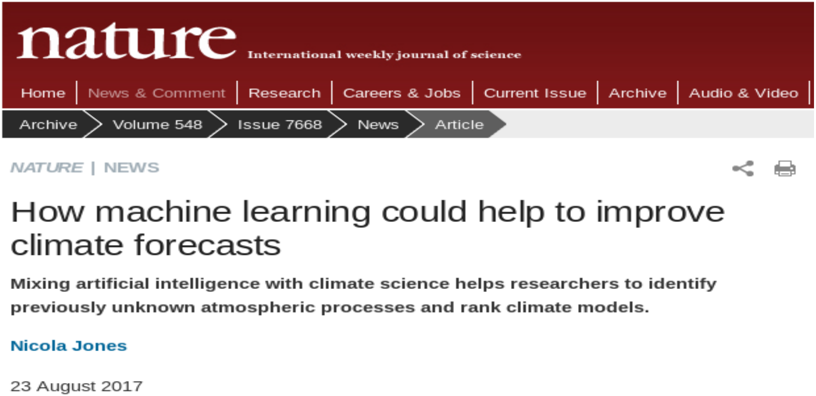
A Look at Data Management Systems for Climate Research

James J. Pan (jamesjpan@outlook.com)

Database Group @ Tsinghua Universty

September 2, 2020

Rise of Machine Learning for Spatial...



nature International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video |

Archive > Volume 548 > Issue 7668 > News > Article

NATURE | NEWS

How machine learning could help to improve climate forecasts

Mixing artificial intelligence with climate science helps researchers to identify previously unknown atmospheric processes and rank climate models.

Nicola Jones

23 August 2017

"If you go to major modeling centers and ask them how they work, the answer won't be machine learning," says Collins. "**But it will get there.**"[1]

[1] Nicola Jones. "How machine learning could help to improve climate forecasts". In: *Nature* 548.7668 (2017), pp. 379–380.

ACM Special Interest Group on Knowledge Discovery and Data Mining (**SIGKDD**)

KDD2017

APPLIED DATA SCIENCE INVITED TALKS

Professor Vipin Kumar

Professor
University of Minnesota
Big Data in Climate: Opportunities and Challenges for Machine Learning
Wednesday 10:00am – 12:00pm, Room 200D

More Information

"*We discuss the challenges involved in analyzing these* **massive data sets** *as well as opportunities they present for both advancing machine learning as well as the science of climate change...*"[2]

[2] Anuj Karpatne and Vipin Kumar. "Big Data in Climate: Opportunities and Challenges for Machine Learning". In: *KDD* (2017), pp. 21–22.

Very Large Databases (**VLDB**)

"In this tutorial, we present the recent work in the database community for handling **Big Spatial Data**...*"[3]

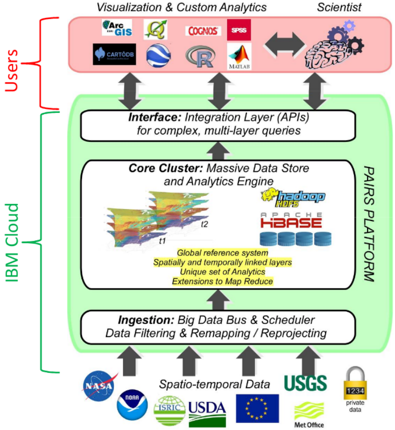
[3] Ahmed Eldawy and Mohamed Mokbel. "The Era of Big Spatial Data". In: *VLDB* (2017).

Why is everyone so excited?

Meteorology in the eyes of data scientists:

- Terabytes of Earth imagery data get generated *per day*
  - Static analysis "macroscale" (*e.g.* deforestation, land-use, urban expansion, ...)
  - Real-time analysis "microscale" (*e.g.* flood monitoring, wildfires, landslides, tornados, ...)
  - Somewhere in between "mesoscale" (*e.g.* soil moisture, crop production, water availability, ...)
- Discovery and Monitoring: mine data to discover new climate relationships (*e.g.* teleconnections, tripoles), monitor real-time interactions (*e.g.* wildfires, floods, effects on rainfall...)
- Model Improvement: refine climate models to improve predictive and reproductive power
- **Data Management**: ingest, store, query, ... to efficiently support data applications

Example: IBM Physical Analytics Integrated Repository and Services (PAIRS)[4]



- E.g. "Give me two years of temperature in Kyoto, plus 2-week forecast"

[4] Siyuan Lu. "IBM PAIRS - A Big Physical Data Service to Accelerate Analytics and Discovery". In: (2017).

