Machine Learning Systems

(294-162)

Amir Gholami & Joseph E. Gonzalez

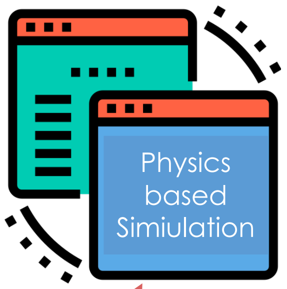
About Amir

PostDoc at RiseLab and BAIR

PhD: UT Austin class of 2017

Back in 2011 – Golden Age of Large Scale Computing

Training Data



→ **ML Model**

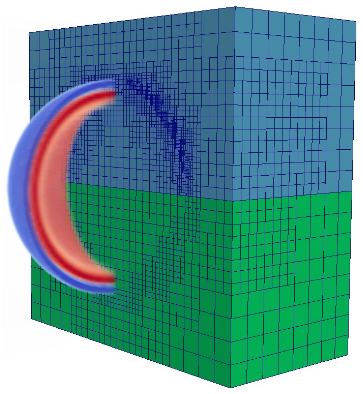
Very Expensive

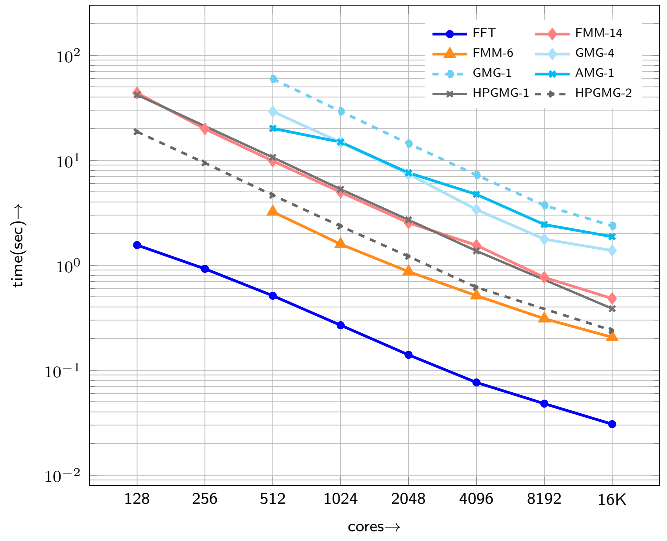
Stampede 1 System at TACC: 6400 Intel Xeon nodes, 10 PFLOPS, $50M (equivalent to 4 DGX-A100 systems today)

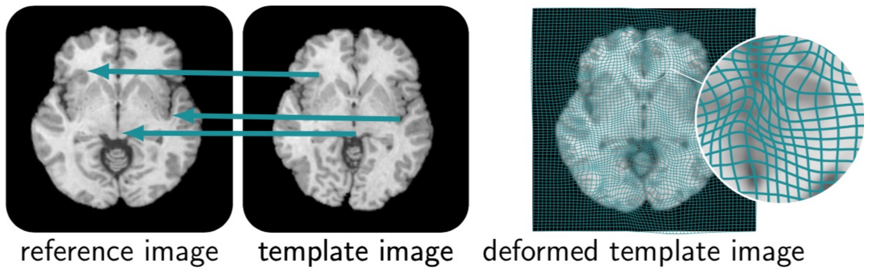
My story …

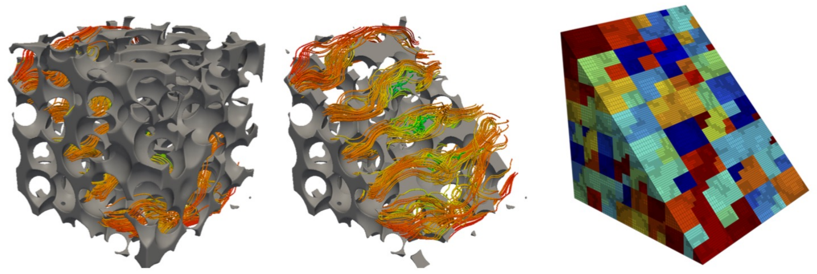
High Performance Computing $\longrightarrow$ Learning Systems

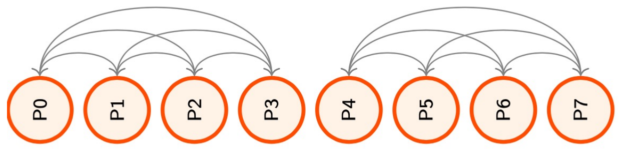
Use Supercomputers to Efficiently Solve Large Scale Problems

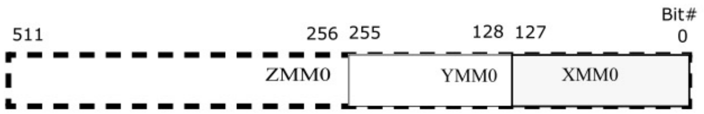






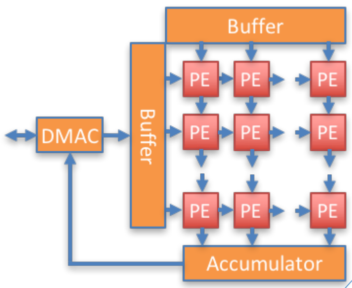


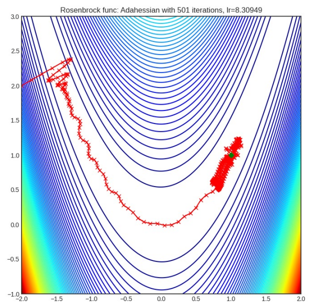


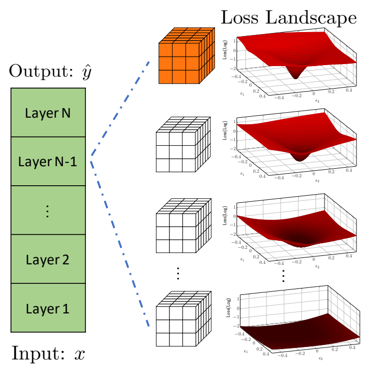


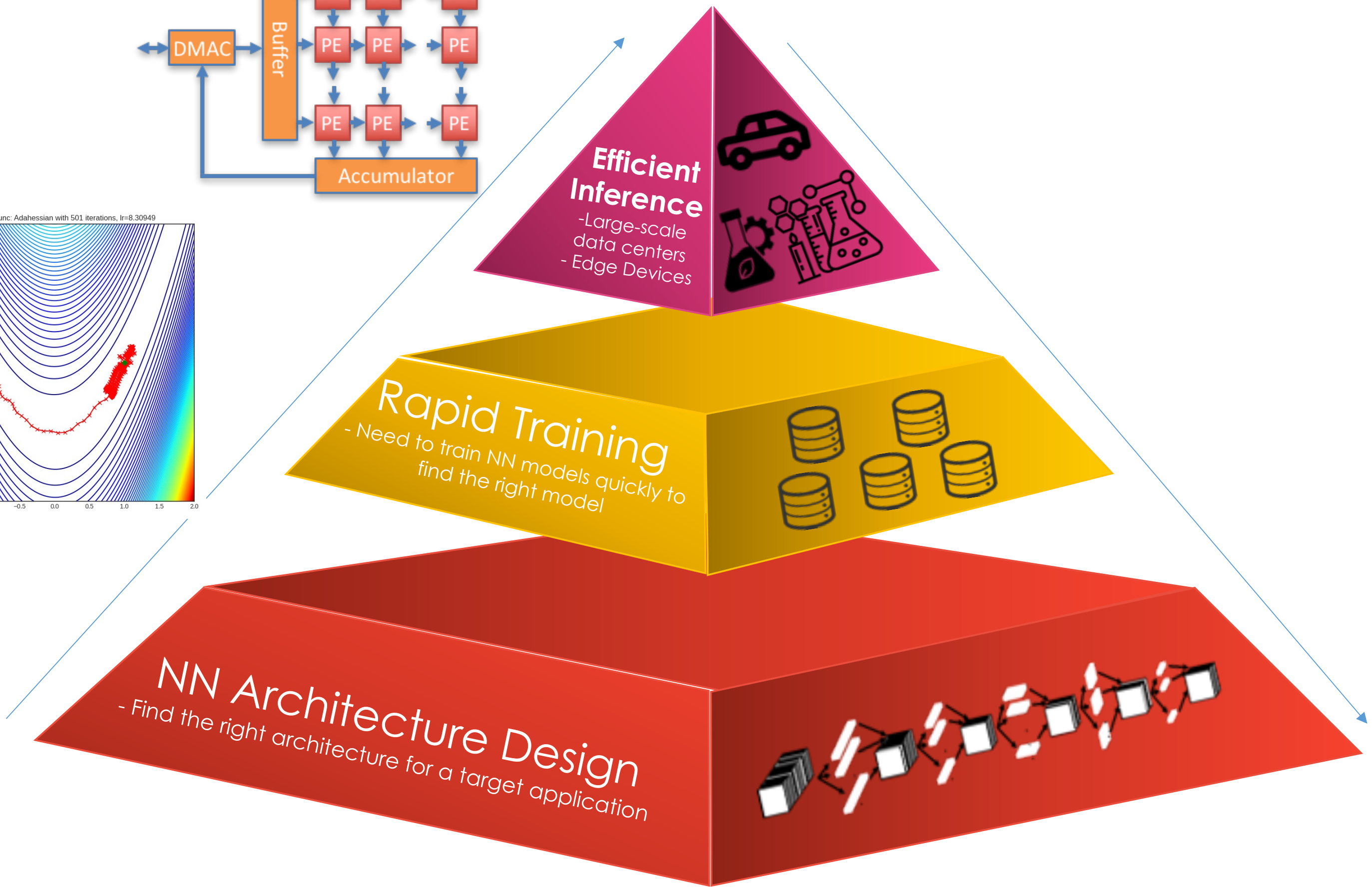
And this entailed a lot of low level compute and communication optimizations

PostDoc Focus: Full Stack Deep Learning








Efficient Inference
-Large-scale data centers
- Edge Devices
Rapid Training
- Need to train NN models quickly to find the right model
NN Architecture Design
- Find the right architecture for a target application

About Joey

About Me

- Currently on leave for **new startup** (http://spiralai.co)
  - Also expecting **second child** in a few weeks…
- Co-director of the RISE Lab
- Co-founder of Turi Inc.

Research

- Machine Learning & Distributed Systems
- Computer Vision, NLP, Secure Learning
- Autonomous Driving, Robotics, Chemistry?
- …

riselab UC Berkeley

BAIR BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

BERKELEY DeepDrive

APACHE Spark™

turi create intelligence™

My story …

Machine Learning $\rightarrow$ Learning Systems

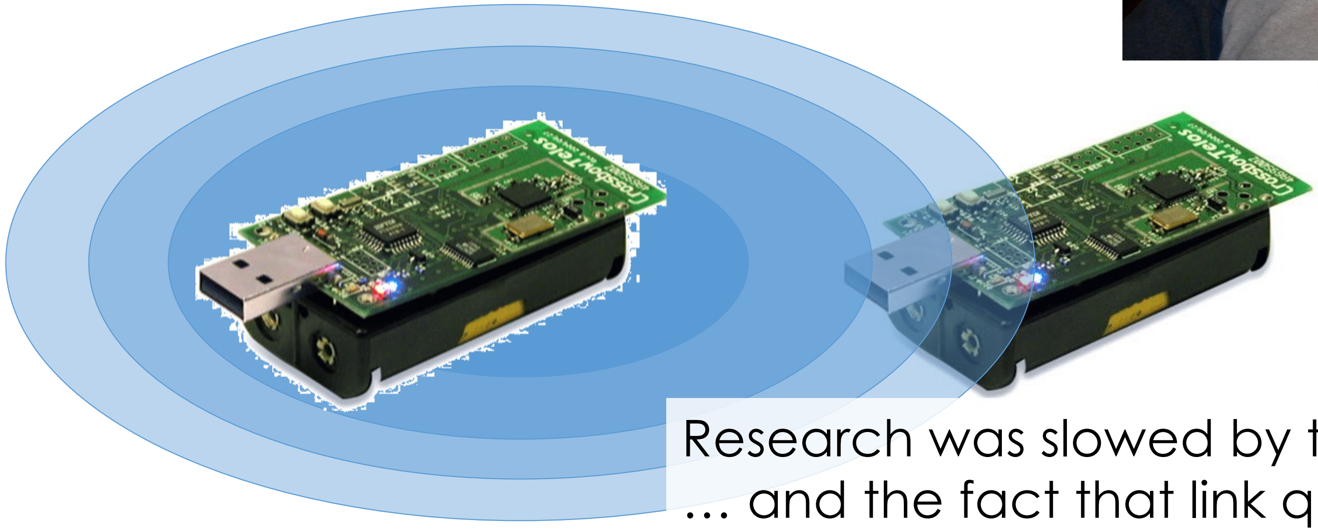
Back in 2007

AT&T
9:42 AM
Text
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
Clock
Calculator
Notes
Settings
Phone
Mail
Safari
iPod

FORECLOSURE
Sale

Back in 2007

I started studying the use of Gaussian Process (GP) models for wireless link quality estimation

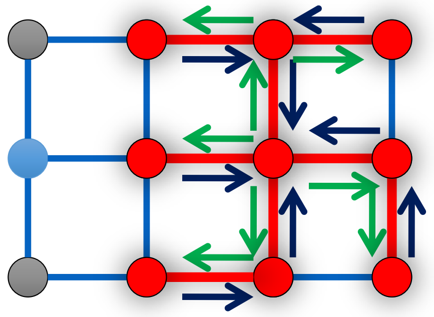


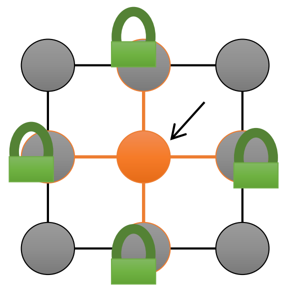
Research was slowed by the speed of training

