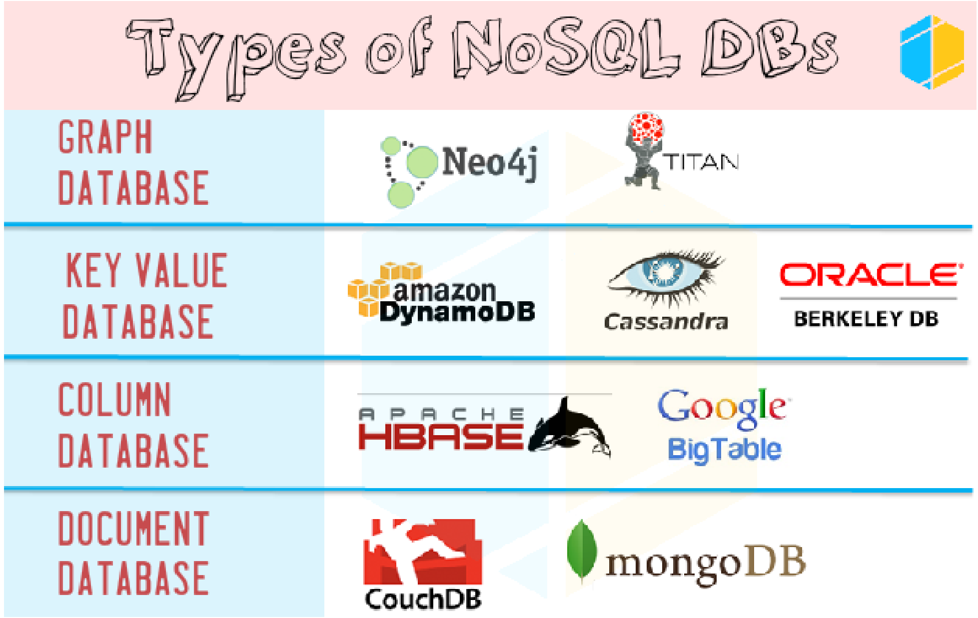
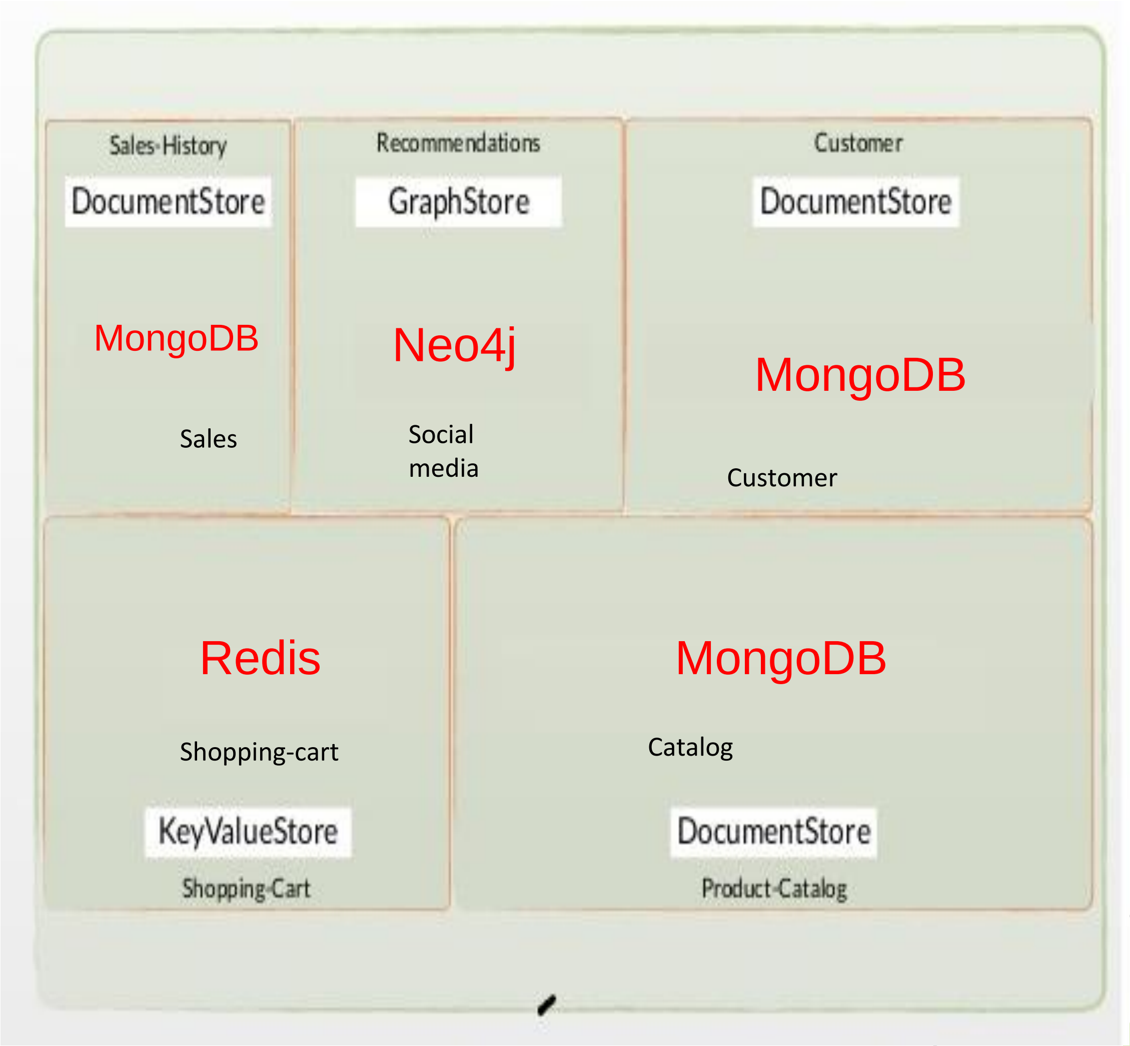