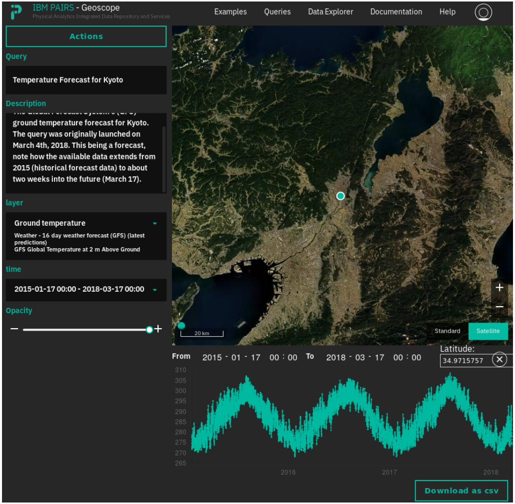


Figure: The IBM Pairs web interface.

- Good for simple retrievals; limited processing capability
- No real-time monitoring
- Limited modeling capability (limited forecasting ability)

Example: ESRI ArcGIS[5]

Policymaker: "How is land use changing in my district?"

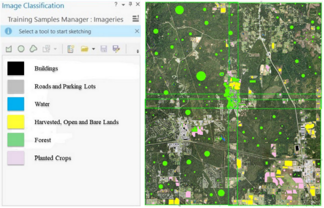


Figure 4. Training Samples Manager and Sampled Labels

Figure: Train a convolution neural network based on expert labels, then feed new images per pixel and collect the predicted land use.

Figure 1. Raw NAIP Image versus Classified Image

Class	Precision	Recall
Buildings	82.50	81.28
Roads	84.78	85.13
Waters	86.14	85.55
Harvested	90.38	91.88
Planted	89.05	88.19
Forest	91.46	92.65

Table 1. Accuracy Assessment of U-Net Model (Precision and Recall in %)

[5] Amin Tayyebi. "High-Resolution Land Cover Mapping using Deep

Background: Filesystems and Data Management

Why use data management software (databases) anyway? Isn't the filesystem good enough?

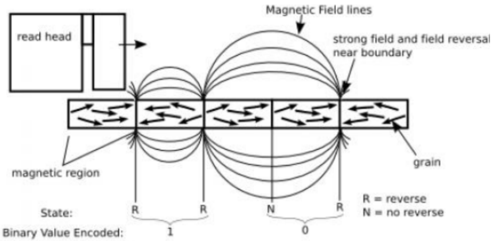


Figure: Magnetic hard drives store bits (0's and 1's) inside magnetic fields. A chunk of data (typically 512 bytes) is called a *sector* and is the minimum size the drive can write.

- Reading a particular "file" involves spinning the disk to visit sectors containing the file contents.
- Sometimes sectors belonging to one file are not physically near each other on the disk, known as "fragmentation".

Comparison of Read Latencies

- **Magnetic disk** at 15,000 RPM: up to 4 milliseconds (4,000,000 ns)
- **Solid-state drive** (NAND): 50,000–100,000 ns
- **RAM**: about 60 ns
- CPU L2 cache: about 10 ns
- CPU access: about 1 ns

Reading from disk is about 10^6 times slower than reading from RAM. In other words, relative difference between **1 second** and **11 days**.