… and the fact that link quality is difficult to predict

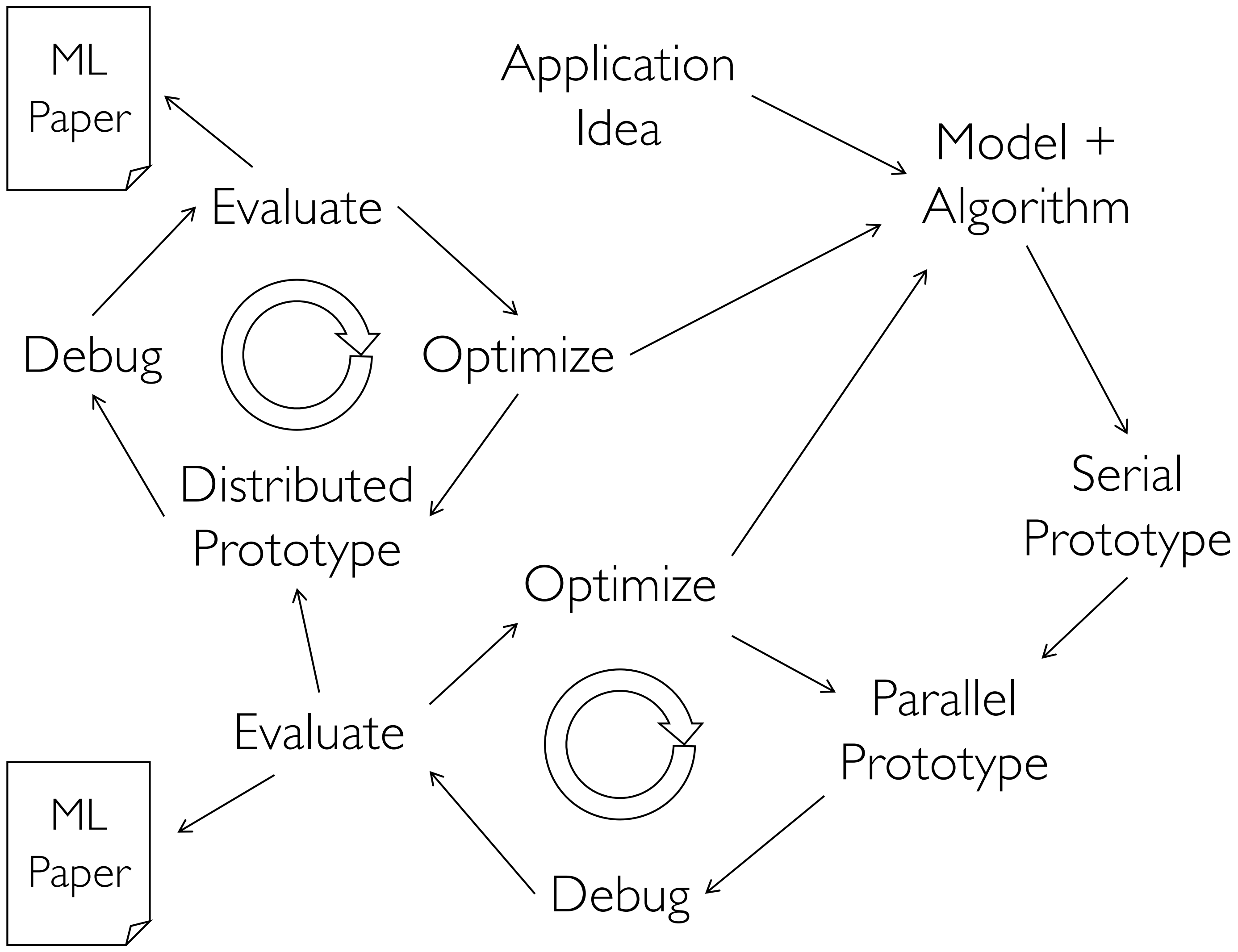
Back in 2008

I started studying parallel inference algorithms and systems





I designed and implemented parallel learning algorithms on top of low-level primitives …


ML Paper
Application Idea
Model + Algorithm
Evaluate
Debug
Optimize
Distributed Prototype
Serial Prototype
Optimize
Parallel Prototype
Evaluate
ML Paper
Debug

Low-Level Primitives

- Shared Memory Parallelism: *pThreads* & *OpenMP*
- Distributed Communication: *Actors* & *MPI*
  - Yes,… I built an RPC framework and then an actor framework
- Hardware APIs + Languages: *GPU Programming (CUDA)*

Low-level parallel primitives offer
fine grained control over parallel execution.

Advantages of the Low-Level Approach

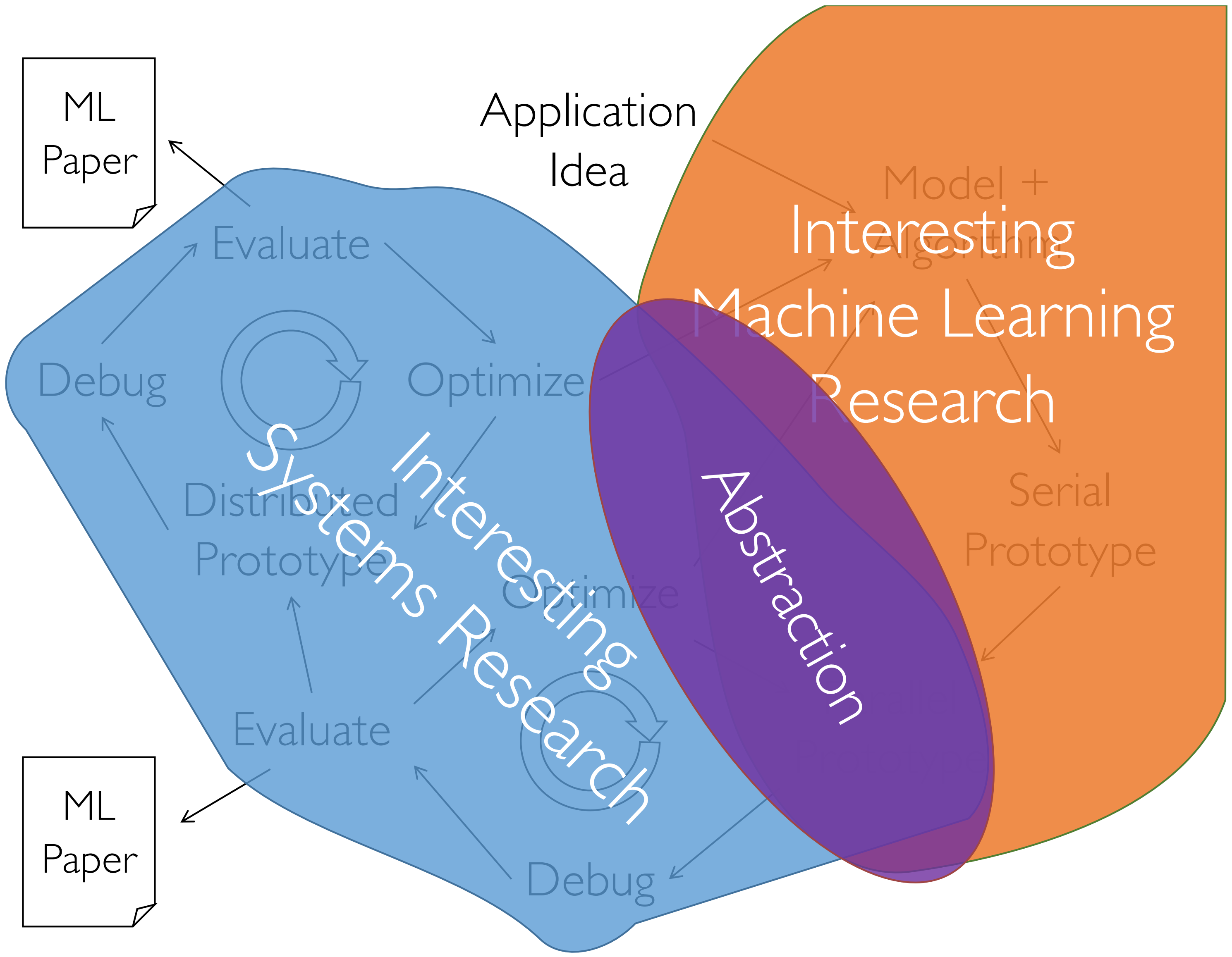
- ➢ Extract maximum performance from hardware
- ➢ Enable exploration of more complex algorithms
  - Fine grained locking
  - Atomic data-structures
  - Distributed coordination protocols

My implementation is better than your implementation.

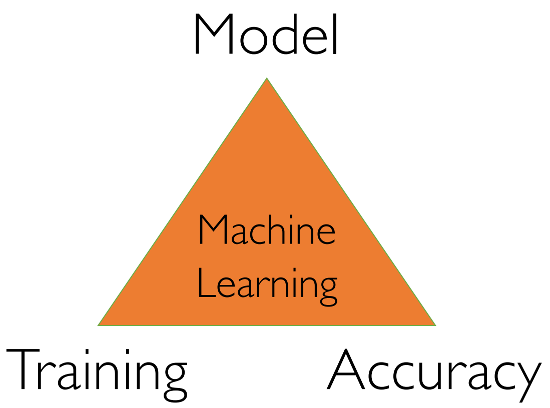
Limitations of the Low-Level Approach

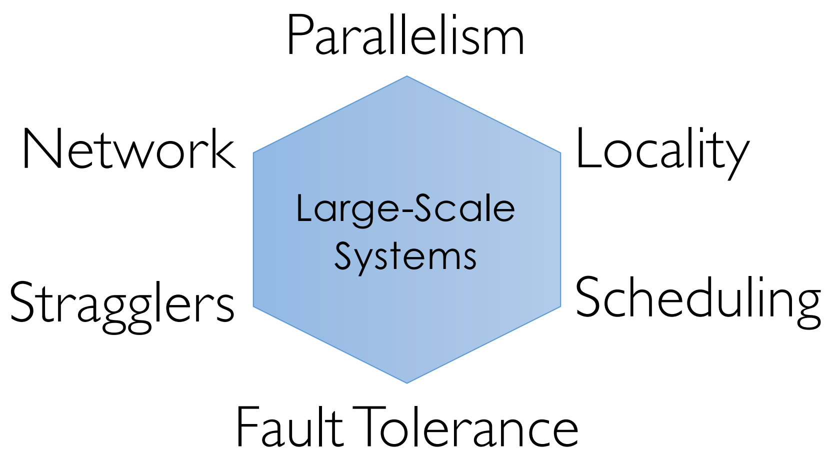
- Repeatedly address the same system challenges
- Algorithm conflates learning and system logic
- Difficult to debug and extend
- Typically does not address issues at scale: hardware failure, stragglers, …

Months of tuning and engineering
for one problem.

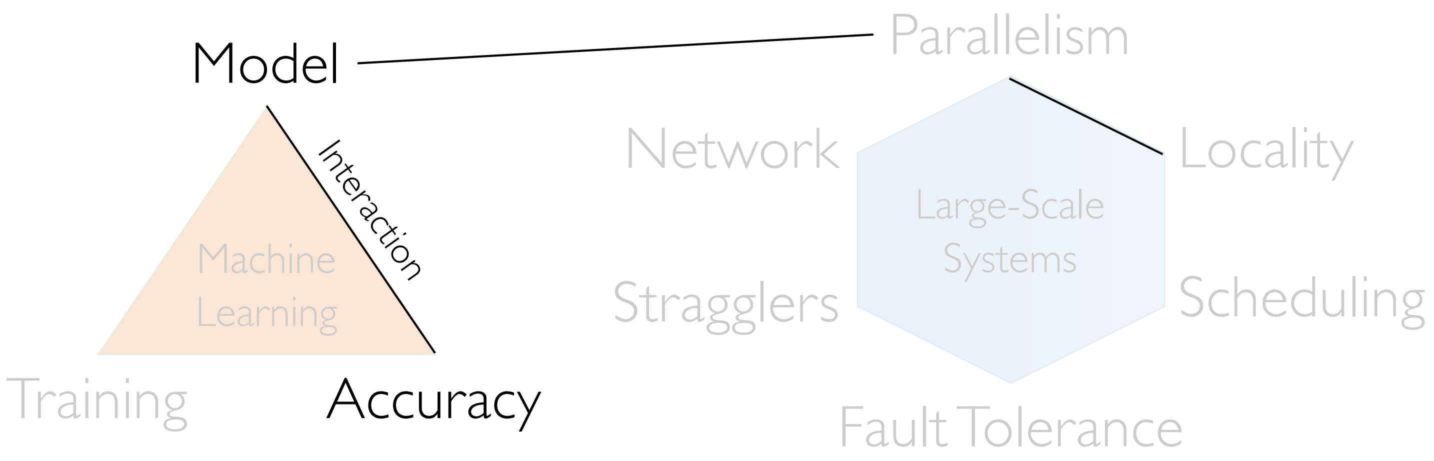

ML Paper
Application Idea
Model + Algorithm
Interesting Machine Learning Research
Evaluate
Debug
Optimize
Distributed Prototype
Interesting Systems Research
Abstraction
Serial Prototype
Optimize
Evaluate
Debug
ML Paper