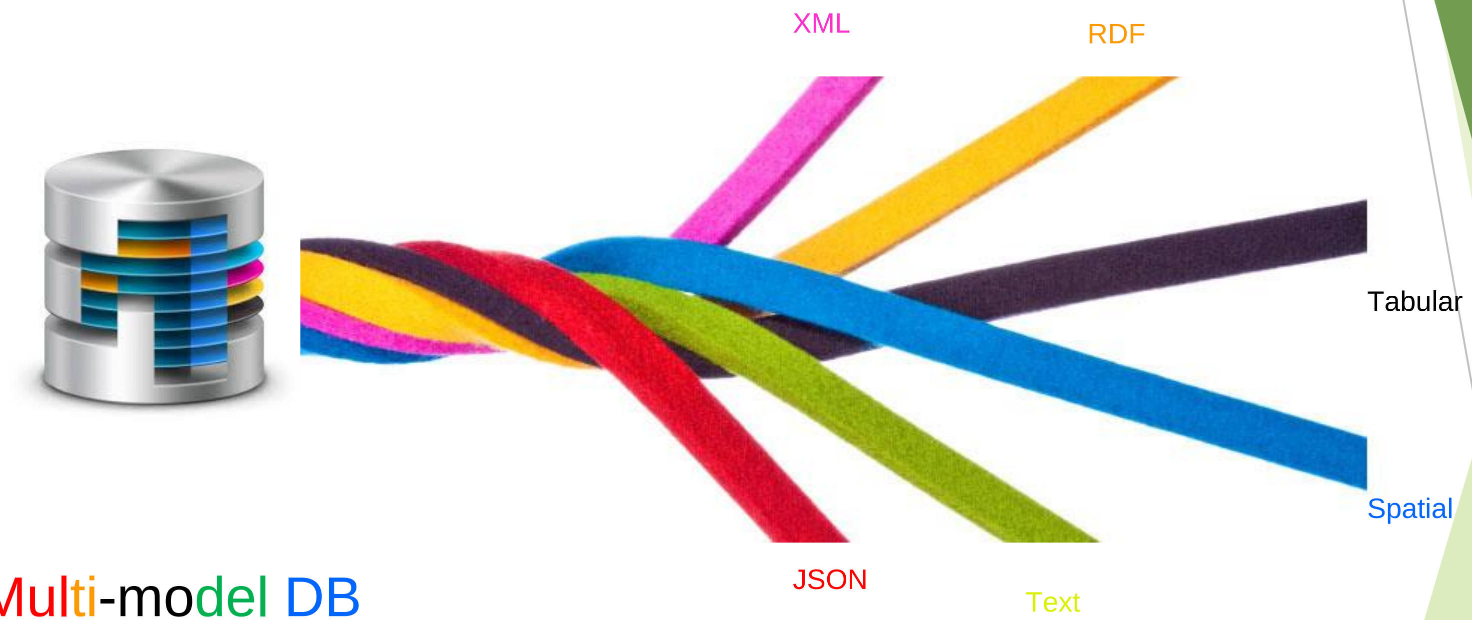
Photo downloaded from: http://www.vikramtakkar.com/2015/12/nosql-types-of-nosql-database-part-2.html

Multiple NoSQL databases



Multi-model DB

- One unified database for multi-model data



Multi-model DB

Challenge: a new theory foundation

Call for a unified model and theory for multi-model data!