Filesystems and Data Management

Both filesystems and databases deal with storage and I/O:

- Filesystems: ext4 (Unix), NTFS (Windows), APFS (Apple)
- Fairly uniform use case: fast file access, security, data integrity, compression, paging, volume resizing, locking
- E.g. journaling systems such as NTFS keep a master log file to support data restore in case of crashes

- Databases: Oracle, SQL Server, Postgres, Teradata, Vertica, MongoDB, MonetDB, kdb+ ...
- **Data contents are related**, e.g. "find total revenue of the Rome store over all Mondays of last year"
- **"No one size fits all"**: different approaches for different use cases

No One Size Fits All: Row-Store vs. Column-Store

Consider the I/O cost of reading/writing this table:

Item	*Category*	*Revenue*
Glove	*Sport*	500
Cap	*Sport*	200
Chair	*Housing*	450
Table	*Housing*	100
Shoe	*Sport*	600

Row-store (e.g. H-Store[6]): {`Glove, Sport, 500, Cap, Sport, 200, Chair, Housing, ...`}

Column-store (e.g. MonetDB[7]): {`Glove, Cap, Chair, ..., Sport, Sport, Housing, ..., 500, 200, 450, ...`}

[6] Robert Kallman et al. "H-store: a high-performance, distributed main memory transaction processing system". In: *VLDB* 1.2 (2008), pp. 1496–1499.

[7] Stratos Idreos et al. "MonetDB: Two Decades of Research in Column-oriented Database Architectures". In: *IEEE Data Eng. Bull.* 35.1 (2012), pp. 40–45.

Management Systems for Spatial Data: Array Databases

Consider the cost of the blue subarray query:

Relational Database

i	*j*	*value*
0	0	32.5
1	0	90.9
2	0	42.1
3	0	96.7
0	1	46.3
1	1	35.4
2	1	35.7
3	1	41.3
0	2	81.7
1	2	35.9
2	2	35.3
3	2	89.9
0	3	53.6
1	3	86.3
2	3	45.9
3	3	27.6

Array Database

32.5	46.3	81.7	53.6
90.9	35.4	35.9	86.3
42.1	35.7	35.3	45.9
96.7	41.3	89.9	27.6

SciDB[8]

Traditional RDBMS:

- Basic types (integer, float, string, ...)
- Relational operators (select, join, group by, ...)
- Good for queries such as "What are the phone numbers for all employees in Delaware?"

ID	*Name*	*Location*	*PhoneNumber*
1	*JohnSmith*	*Delaware*	123 – 456 – 7890
2	*JaneDoe*	*Delaware*	123 – 456 – 7891
...	...	...	...

Array-Based DBMS:

- **Array** data model
- Array operators (structural operators, content operators)
- Good for multidimensional queries such as point time-range query, spatial aggregation query

2	5	4
2	1	8
...	...	...

[8] The SciDB Development Team. "Overview of SciDB". In: *SIGMOD* (2010).

SciDB: Chunk Storage

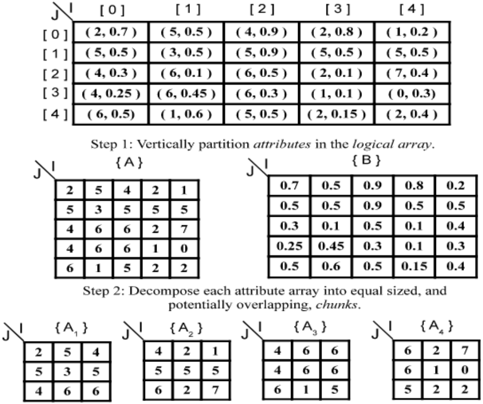


Figure 5: SciDB Storage Manager

SciDB

- Designed to support spatial operators (e.g. Gaussian smoothing)
- Chunk partitioning motivated by typical access patterns

Benchmarks[9]

MySQL compared with **SciDB**; Cluster size 10 nodes, 2GB Ram + 3.2 GHz CPU per node