Design Complexity

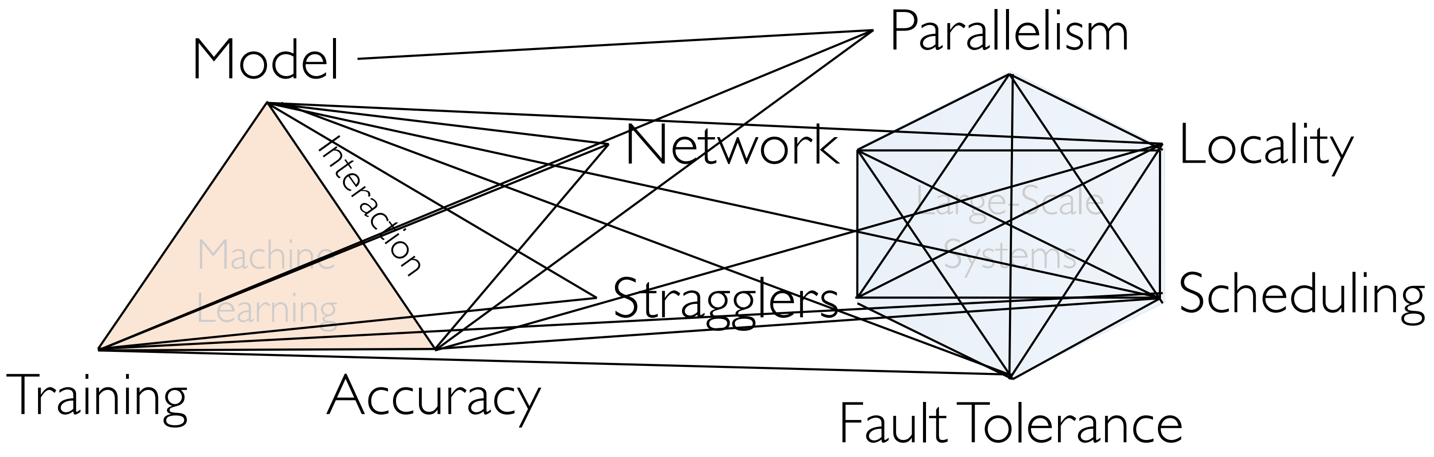




Design Complexity

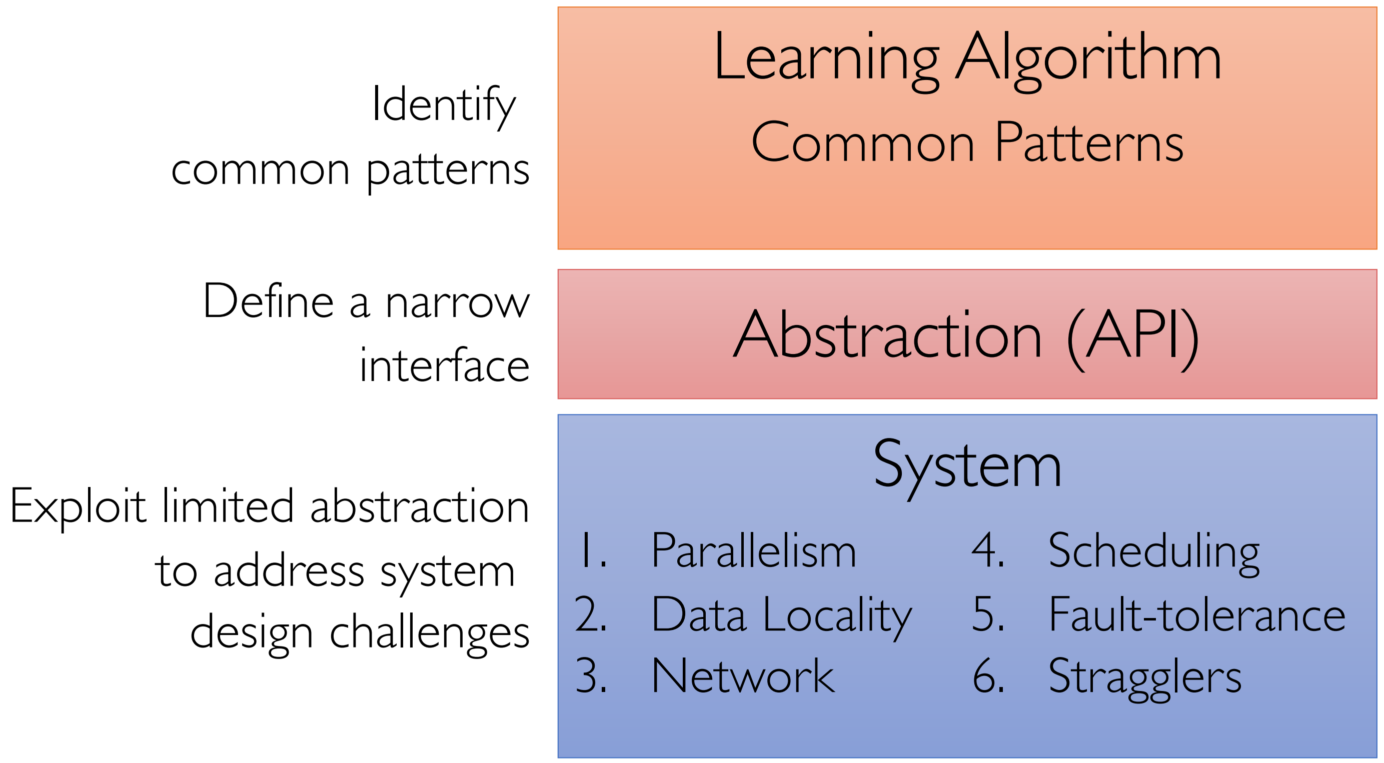


Design Complexity



Learning systems combine the complexities of machine learning with system design

Managing Complexity Through Abstraction

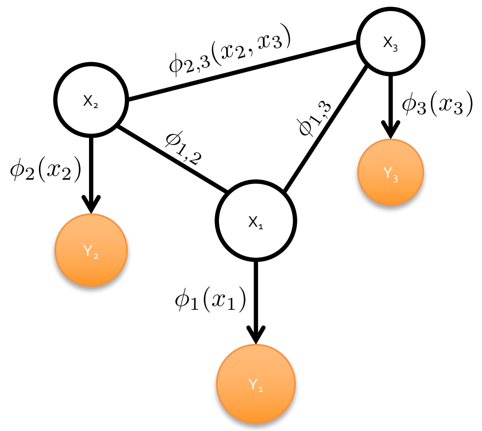


PhD in Machine Learning from CMU 2013

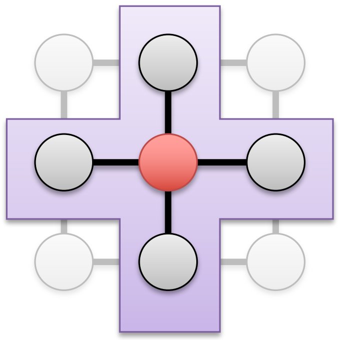
Machine Learning

Abstractions

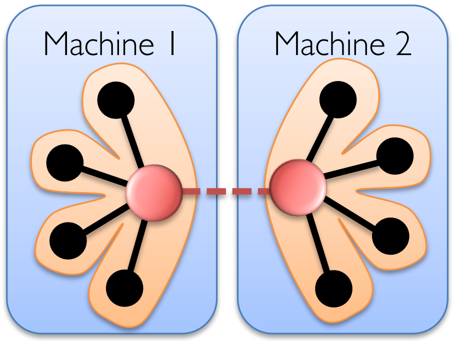
Scalable Systems



Graphical Model Inference



Vertex Program



GraphLab/GraphX System

Looking Back on AI Systems

How did the field evolve during the time I was doing my PhD

ML Systems Workshop ICML'16

OVERVIEW
ACCEPTED PAPERS
CALL FOR PAPERS
IMPORTANT DATES
INVITED SPEAKERS
ORGANIZERS
SCHEDULE
SITEMAP

Overview

The **ML Systems Workshop** will be held as part of ICML in NYC on **June 24, 2016, 8:30am**

Location

Microsoft Technology Center
(11 Times Square, 8th avenue between 41st and 42nd streets)
Room Central Park on 6th Floor
Entrance at the intersection of 8th avenue and 41st street
Phone: 212-245-2100

Map/Directions
2 blocks from the Marriot Marquis Hotel, (between 41st and 42nd streets on 8th Avenue)

Overview

This workshop is a follow up to the ICML audience of the well attended Learning Systems workshop at NIPS 2015 and the Software Engineering for Machine Learning workshop at NIPS 2013.

A new area is emerging at the intersection of machine learning (ML) and systems design. This birth is driven by the explosive growth of diverse applications of ML in production, the continued growth in data volume, and the complexity of large-scale learning systems. The goal of this workshop is to bring together experts working at the crossroads of ML, system design and software engineering to explore the challenges faced when building practical large-scale machine learning systems. In particular, we aim to elicit new connections among these diverse fields, identify tools, best practices and design principles. The workshop will cover ML and AI platforms and algorithm toolkits (Caffe, Tensor Flow, Torch etc), as well as dive into Machine learning focused developments in distributed learning platforms, programming languages, data structures and general purpose GPU programming.

Machine learning community has had an evolving focus on AI Systems

Fast Algorithms — Distributed Algorithms — Deep Learning Frameworks — Transformers Everywhere

2009 $\longrightarrow$ 2022

ML for Systems — Machine Learning Frameworks — RL for Systems — Massive General Models

$\longrightarrow$ Integration of Communities

Systems and the Third Wave of AI

Waves of AI Research

- **1950 to 1974:** *Birth of AI*
  - 1950 Alan Turing publishes the *Imitation Game*
  - 1951 Marvin Minsky builds first neural network machine (SNARC)
  - DARPA starts to pour money into AI Research (limited oversight)
  - ***Incredible optimism***

"Within ten years a digital computer will be the world's chess champion"
-- Herbert A. Simon and Allen Newell (1958)

"In from three to eight years we will have a machine with the general intelligence of an average human being"
– Marvin Minksy (1970)

Waves of AI Research

1950 to 1974: *Birth of AI*

1974 to 1980: *First AI Winter*

- ➢ Technology was unable to meet the high expectations
  - ➢ **Insufficient computer power**
  - ➢ Over emphasis on combinatorial search
  - ➢ Insufficient knowledge (data)
- ➢ 1969 Book "Perceptrons" by Minsky and Papert
  - ➢ Showed that certain basic functions (e.g., parity) cannot be computed using local connections
  - ➢ Strong argument against connectionist approaches to AI (neural networks)
- ➢ Government funding dries
  - ➢ Not meeting objectives
  - ➢ DARPA focuses more on immediate impact

Waves of AI Research

- **1950 to 1974:** *Birth of AI*
- **1974 to 1980:** *First AI Winter*
- **1980 to 1987:** *Second Wave of AI*
  - Expert systems start to solve real-world problems
    - CMU developed XCON (AI for Systems) for DEC → saves $40M annually
    - Combines "knowledge" (expert rules "data") with logic
  - Japanese government invests $850M in AI Research
    - Other governments respond by also investing heavily
  - Emergence of the **AI hardware** and **software industry**

Waves of AI Research

1950 to 1974: *Birth of AI*

1974 to 1980: *First AI Winter*

1980 to 1987: *Second Wave of AI*

1987 to 1993: *Second AI Winter*

- ➢ Expert systems too brittle to maintain
- ➢ Collapse of the specialized **AI hardware market**
  - ➢ Commodity hardware was improving too rapidly
  - ➢ Ultimately over 300 AI companies would vanish from the market
- ➢ Massive government funding cuts

Waves of AI Research

- **1950 to 1974:** *Birth of AI*
- **1974 to 1980:** *First AI Winter*
- **1980 to 1987:** *Second Wave of AI*
- **1987 to 1993:** *Second AI Winter*
- **1993 to 2011:** *AI Goes Stealth Mode (aka Machine Learning)*
  - Hardware becomes fast enough, and AI techniques start to work
    - Deep Blue beats Garry Kasparov (1997)
    - OCR, Speech Recognition, Google Search, …
  - Confluence of ideas and techniques: optimization, statistics, probability theory, and information theory

- **1950 to 1974:** *Birth of AI*
- **1974 to 1980:** *First AI Winter*
- **1980 to 1987:** *Second Wave of AI*
- **1987 to 1993:** *Second AI Winter*
- **1993 to 2011:** *AI Goes Stealth Mode (aka Machine Learning)*
  - Hardware becomes fast enough, and AI techniques start to work
    - Deep Blue beats Garry Kasparov (1997)
    - OCR, Speech Recognition, Google Search, …
  - Confluence of ideas and techniques: optimization, statistics, probability theory, and information theory

- **1974 to 1980:** *First AI Winter*
- **1980 to 1987:** *Second Wave of AI*
- **1987 to 1993:** *Second AI Winter*
- **1993 to 2011:** *AI Goes Stealth Mode (aka Machine Learning)*
- **2011 to 2019:** *Third Wave (AI Goes Deep)*
  - Large quantities of **data** in conjunction with **advances in hardware** and **software** enable the **design** and **training** of **complex models**
  - New applications emerge
    - Autonomous driving, home automation, AR/VR, …
  - AI Market frenzy →
    - Gartner Forecasts Worldwide Artificial Intelligence Software Market to Reach $62 Billion in 2022
    - Worldwide revenues for the artificial intelligence (AI) market, including software, hardware, and services, are forecast to grow 16.4% year over year in 2021 to $327.5 billion, according to the latest release of the International Data Corporation (IDC) Worldwide Semiannual Artificial Intelligence Tracker.

Will there be a 3rd winter?

Joey's Forecast: A Chilly Spring and a Warm Summer

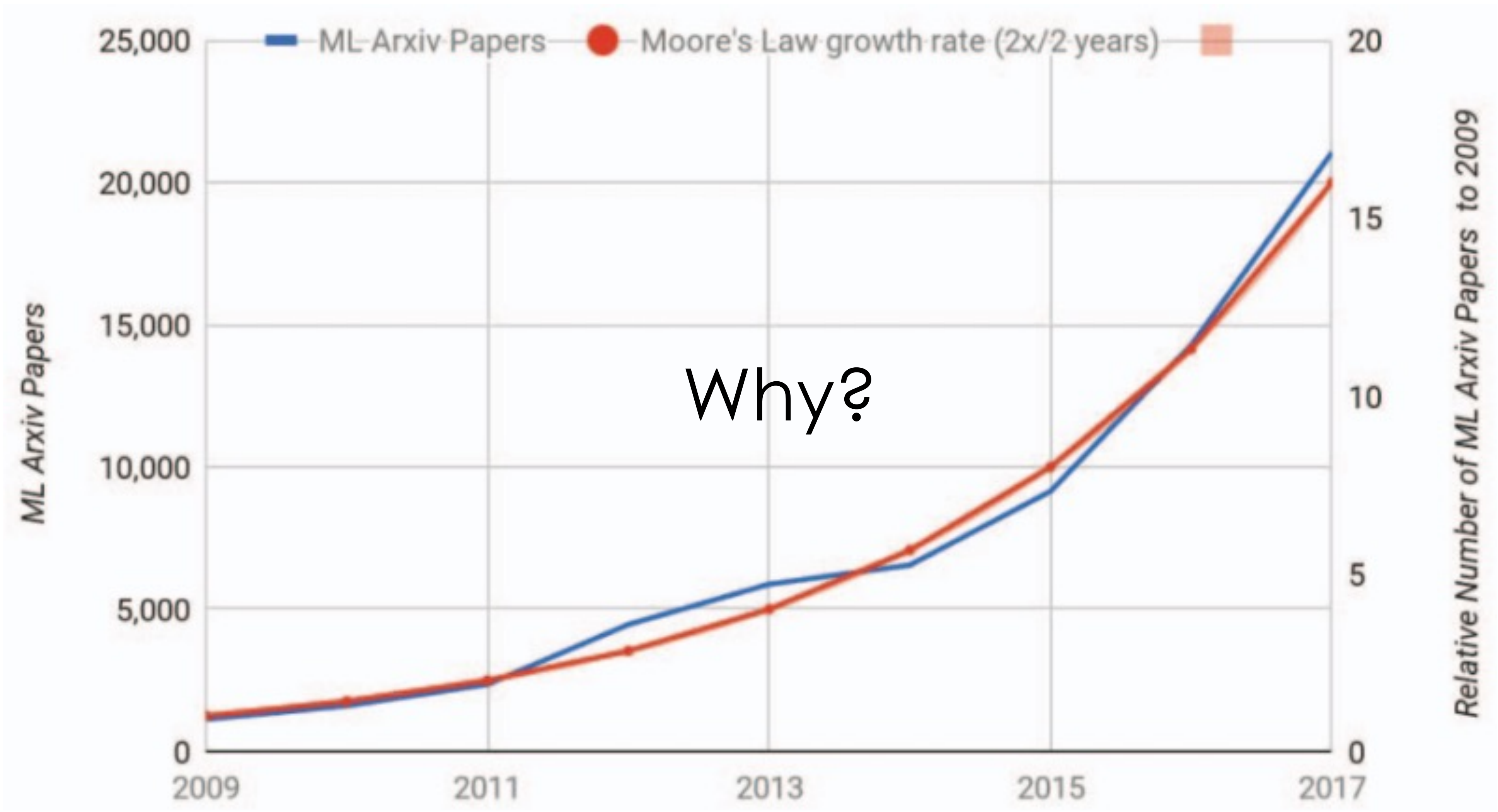
Hype still exceeds reality…

However, we are finally delivering significant value from AI-Systems.

Those systems are also continuing to improve with investment.

They are enabling new products and creating market opportunities.