The theory of relations (150 years old) is not adequate to mathematically describe modern (NoSQL) DBMS.

Two possible theory foundations

Quantum framework; Approximate query processing for open field in multi-model databases

Category theory: Exact query processing and schema mapping for close field in multi-model databases

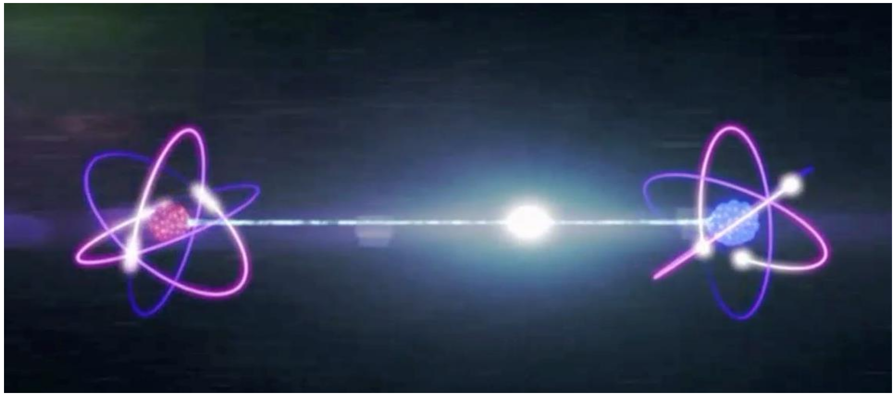
Quantum framework

Database is based on the components:
Logic (SQL expressions)
Algebra (relational algebra)'

The Quantum framework adds quantum probability and quantum algebra.

Why not classical probability

Apply three rules on multi-model data
Quantum superposition
Quantum entanglement
Quantum inference



Unifying multi-model data in Hilbert space

Relation XML

Hilbert space

RDF Graph

Use quantum probability to answer the query approximately

Two possible theory foundations

Quantum framework; Approximate query processing for open field in multi-model databases

Category theory: Exact query processing and schema mapping for close field in multi-model databases

What is category theory?

It was invented in the early 1940’s

Category theory has been proposed as a new foundation for mathematics (to replace set theory)

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, "g \circ f"'] & Y \arrow[d, "g"] \\ & Z \end{tikzcd}$$

A category has the ability to compose the arrows associatively

Unified data model

- One unified data model with objects and morphisms

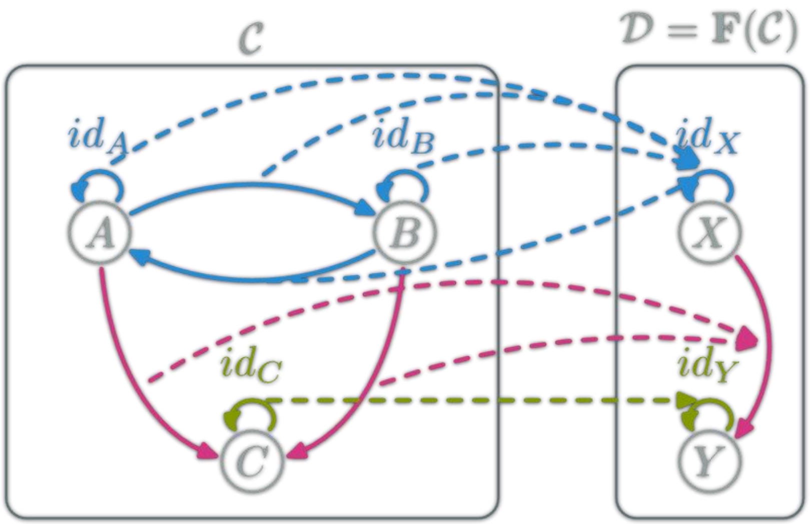
$$
\begin{tikzcd}
X \arrow[r, "f"] \arrow[dr, "g \circ f"'] & Y \arrow[d, "g"] \\
& Z
\end{tikzcd}
$$