DBMS	Dataset	Loading/Cooking [min]				Query Runtimes [min]									
		Load	Obsv	Group	Total	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Total
MySQL	*small*	760	110	2	872	123	21	393	0.4	0.36	0.6	0.6	49	50	638
	normal	770	200	90	1060	54	44	161	50	32	51	52	395	395	1234
	(scaleup)	(1.0)	(1.8)	(45)	(1.2)	(0.4)	(2.1)	(0.4)	(125)	(89)	(85)	(87)	(8.1)	(7.9)	(1.93)
SciDB	*small*	34	1.6	0.6	36	8.2	0.2	3.7	0.007	0.01	0.01	0.01	1.8	1.9	16
	normal	67	1.9	15	84	3.6	0.07	1.7	0.015	0.017	0.02	0.11	2.2	2.3	10
	(scaleup)	(2.0)	(1.2)	(25)	(2.3)	(0.4)	(0.4)	(0.4)	(2.1)	(1.7)	(2)	(11)	(1.2)	(1.2)	(0.63)
(MySQL /SciDB)	*small*	(22)	(69)	(3.3)	(24)	(15)	(105)	(106)	(57)	(36)	(60)	(60)	(27)	(26)	(40)
	normal	(12)	(105)	(6)	(13)	(15)	(630)	(95)	(3330)	(1880)	(2550)	(470)	(180)	(170)	(120)

TABLE I

BENCHMARK RESULTS. (*num*) IS A RATIO OF RUNTIMES, EITHER *normal* VS. *small* (SCALEUP) OR MYSQL VS. SCIDB.

- "small": single-machine, 160 3750x3750 images, 99 GB
- "normal": distributed, 400 7500x7500 images, 990 GB

Takeaway: SciDB achieves 2-orders speedup compared to MySQL across the 9 benchmark queries on the cluster

[9] Philippe Cudre-Mauroux et al. "SS-DB: a standard science DBMS benchmark". In: *XLDB* (2012).

Application: EarthDB (MODIS)[10]

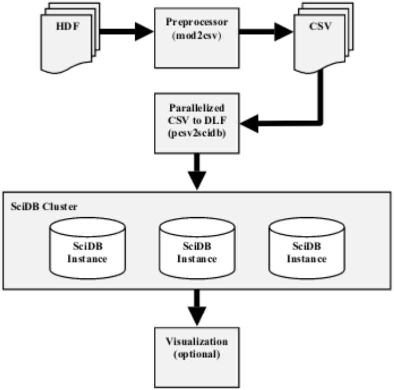


Figure 3: EarthDB High-Level Data Flow

- Extends SciDB to support MODIS data

[10] Gary Planthaber, Michael Stonebraker, and James Frew. "EarthDB: Scalable Analysis of MODIS Data using SciDB". In: *SIGSPATIAL* (2012), pp. 11–19.

Application: AscotDB (Astronomy)[11]

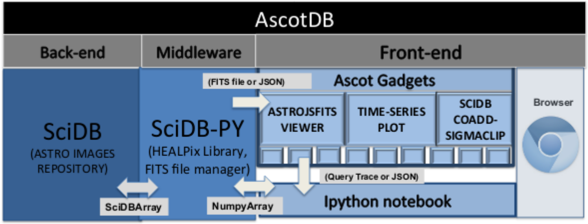


Figure 1: AscotDB architecture: SciDB as back-end, python middleware, Ascot and IPython as front-ends.

- Adds spherical support to SciDB's Cartesian operators
- Adds efficient spherical operators
- Adds Python bindings and graphical interface
- Interdisciplinary collaboration between astronomers and database experts

[11] Jacob Vanderplas et al. "Squeezing a Big Orange into Little Boxes: The AscotDB System for Parallel Processing of Data on a Sphere". In: (2015).

TileDB[12]

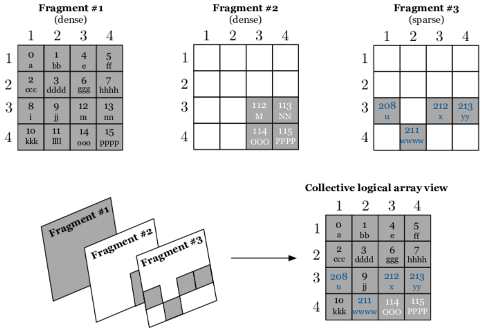


Figure 4: Fragment examples

- Better support for sparse arrays
- Avoids fixed-dimension chunking and full chunk reads
- Key idea: convert random-access writes into sequential appends by storing update **fragments**

[12] Stavros Papadopoulos et al. "The TileDB Array Data Storage Manager". In: *VLDB* (2016).