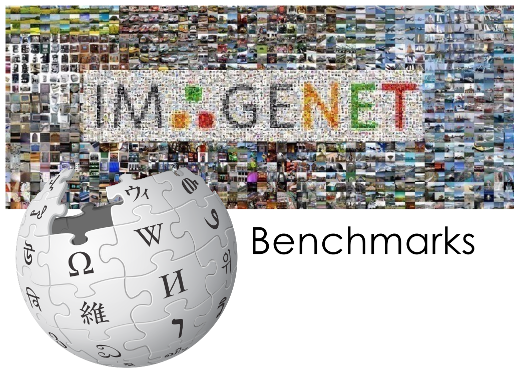
Research in AI and Systems for AI is Exploding



“A New Golden Age in Computer Architecture: Empowering the Machine-Learning Revolution”, https://ieeexplore.ieee.org/document/8259424

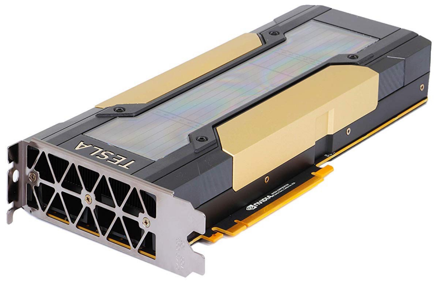
New Forces Driving AI Revolution

Data



Benchmarks

Compute



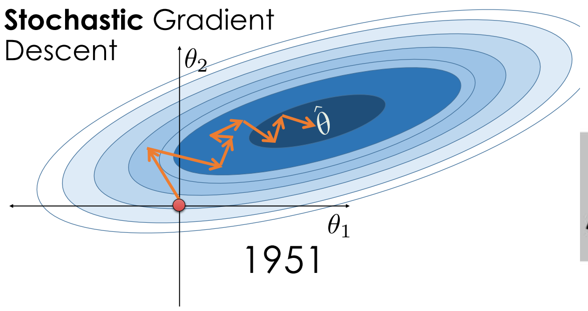
Abstractions

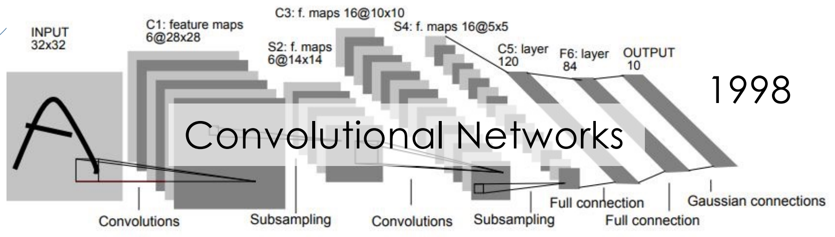
PyTorch

Caffe

TensorFlow

Advances in Algorithms and Models





What defines good AI-Systems Research Today?

What defines good AI-Systems Research Today?

Big Ideas in ML Research

- Generalization (Underfitting/Overfitting)
  - What is being “learned”?
- Inductive Biases and Representations
  - What assumptions about domain enable efficient learning?
- Efficiency (Data and Computation)
  - How much data and time are needed to learn?
- Details: Objectives/Models/Algorithms

What makes a great (accepted) paper?

- **State of the art results**
  - Accuracy, Sample Complexity, Qualitative Results **…**
- **Novel settings,** problem formulations, and **benchmarks**
- Innovation in **techniques**: architecture, training methodology, …
- **Theoretical results** that provide a deeper understanding

----

- **Narrative** and **framing** in prior work and **current trends?**
- **Parsimony?** Are elaborate solutions rejected? If they work better?
- **Verification** of prior results?

What defines good AI-Systems Research Today?

Big Ideas in Systems Research

- **Managing Complexity**
  - Abstraction, modularity, layering, and hierarchy
- **Tradeoffs**
  - What are the fundamental constraints?
  - How can you reach new points in the trade-off space?
- **Problem Formulation**
  - What are the requirements and assumptions?

What makes a great (accepted) paper?

- **State of the art results**
  - throughput, latency, resource reqs., scale, …
- **Problem formulations** and **benchmarks**
- Innovation in **techniques**
  - Algorithms, data-structures, policies, software abstractions.
- What you **remove** or **restrictions** often more important
- **Narrative** and **framing** in prior work and **current trends?**
- **Verification** of prior results?
- **Open source?** Real-world use?

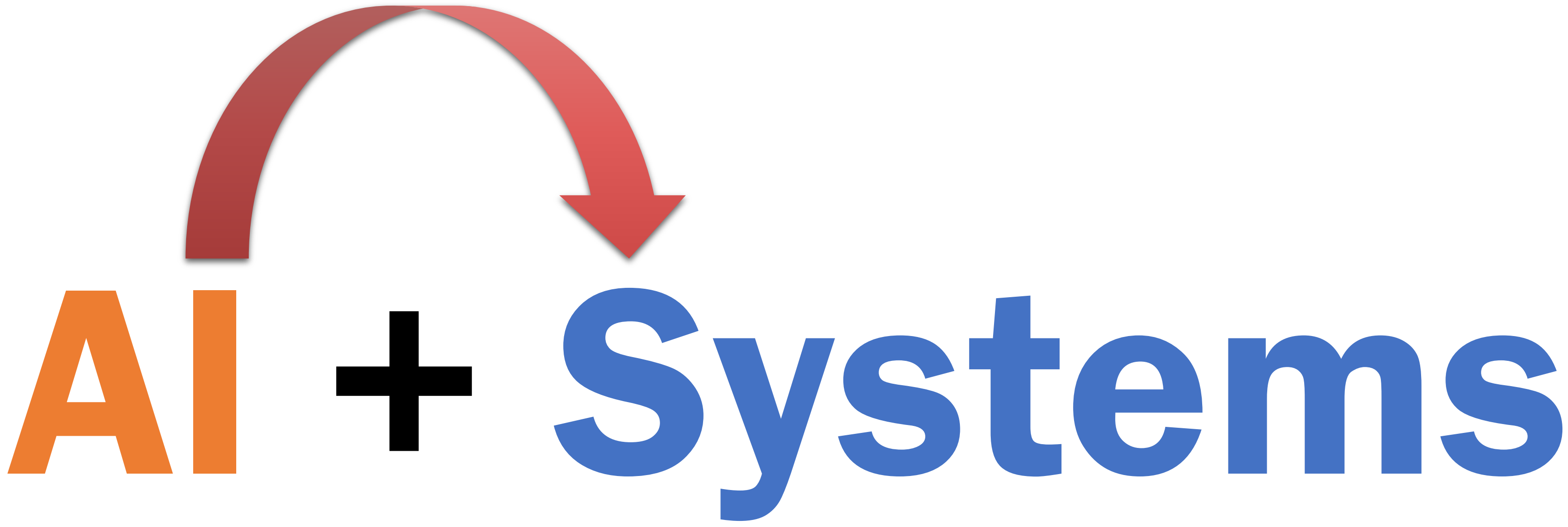
What defines good AI-Systems Research Today?

What is AI-Systems Research?

- Good AI and Systems research
  - Provides **insights to both communities**
  - **Builds on big ideas** in prior AI and Systems Research
- Leverages understanding of both domains
  - Studies **statistical and computational tradeoffs**
  - Identify **essential abstractions** to bridge AI and Systems
  - Reframes **systems problems as learning problems**
- More than just great open-source software!
  - But software impact often matters…

Kinds of AI-Systems Research

Advances in **AI** are being used to address fundamental challenges in **systems**.



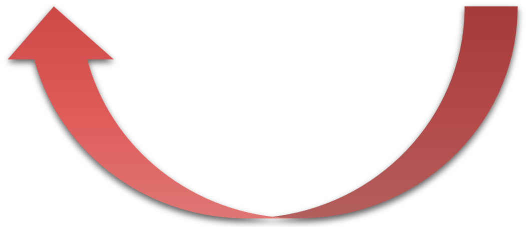
AI + Systems

- Reinforcement Learning for
  - Pandas code generation
  - SQL join planning
  - Network packet classification
  - Autoscaling
- Bandit Algorithms for radio link adaptation
- Wireless link quality estimation
- Multi-task learning for straggler mitigation
- VM Selection using Trees ..

AI + Systems

Advances in **systems** are enabling substantial progress in **AI**

AI + Systems



Developing Systems for:

- Autonomous Vehicles
- Reinforcement Learning
- Secure Machine Learning
- Prediction Serving
- Experiment Management

Advancing AI

- Dynamic Neural Nets
- Prediction on Compressed Data
- Distributed Training
- Distributed Auto-ML

AI + Systems Research

riselab UC Berkeley

BAIR

BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Caffe

APACHE Spark™

Scalabel

Clipper

RAY

MLbase

KeystoneML

rllib

tune

About You?

Background Requirements

- You have taken **previous courses** in either
  - Systems (for example CS162) → comfortable **building systems** and familiar with **big ideas in system design**
  - Machine Learning (for example CS189) → comfortable **training neural networks** and familiar with **big ideas in ML**

- You are **actively involved in research** in either systems or machine learning (or both)

If this does not describe you, email me (jegonzal@Berkeley.edu) and we can talk about your situation.

Enrolling in the Class (and the Waitlist)

- We were able to **extend to 45 students**
- Currently prioritizing **PhD** and **MS students** (>45 students)
  - Willing to admit strong undergrads if there is room
  - Fill out this form: https://forms.gle/cn1DS5owivBbniA78
- In previous offerings around **10% of students dropped**
- This is a seminar class – **attendance is required.**
  - **You may attend virtually (via zoom)!**

If you would like to enroll please participate in first weeks.

Reasons not to take this class … ☹

- If you want to learn how to **train models** and **use TensorFlow**, **PyTorch**, and **SkLearn**
  - Take: CS C100, CS182, CS189
- If you want learn how to **use big data systems**
  - Take: CS162, CS186
- You are ***not*** interested in learning how to **evaluate**, **conduct**, and **communicate** research
  - Why not?
- You are **unable to (virtually) attend lecture**

If you plan to drop this class, please drop it soon so others can enroll!

Goals for the Class

What do you expect from this class?

My Goals for the Class

- Identify and **solve impactful problems** at the intersection of AI and Systems
- Learn about the **big ideas** and **key results** in AI systems
- Learn how to read, evaluate, and **peer review papers**
- Learn how to **write great papers** that also get published!

Have Fun!

How will we achieve these goals?

- **Lectures and Reading:** study developments in AI Systems
  - Identify the key open problems and research opportunities
  - **Guest speakers** every week to **discuss research** and **process**.
- **Projects:** collaboration between Systems and AI students
  - Explore open problems and produce top tier publications
- **In Class Reviews:** weekly reviews of papers
  - Learn both how to evaluate papers and understand how papers are reviewed

Simple Grading Policy

20% Class Participation

- **Attend lecture** and **participate** in class discussions
- **Lead 3 in class** discussions (sign-up)
  - https://docs.google.com/spreadsheets/d/1mh9JG1CVgUKMy98xSm8ssl8FEWblTL_8rcBjH2tk0SE/edit?usp=sharing

20% Weekly Reviews

- **Submit weekly paper reviews** (google form)!

60% Class Projects

- Do great research!

Problems

What makes a good problem?

What makes a **good problem?**

- **Impact:** People care about the solution
  - … and progress advances our understanding (**research**)
- **Metrics:** You know when you have succeeded
  - Can you **measure progress** on the solution?
- **Divisible:** The problem can be divided into smaller problems
  - You can identify the first sub-problem.
- **Your Edge:** Why is it a good problem ***for you?***
  - Leverage your strengths and imagine a new path.

Can you Solve a Solved Problem?

- Ideally you want to solve a **new** and **important** problem
- A **new solution** to a solved problem can be impactful if:
  - It supports a **broader set of applications** (users)
  - It **reveals a fundamental trade-off** or
  - Provides a **deeper understanding** of the problem space
  - **10x Better?**
    - Often publishable…
    - Should satisfy one of the three above conditions.

Class Organization

Weekly Topic Organization

- Each week we will cover a new topic
- There will be **3 required papers** to read each week
- **Format of Each Lecture (3 hours!!!!)***
  - **~45 Minute** review of topic area for the **next WEEK**
    - We want to give context to help with reading
  - **~60 Minute: Extended PC discussion** around the reading
    - **20 minutes** per paper **3 presenters** (Summary, Strengths, Weaknesses)
    - **Fixed Slide Format (Link)**
  - **~15 Minute Break**
  - **60 Minutes: Presentation from Invited Speaker** with **Discussion**

*Format may change based on speaker constraints.

PC Meeting Format

https://docs.google.com/presentation/d/1il7arlEPDTQp2VpHtOTS_ghg_oG03TT0ulwyqG6ZHUM/edit?usp=sharing

In Class PC Meeting Format (V.1)

For each of the three papers: (20 Minutes per paper)

- **Neutral:** recap of the paper (neutral opinion) [5 Minutes]
- **Advocate:** Strengths of the paper [5 Minutes]
- **Critic:** Weaknesses of the paper [5 Minutes]
- Class will discuss **rebuttal** and **improvements** [5 Minutes]

The **Neutral** Presenter will Summarize

- What is the **problem** being solved and what was the **solution**? (Summary!)
- What is the **context of this work**? What is the **most relevant related work?**
- What **metrics** did they use to evaluate their solution?
  - What was the **Baselines** of comparison?
- What was the **key insight** or **enabling idea**?
- What are the **claimed technical contributions**?

Advocate and **Critic** Will Discuss

- **Novelty** and **Impact**
  - Are the problem and solution novel and how will the solution affect future research?
- **Technical Qualities**
  - Are the problem **framing** and **assumptions** reasonable
  - Discuss merits of the technical **contributions**
  - Does the **evaluation** support claims and reveal limitations of the proposed approach?
- **Presentation**
  - Discuss the **writing clarity** and **presentation of results**
  - Positioning of **related work**

Break + Action Items

- Go to website: https://ucbrise.github.io/cs294-ai-sys-sp22/
- Join Slack (we are discontinuing piazza)
  - https://join.slack.com/t/cs294-ai-sys-sp22/shared_invite/zt-120vm629j-pos6xUlfrt_xgaCMM_aaUw
- Signup for **3 discussion** slots as **different roles here**:

https://docs.google.com/spreadsheets/d/1mh9JG1CVgUKMy98xSm8ssl8FEWbITL_8rcBjH2tk0SE/edit?usp=sharing

Context for
Big Data Systems

Why discuss big data systems in this class?