Relation XML

RDF Graph

Unified language model

- One unified language model with functors



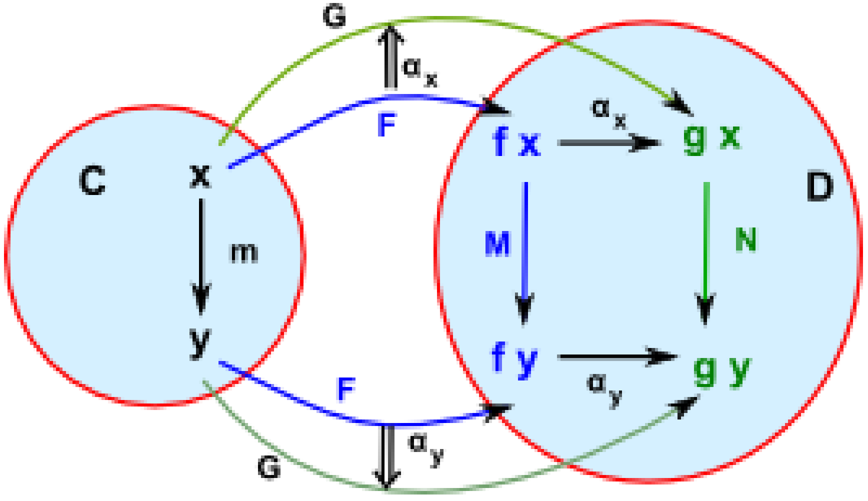
XPath
Xquery

SPAQL

SQL

Transformation

- Natrual transformation between multiple language for multi-model data



Ongoing research topics

- Approximate query processing based on quantum framework

- Multi-model data integration based on category theory

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, "g \circ f"'] & Y \arrow[d, "g"] \\ & Z \end{tikzcd}$$

Conclusion

1. Parameter tuning is important for Big data platform like Hadoop and Spark.

2. Emerging two new theoretical foundations on multi-model databases: quantum framework and category theory

Reference(1)

- Jiaheng Lu, Irena Holubová: Multi-model Data Management: What's New and What's Next? EDBT 2017: 602-605
- Chunbin Lin, Jiaheng Lu, Zhewei Wei, Jianguo Wang, Xiaokui Xiao: Optimal algorithms for selecting top-k combinations of attributes: theory and applications. VLDB J. 27(1): 27-52 (2018)
- Juwei Shi, Jia Zou, Jiaheng Lu, Zhao Cao, Shiqiang Li, Chen Wang: MRTuner: A Toolkit to Enable Holistic Optimization for MapReduce Jobs. PVLDB 7(13): 1319-1330 (2014)
- Jiaheng Lu, Chunbin Lin, Wei Wang, Chen Li, Haiyong Wang: String similarity measures and joins with synonyms. SIGMOD Conference 2013: 373-384

Reference(2)

- Yu Liu, Bolong Zheng, Xiaodong He, Zhewei Wei, Xiaokui Xiao, Kai Zheng, Jiaheng Lu: ProbeSim: Scalable Single-Source and Top-k SimRank Computations on Dynamic Graphs. PVLDB 11(1): 14-26 (2017)
- Pengfei Xu, Jiaheng Lu: Top-k String Auto-Completion with Synonyms. DASFAA (2) 2017: 202-218
- Tao Guo, Xin Cao, Gao Cong, Jiaheng Lu, Xuemin Lin: Distributed Algorithms on Exact Personalized PageRank. SIGMOD Conference 2017: 479-494

Reference(3)

- Yu Liu, Jiaheng Lu, Hua Yang, Xiaokui Xiao, Zhewei Wei: Towards Maximum Independent Sets on Massive Graphs. PVLDB 8(13): 2122-2133 (2015)
- Jiaheng Lu, Chunbin Lin, Wei Wang, Chen Li, Xiaokui Xiao: Boosting the Quality of Approximate String Matching by Synonyms. ACM Trans. Database Syst. 40(3): 15:1-15:42 (2015)
- Jinchuan Chen, Yueguo Chen, Xiaoyong Du, Cuiping Li, Jiaheng Lu, Suyun Zhao, Xuan Zhou: Big data challenge: a data management perspective. Frontiers Comput. Sci. 7(2): 157-164 (2013)


Thanks!


Q&A