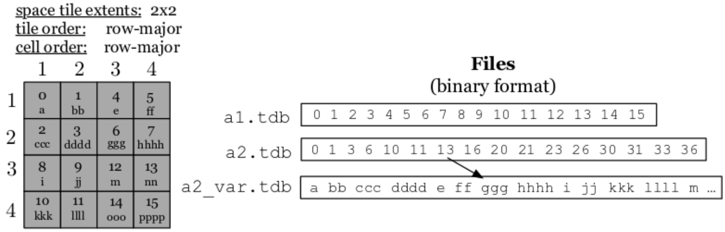


Figure 6: Physical organization of dense fragments

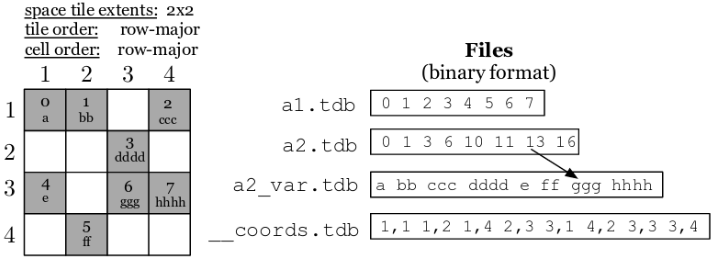
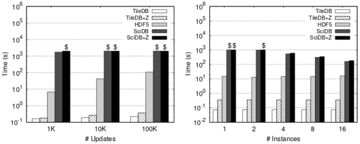
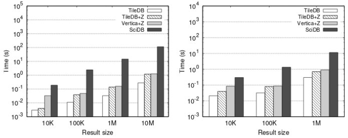


Figure 7: Physical organization of sparse fragments

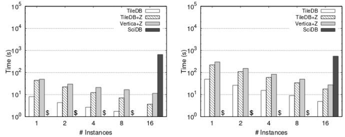


(a) vs. # updates (HDD) (b) vs. # instances (SSD)

Figure 10: Random update performance of dense arrays



(a) DQ vs. result size (HDD) (b) SQ vs # result size (HDD)



(c) DQ vs. # instances (SSD) (d) SQ vs. # instances (SSD)

Figure 14: Subarray performance for sparse arrays

- Random-access update on dense arrays is 2-order faster than HDF5, 3-order faster than SciDB
- Subarray on sparse arrays is 2-order faster than SciDB
- Dense read comparable to HDF5

Array DB vs HDF5

Array DB

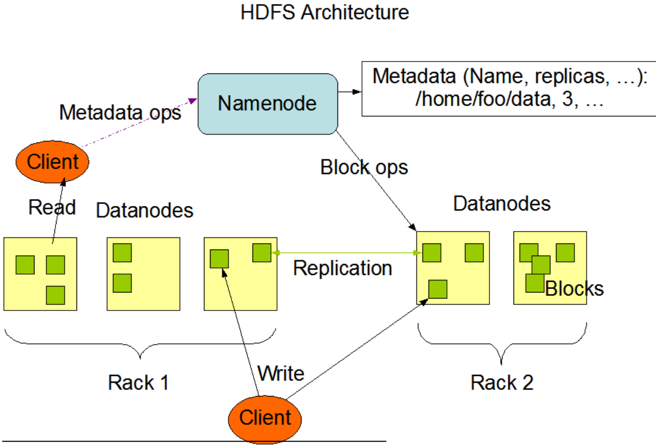
- Is a database
- Can serve as warehouse of files from various sources
- Supports native operators, e.g. join, query
- Different flavors support different use cases, e.g. dense/sparse arrays, writes/reads, ...
- Easy scale-out on clusters
- Suitable for frequent, varied access

HDF5

- Is a file format and library
- Typically stores one "data product" per file
- Operators are typically at the application level, e.g. a Python program to join two datasets
- Not meant to be distributed in a cluster
- Suitable for one-shot processing

GeoTrellis (http://geotrellis.io)

- In-memory big raster analysis on top of Apache Spark
- Data is managed by Hadoop Filesystem[13] (HDFS) for fault-tolerance and to support map-reduce style computation
- "Moving computation is cheaper than moving data"



[13] "HDFS Architecture Guide". In: ().