- Data is central to machine learning
  - But what systems do you use for your data?
- Data systems are often overlooked in ML-Systems research
  - Potential area for exciting new research
- *Industry:* **Data systems control everything!**
- **3 Papers:**
  - How to incorporate ML into a classic database systems
  - How to scale data processing for ML workloads
  - Current trends in large-scale data systems driven by ML

Relational Database Systems

- *Logically* organize data in **relations** (tables)
- Structured Query Language **(SQL)** to define, manipulate and compute on data.
  - A common language used by many data systems
  - Describes logical organization of data as well as computation

SQL is a **Declarative** Language

- **Declarative:** "Say ***what*** you want, not ***how*** to get it."
  - **Declarative Example:** *I want a table with columns "x" and "y" constructed from tables "A" and "B" where the values in "y" are greater than 100.00.*
  - **Imperative Example:** For each record in table "A" find the corresponding record in table "B" then drop the records where "y" is less than or equal to 100 then return the "x" and "y" values.
- **Advantages** of declarative programming
  - Enable the system to find the best way to achieve the result.
  - More compact and easier to learn for non-programmers (Maybe?)
- **Challenges** of declarative programming
  - System performance depends heavily on automatic optimization
  - Limited language (not Turing complete) → need extensions

Physical **Data Independence**

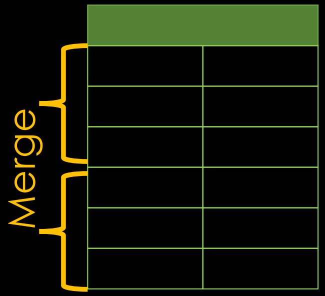
- Physical data layout/ordering is **determined by system**
  - goal of maximizing performance
- **Data Clustering**
  - Organize group of records to improve access efficiency
  - Example: grouped/ordered by key
- **Implications on Learning?**
  - Record ordering may depend on data values
  - Arbitrary ordering ≠ Random ordering

User Defined Aggregates

- Provide a **low-level API** for defining functions that aggregate state across records in a table
  - Much like **fold** in functional Programming

```
CREATE AGGREGATE agg_name (…){
   # Initialize the state for aggregation.
   initialize(state) → state
   # Advance the state for one row.  Invoked repeatedly.
   transition(state, row) → state
   # Compute final result.
   terminate(state) → result
   # (Optional) Merge intermediate states from parallel executions.
   merge(state, state) → state

}
```



Closed Relational Model and Learning

- All operations on tables produce tables…
- Training a model on a table produces?
  - A row containing a model
  - A table containing model weights
  - An (infinite) table of predictions
    - MauveDB: Supporting Model-based User Views in Database Systems
- Predictions as views
  - Opportunity to **index** predictions
  - Relational operations to manipulate predictions

Out-of-core Computation

- Database systems are typically designed to operate on **databases larger than main memory** (big data?)
- Algorithms must manage **memory buffers** and **disk**
  - Page level memory buffers
  - Sequential reads/writes to disk
- Understand **relative costs** of memory vs disk

Reasoning about Memory Hierarchy

Latency Numbers Every Programmer Should Know --Jeff Dean

L1 Cache	**0.5**	**ns** (few clock cycles)
L2 Cache	**7**	**ns**
Main Memory	**100**	**ns**
Read 1MB from RAM (Seq.)	**250K**	**ns**
Read 1MB SSD (Seq.)	**1M**	**ns** (1ms)
Read 1MB Disk (Seq.)	**20M**	**ns** (20ms)

Reasoning about Memory Hierarchy

Latency Numbers Every Programmer Should Know	***Human Readable***	**Database Systems**
L1 Cache	**1 second**	
L2 Cache	**14 seconds**	
Main Memory	**3.3 minutes**	Page Buffers
Read 1MB from RAM (Seq.)	**5.8 days**	Page Buffers
Read 1MB SSD (Seq.)	**23 days**	Sequential Read/Write
Read 1MB Disk (Seq.)	**1.3 years**	Sequential Read/Write

Transactional vs Analytics Data Systems

- **Transactional (OLTP) Systems:** *address source of truth data used by live services*
  - *Bank records, user profiles,*
  - *Designed for low-latency, availability, transactional consistency, durability*
- ***Analytics (OLAP) Systems :*** *store and manage data to support large-scale analysis*
  - *Business intelligence, sales, fraud, targeting*
  - *Designed for high-throughput, scalability, rich computation*

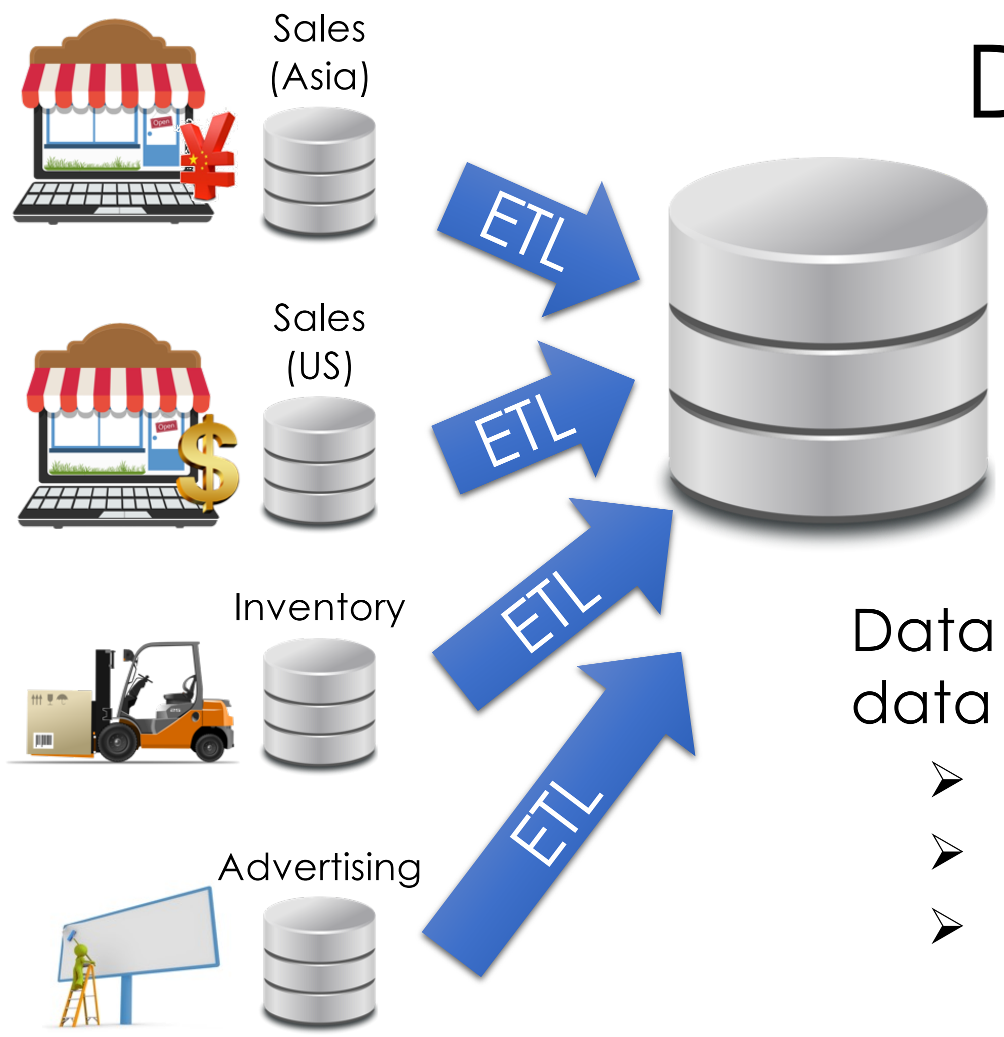
Sales
(Asia)
Sales
(US)
Inventory
Advertising

Operational Data Stores

- Capture **the now**
- Many different databases across an organization
- Mission critical… be careful!
  - Serving live ongoing business operations
  - Managing inventory
- Different formats (e.g., currency)
  - Different schemas (acquisitions ...)
- Live systems often don't maintain history

We would like a consolidated, clean, historical snapshot of the data.

Data Warehouse



Collects and organizes historical data from multiple sources

Data is *periodically* **ETL**ed into the data warehouse:

- **Extracted** from remote sources
- **Transformed** to standard schemas
- **Loaded** into the (typically) relational (SQL) data system

Extract → Transform → Load (ETL)

Extract & Load: provides a snapshot of operational data

- Historical snapshot
- Data in a single system
- Isolates analytics queries (e.g., Deep Learning) from business critical services (e.g., processing user purchase)
- Easy!

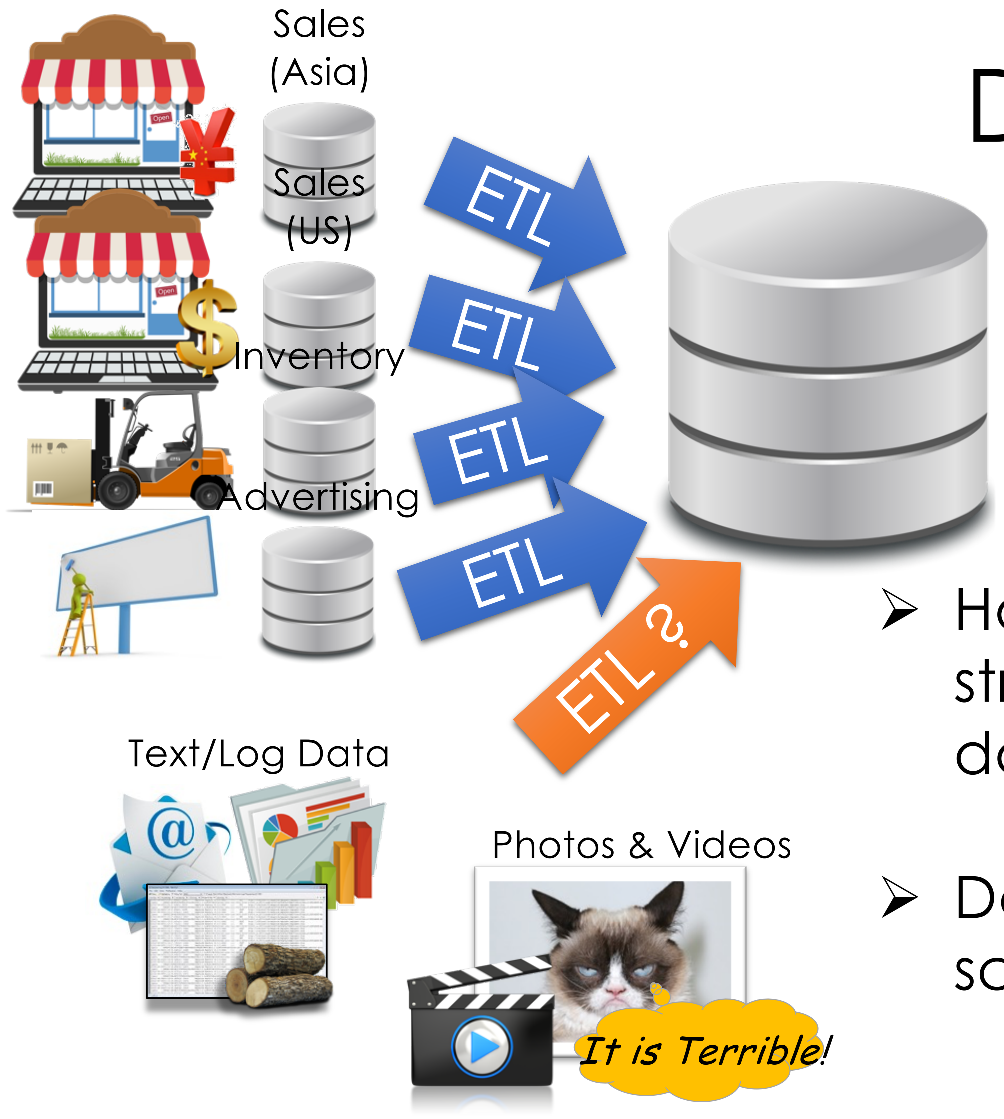
Transform: clean and prepare data for analytics in a unified representation

- **Difficult →** often requires specialized code and tools
- Different schemas, encodings, granularities

Recent Trend: ETL --> ELT

- Previously transformation was needed to **reduce burden** on data warehouse
- **Improvements** in data warehouse **performance** have eliminated the need to transform before loading
  - Driven by trends in datalakes …
- New ecosystem of software for:
  - **Extract and Load:** e.g., Fivetran and Stitch
  - **Transform:** DBT

Data Warehouse

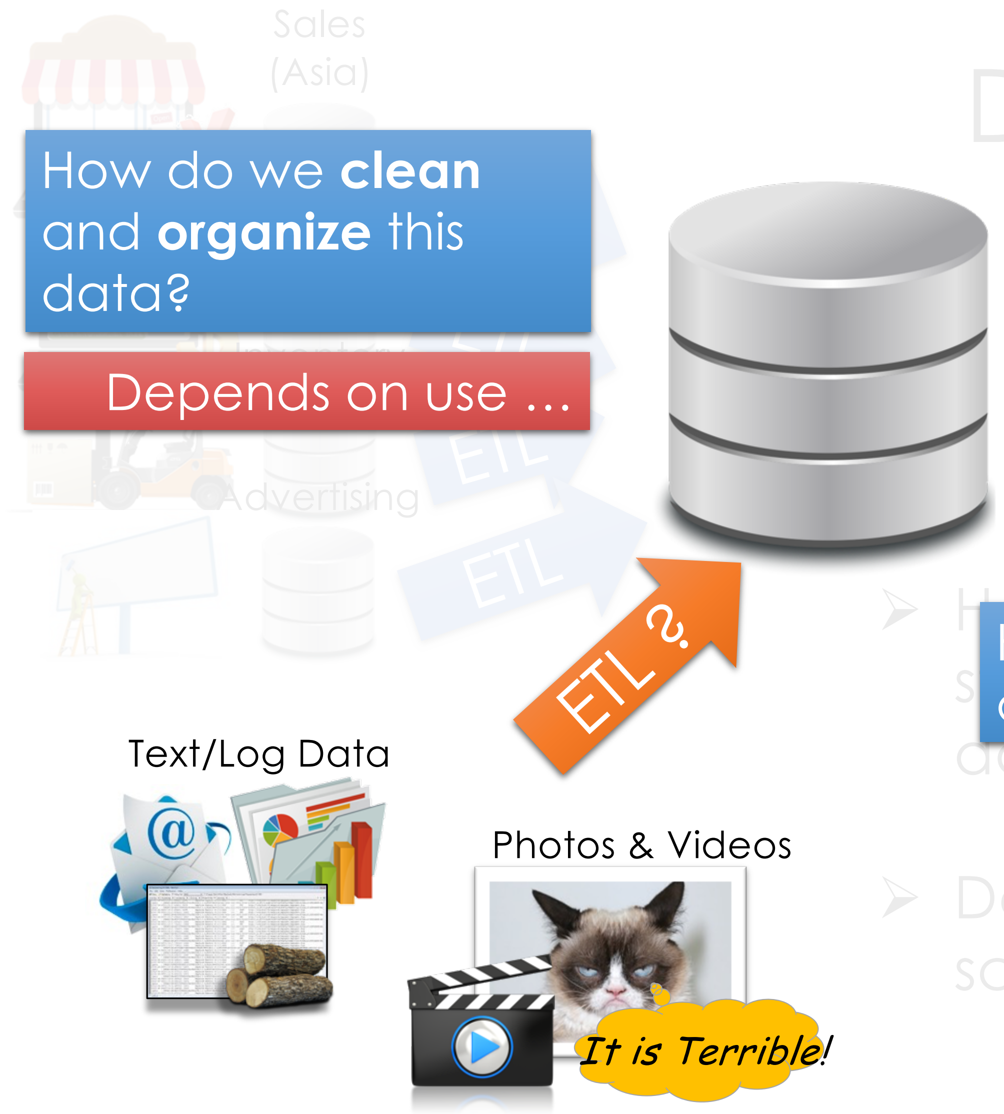


Collects and organizes historical data from multiple sources

- How do we deal with semi-structured and unstructured data?
- Do we really want to force a schema on load?

Data Warehouse

Collects and organizes historical data from multiple sources

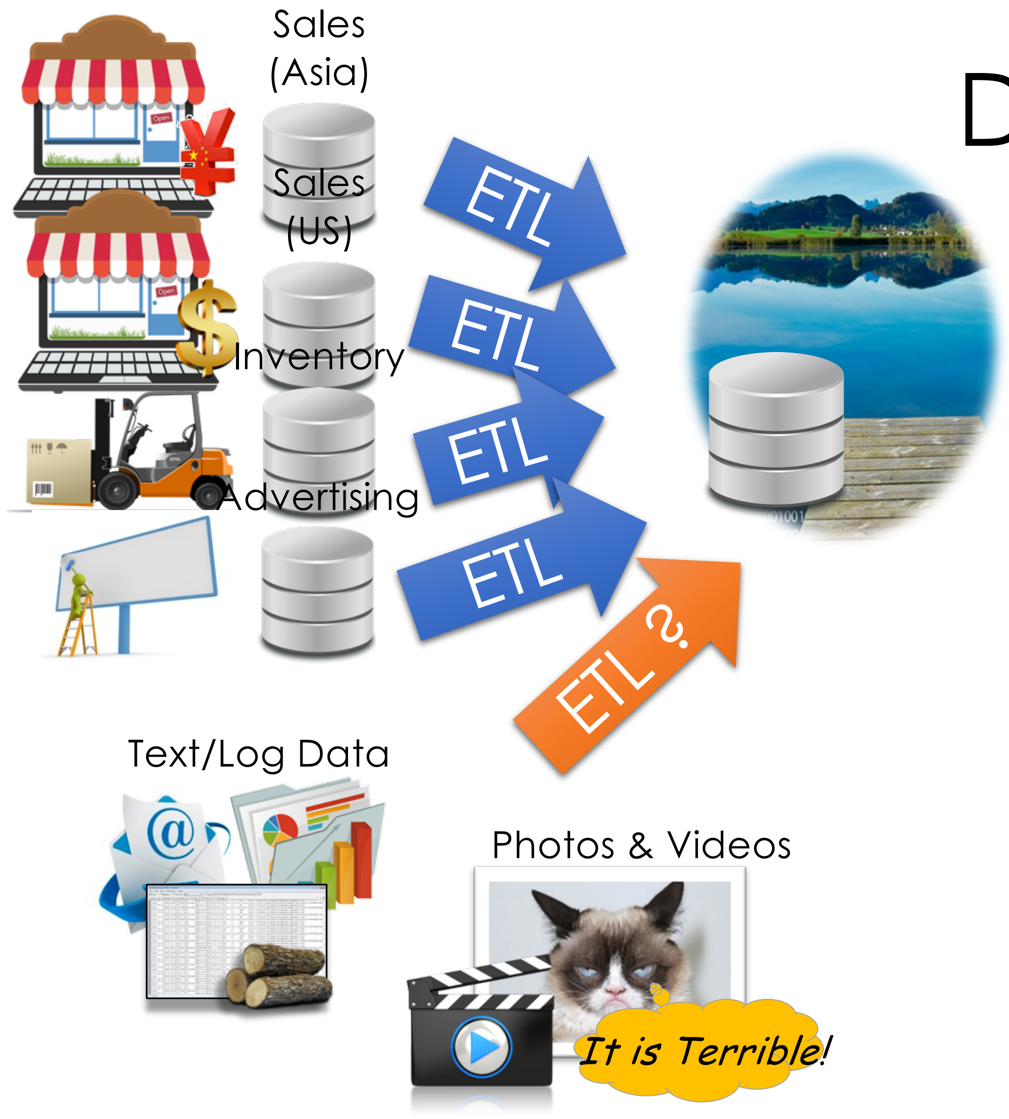


How do we **clean** and **organize** this data?

Depends on use …

How do we **load** and **process** this data in a relational system?

Depends on use…
Can be difficult...
Requires thought...



Data Lake*

Store a copy of all the data

- in one place
- in its original "natural" form

Enable data consumers to choose how to transform and use data.

- *Schema on Read*

What could go wrong?

*Still being defined…[Buzzword Disclaimer]

The Dark Side of Data Lakes

- Cultural shift: *Curate* → *Save Everything!*
  - Noise begins to dominate signal
- Limited data governance and planning

  Example: *hdfs://important/joseph_big_file3.csv_with_json*
  - **What** does it contain?
  - **When** and **who** created it?
- No cleaning and verification → lots of dirty data
- New tools are more complex and old tools no longer work

A Brighter Future for Data Lakes

- Data Scientists and ML Engineers
  - Distributed data processing and cleaning
  - Machine learning, computer vision, and statistical sampling
- Technologies are improving
  - SQL over large files
  - Self describing columnar file formats (Parquet, ORC)
- Organizations are evolving
  - Tracking data usage and file permissions
  - New job title: data engineers
- Starting to see convergence of **warehouse** and **data lakes**
  - **Lakehouse™?**

How do we **store** and **compute** on large unstructured datasets

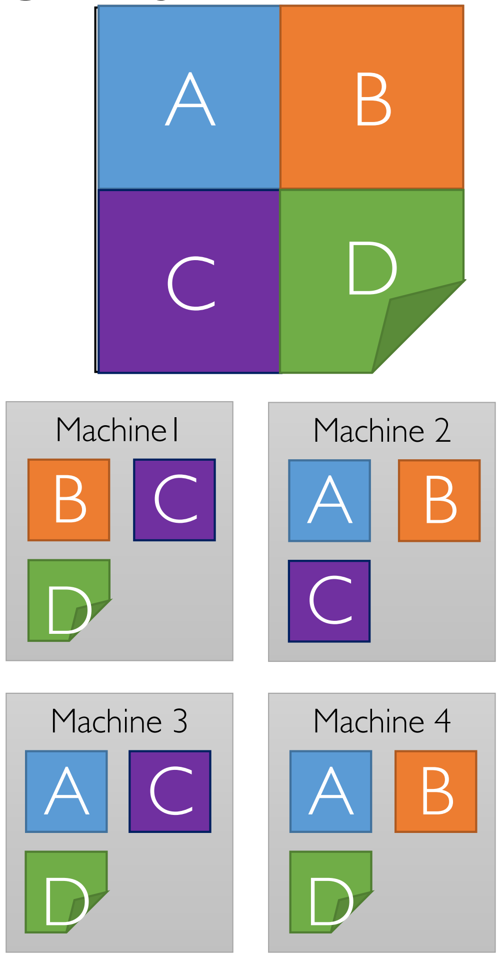
- Requirements:
  - Handle very **large files** spanning **multiple computers**
  - Use **cheap** commodity devices that **fail frequently**
  - **Distributed data processing** quickly and **easily**
- Solutions:
  - **Distributed file systems →** spread data over multiple machines
    - Assume machine **failure** is common → **redundancy**
  - **Distributed computing →** load and process files on multiple machines concurrently
    - Assume machine **failure** is common → **redundancy**
    - **Functional programming** computational pattern → **parallelism**

Fault Tolerant Distributed File Systems

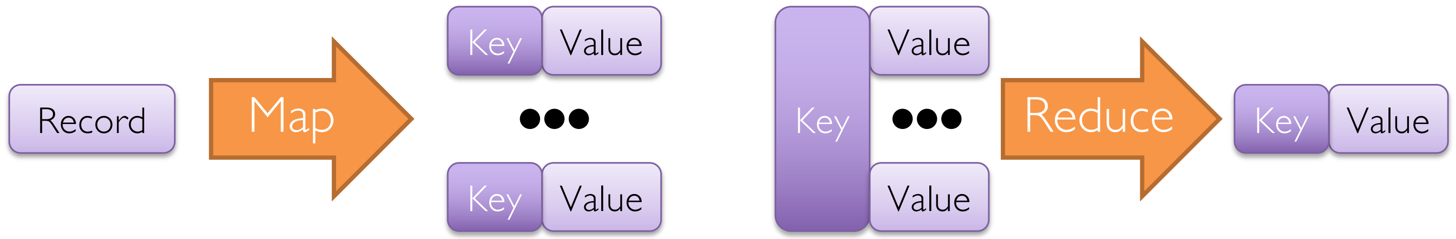
[Ghemawat et al., SOSP'03]

- Split large files over multiple machines
  - Easily support massive files spanning machines
- Read parts of file in parallel
  - Fast reads of large files
- Often built using cheap commodity storage devices

Cheap commodity storage devices will fail!



The Map Reduce Abstraction



Example: *Word-Count*

```
Map(docRecord) {
  for (word in docRecord) {
    emit (word, 1)
  }
}
```

```
Reduce(word, counts) {
  emit (word, SUM(counts))
}
```

Map: Deterministic

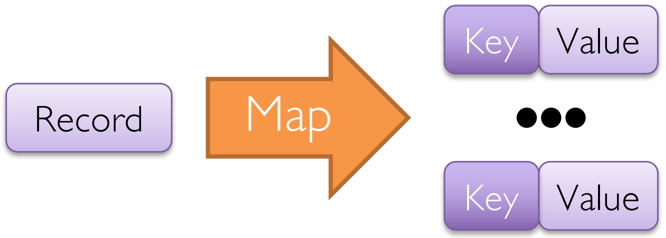
Reduce: Commutative and Associative

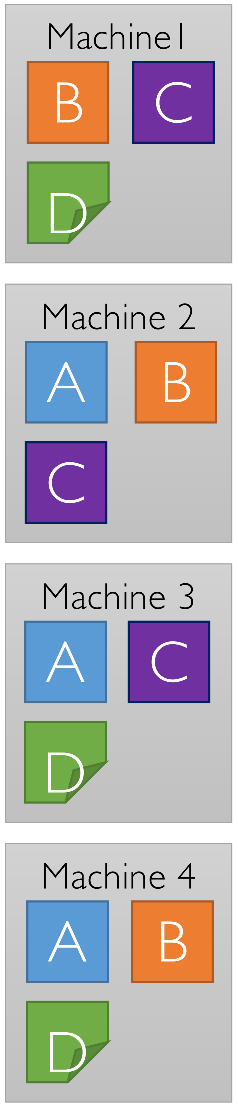
[Dean & Ghemawat, OSDI'04]

Key properties of Map And Reduce

- **Deterministic <u>Map</u>:** allows for re-execution on failure
  - If some computation is lost we can always re-compute
  - Issues with samples?
- **Commutative <u>Reduce</u>:** *allows for re-order of operations*
  - Reduce(A,B) = Reduce(B,A)
  - Example (addition): A + B = B + A
  - Is floating point math commutative?
- **Associative <u>Reduce</u>:** *allows for regrouping of operations*
  - Reduce(Reduce(A,B), C) = Reduce(A, Reduce(B,C))
  - Example (max): max(max(A,B), C) = max(A, max(B,C))