GeoTrellis: NDVI Example

$NDVI = (NIR - Red)/(NIR + Red)$ In GeoTrellis:

```
rdd.mapValues { (red, nir) => (nir - red) /
(nir + red) } .reduceByKey(_.localMax(_))
```

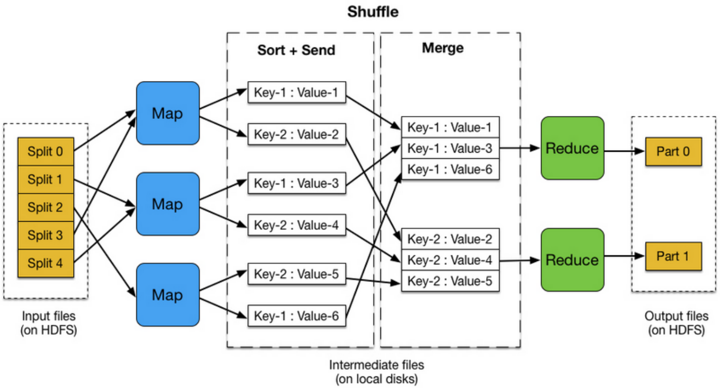
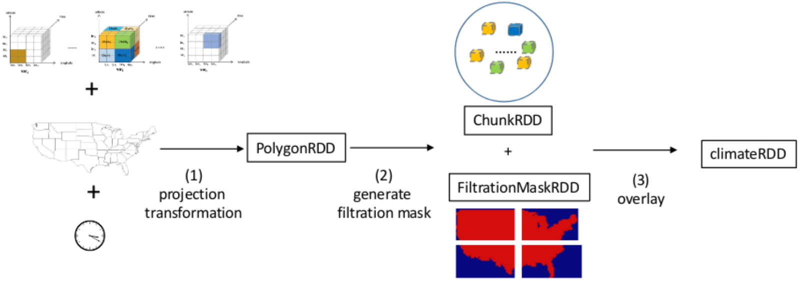


Figure: From http://www.sunlab.org/teaching/cse6250/spring2017/lab/image/post/mapreduce-flow.jpg

ClimateSpark[14]

- Adds spatiotemporal index to improve data access
- HDF, netCDF files stored inside HDFS, then indexed, chunked, and finally converted to in-memory RDD's
- RDDs are processed in Spark via map reduce, same as GeoTrellis



[14] Fei Hu et al. "ClimateSpark: An in-memory distributed computing framework for big climate data and analytics". In: *Computers and Geosciences* (2018).

ClimateSpark

- Experiment: 20 nodes, 24 CPU@2.35 GHz + 24GB RAM per node; $1/2^o * 5/8^o$ image resolution, roughly 9 TB MERRA2 data spanning 16 years

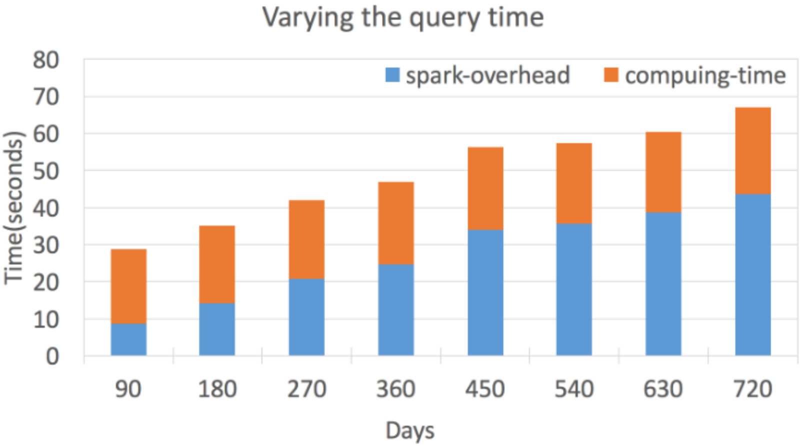


Figure 1. Run time for querying the data in Virginia and computing the monthly mean when varying the query time

Scalable Machine Learning: MLlib

Many ML operations are a good fit for map reduce. MLLib[15] adds machine learning operators onto Apache Spark. Example, gradient descent:

$$w \leftarrow w - \alpha \cdot \sum_i g(w; x_i, y_i)$$

```
for (i <- 1 to n) {
  val gradient =
    points.map { p =>
      (1/(1 + exp(-p.y*w.dot(p.x)) - 1)*p.y*p.x
    ).reduce(_ + _)
  w -= alpha*gradient
}
```

[15] Xiangrui Meng. "MLlib: Scalable Machine Learning on Spark". In: ().