Executing Map Reduce



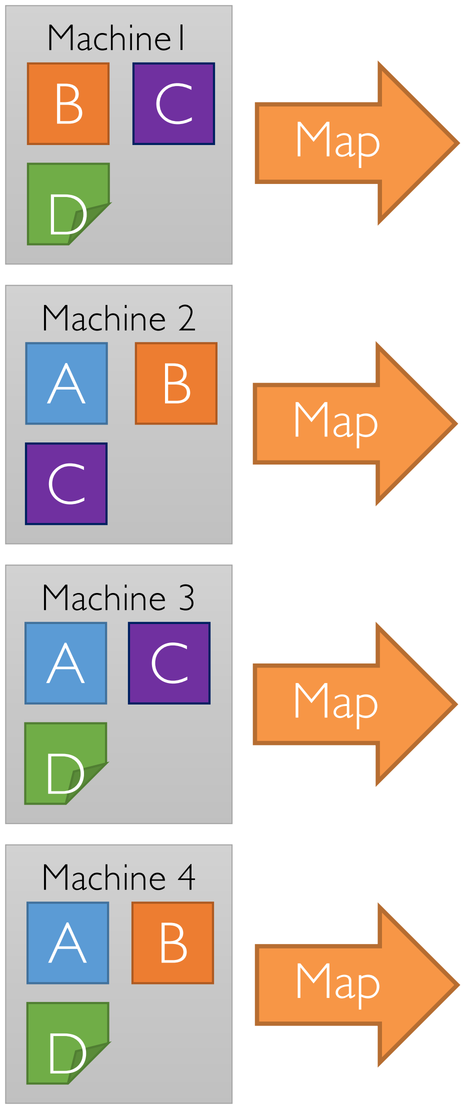


Executing Map Reduce

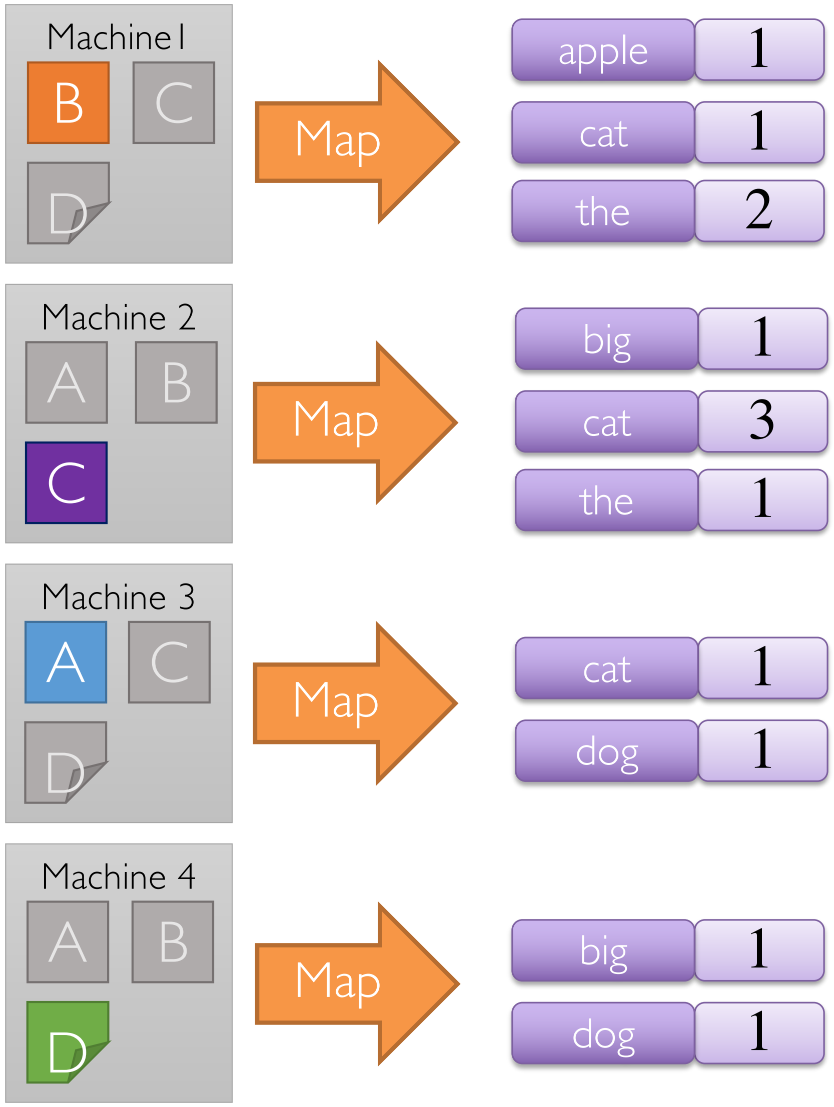
Distributing the Map Function

Executing Map Reduce

Distributing the Map Function



Executing Map Reduce

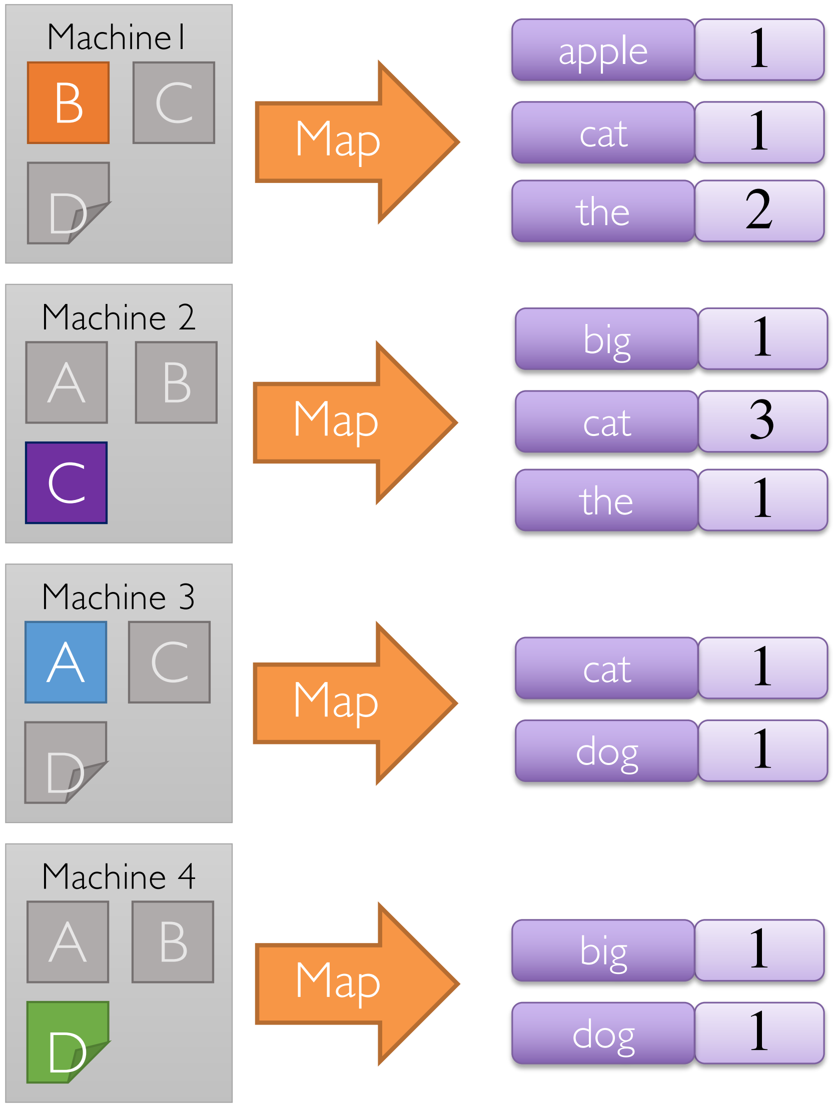


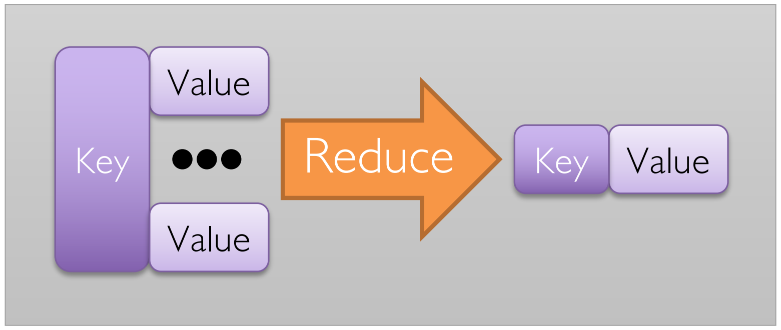
The map function applied to a local part of the big file.

Run in Parallel.

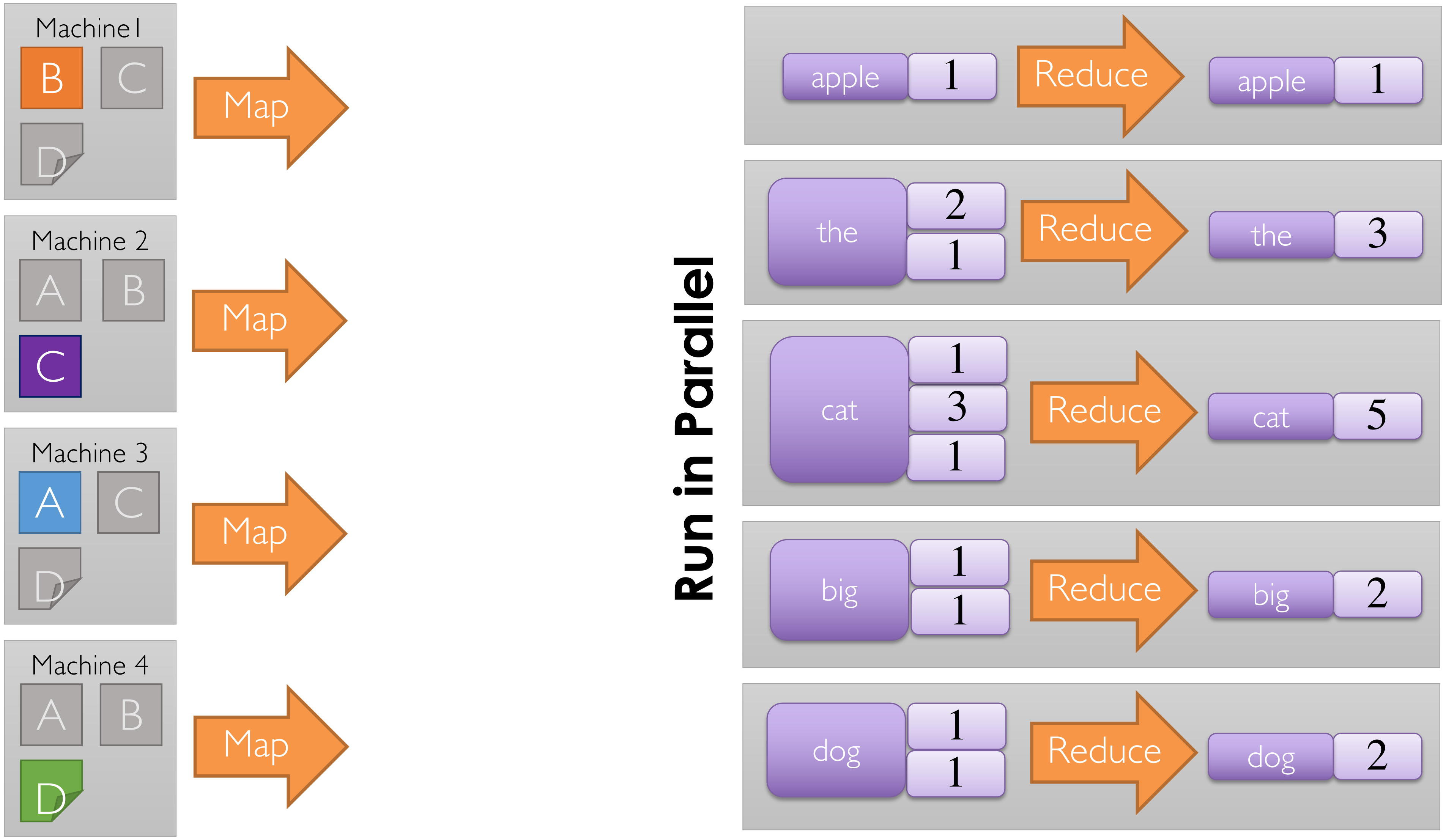
Output is cached for fast recovery on node failure

Executing Map Reduce

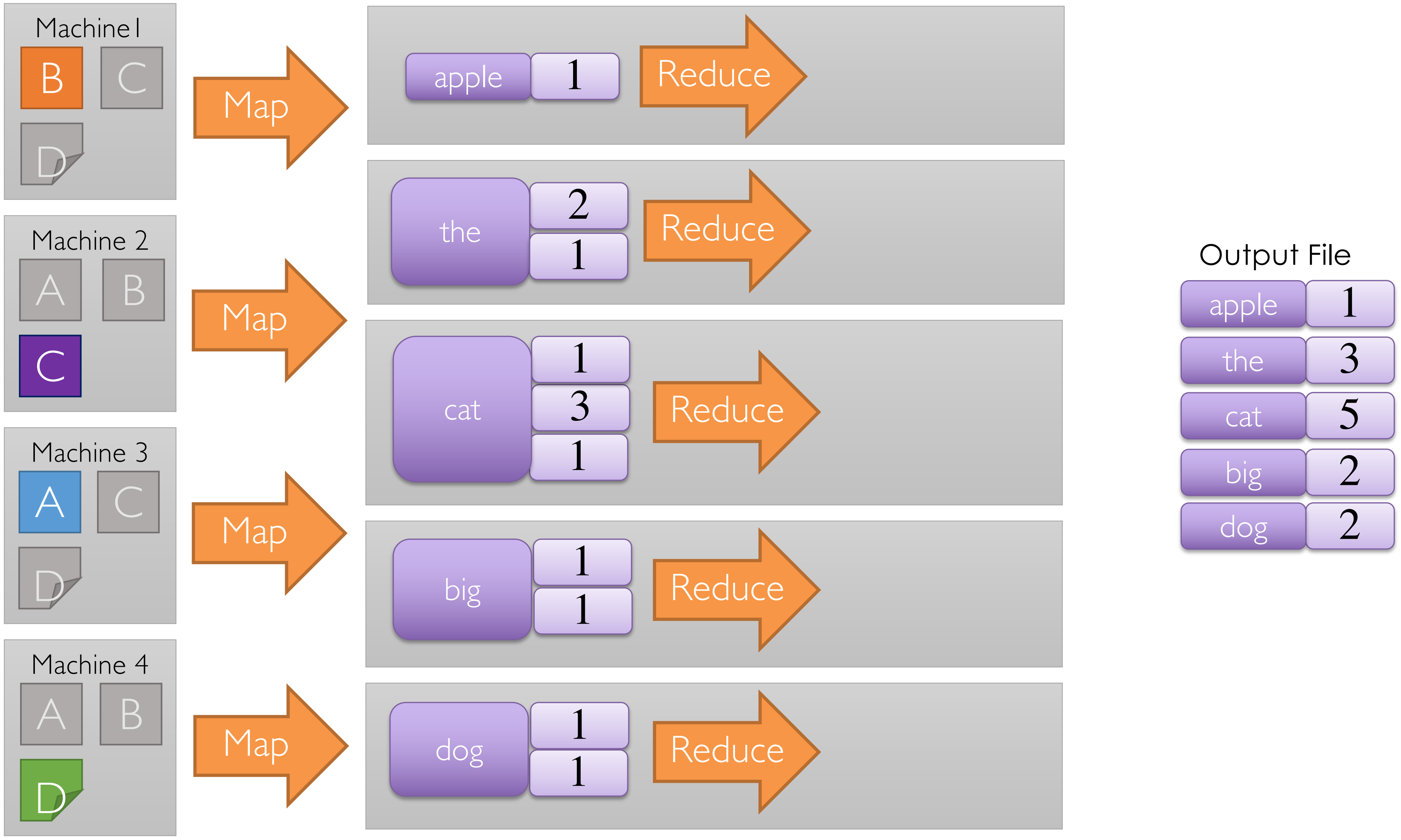




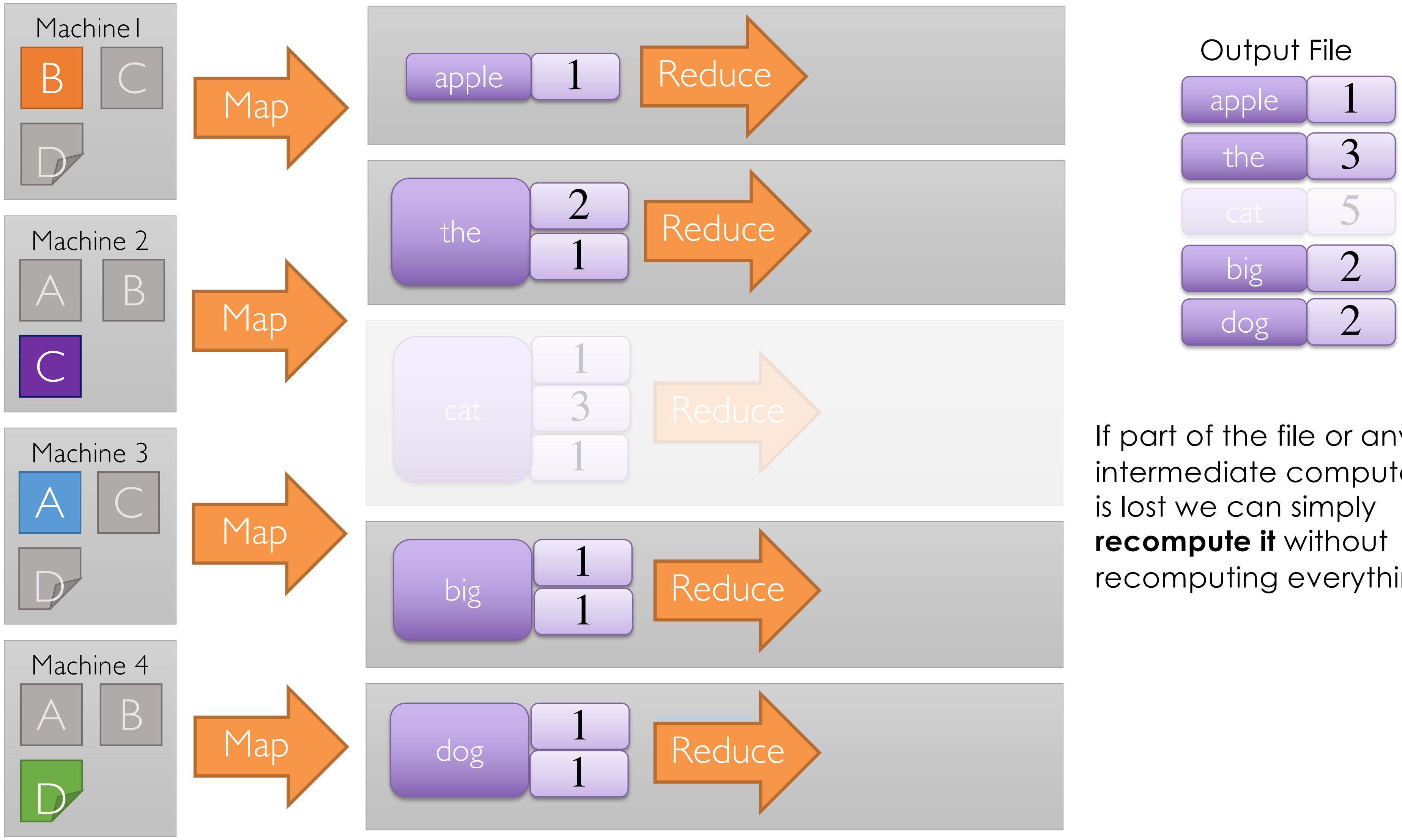
Reduce function can be run on many machines …