Scalable Machine Learning: MLlib

MLlib is a standard Spark component, thus is tightly integrated with Spark data operations such as Spark SQL:

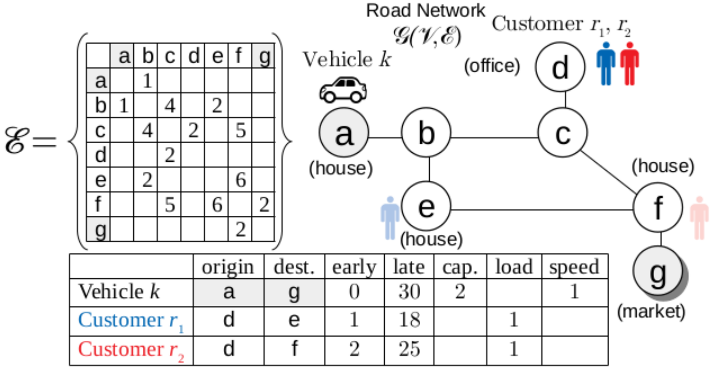
```
val trainingTable = sql("""SELECT... FROM... JOIN...""")
val training = traingTable.map { row => ... }
val model = SVMWithSGD.train(training)
```

Integrating machine learning with existing data management tools is an exciting area of research[16].

[16] Arun Kumar, Matthias Boehm, and Jun Yang. "Data Management in Machine Learning: Challenges, Techniques, and Systems". In: *SIGMOD* (2017).

Ridesharing: A different kind of spatial problem

17



$\mathcal{E} =$

	a	b	c	d	e	f	g
a		1					
b	1		4		2		
c		4		2		5	
d			2				
e		2				6	
f			5		6		2
g						2	

	origin	dest.	early	late	cap.	load	speed
Vehicle k	a	g	0	30	2		1
Customer r_1	d	e	1	18		1	
Customer r_2	d	f	2	25		1	

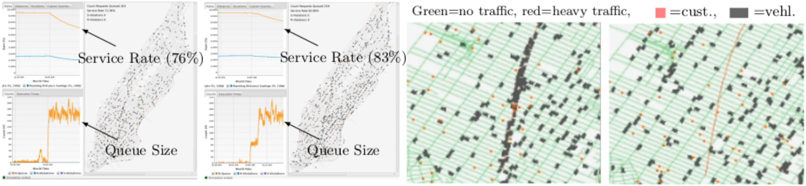
Figure 1: ***Ridesharing example.***

Figure: How to optimally match vehicles to customers and design the service route?

[17] James Pan, Guoliang Li, and Juntao Hu. "Ridesharing: Benchmark, Simulator, and Evaluation". In: *VLDB* (2019).

Ridesharing

[18]



(a) GRINS *No Traffic*
(b) GRINS-F *No Traffic*
(c) GRINS-F *Broadway Traffic*
(d) GRINS-FB *Broadway Traffic*

Figure 4: Jargo usage scenario.

Figure: How to evaluate ridesharing algorithms? Jargo can achieve real-time simulation of 1,000's vehicles on commodity machine by manipulating data of the system instead of physical simulation.

[18] James Pan, Guoliang Li, and Yong Wang. "Evaluating Ridesharing Algorithms using the Jargo Stochastic Simulator". In: *VLDB* (2020).

Summary

Key takeaways:

- New knowledge hiding in vast amounts of data
- Data management helps work around physical limitations of storage
- Array DB (SciDB, EarthDB, TileDB) for new science applications
- Integration of ML with data management, e.g. Spark MLlib
- **Very active space**

Thank you!

Q&A