Machine 1
B
C
D
Map
Machine 2
A
B
C
Map
Machine 3
A
C
D
Map
Machine 4
A
B
D
Map
Run in Parallel
apple 1
Reduce
apple 1
the 2 1
Reduce
the 3
cat 1 3 1
Reduce
cat 5
big 1 1
Reduce
big 2
dog 1 1
Reduce
dog 2

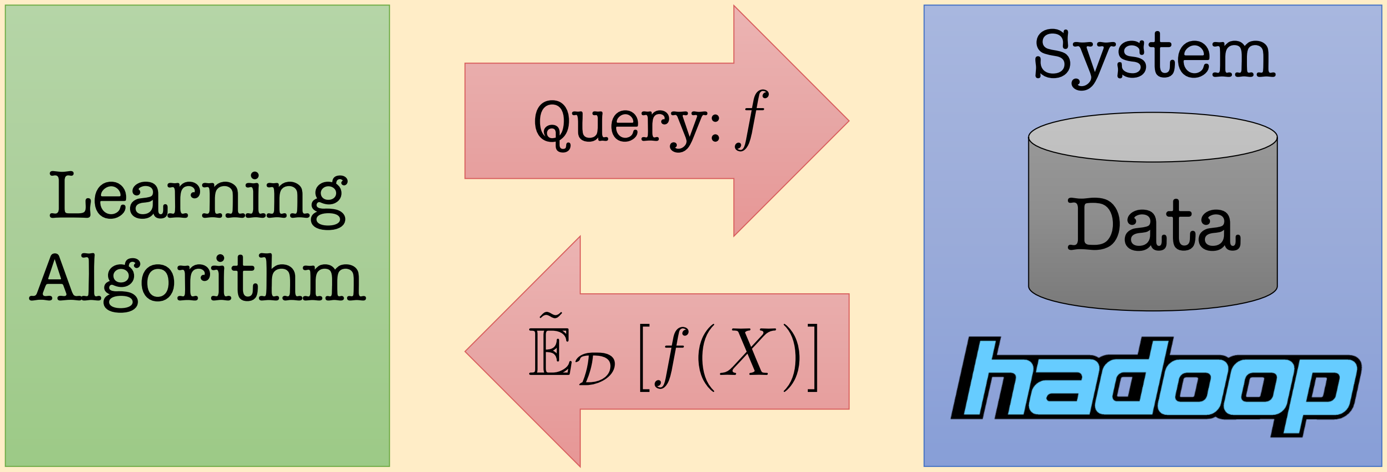


apple	1
the	3
cat	5
big	2
dog	2



If part of the file or any intermediate computation is lost we can simply **recompute it** without recomputing everything.

LEARNING FROM STATISTICS (AGGREGATION)*



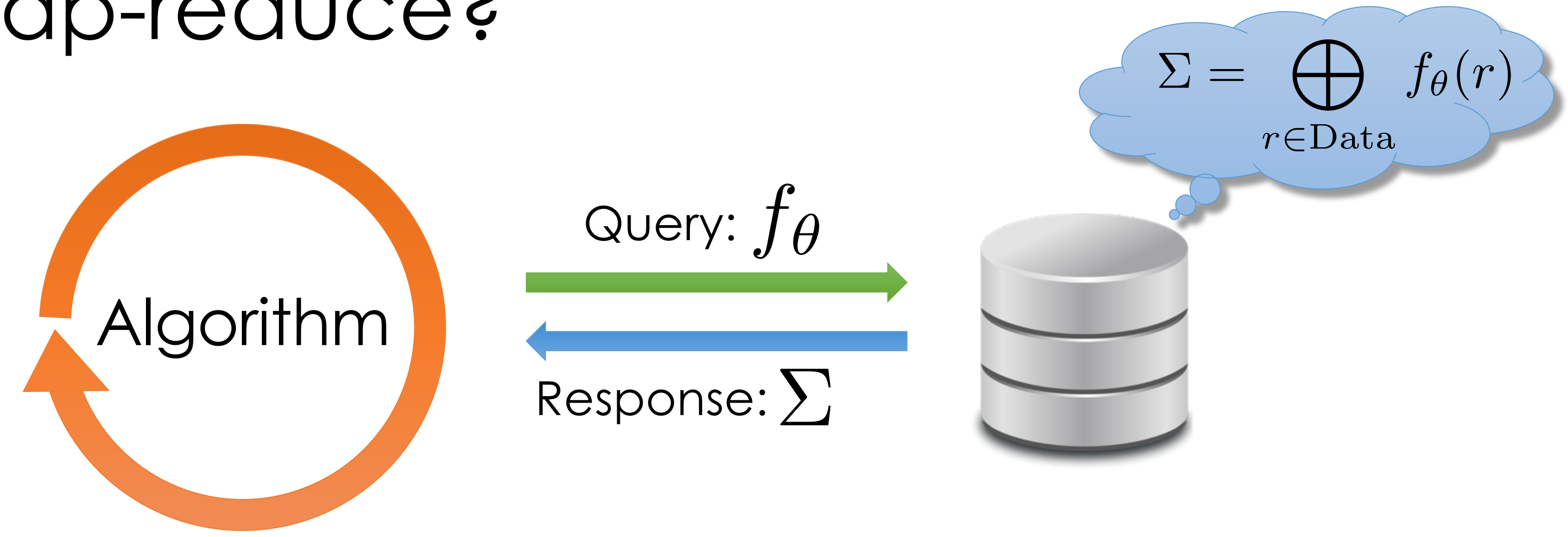
- D. Caragea et al., *A Framework for Learning from Distributed Data Using Sufficient Statistics and Its Application to Learning Decision Trees*. Int. J. Hybrid Intell. Syst. 2004
- Chu et al., *Map-Reduce for Machine Learning on Multicore*. NIPS'06.

*next set of slides are old!

Can we compute

$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

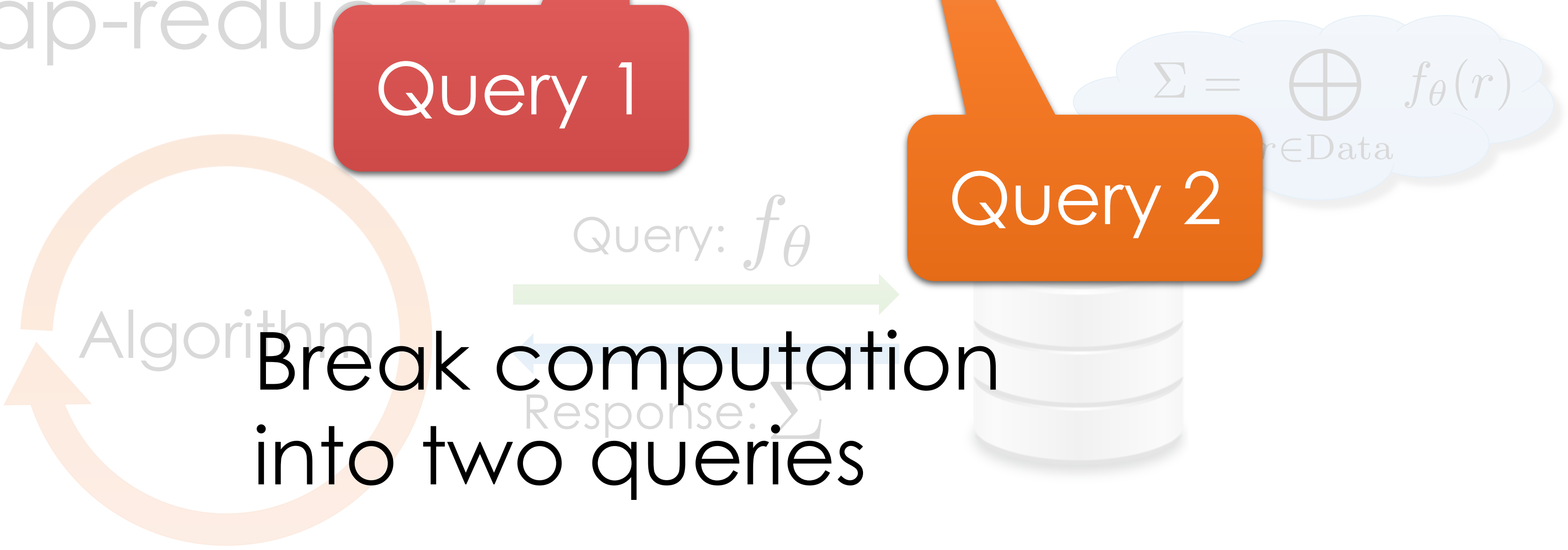
using the statistical query pattern in map-reduce?



Can we compute

$$\hat{\theta} = (\underline{X^T X})^{-1} \underline{X^T Y}$$

using the statistical query pattern in map-reduce?



Break computation into two queries

Cost Analysis

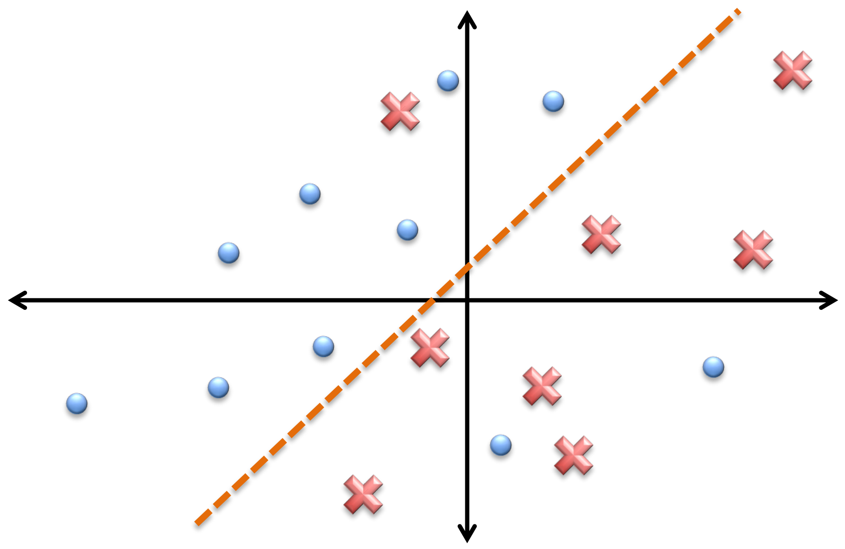
$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

Comutation	Cost
$A = X^T X$	$\Rightarrow O(np^2)$
$C = X^T Y$	$\Rightarrow O(np)$
$B = A^{-1}$	$\Rightarrow O(p^3)$
BC	$\Rightarrow O(p^2)$

When **n >> p** we want to distribute this computation

What about

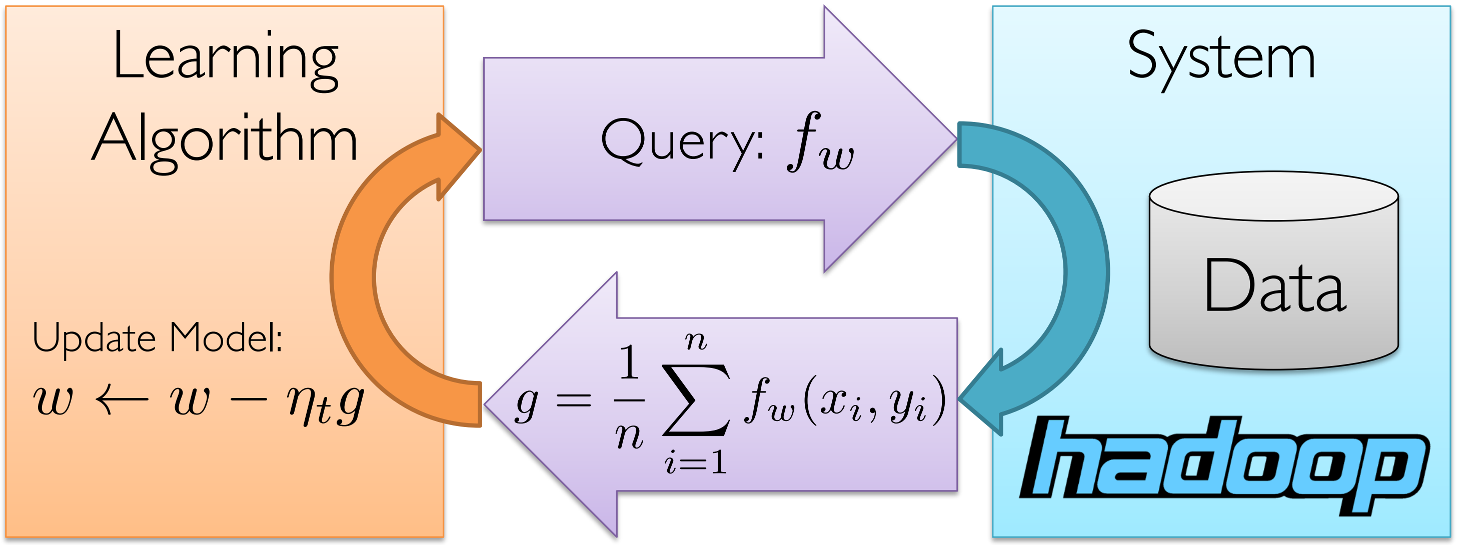
Logistic Regression using **Gradient Descent**?



Logistic Regression in Map-Reduce

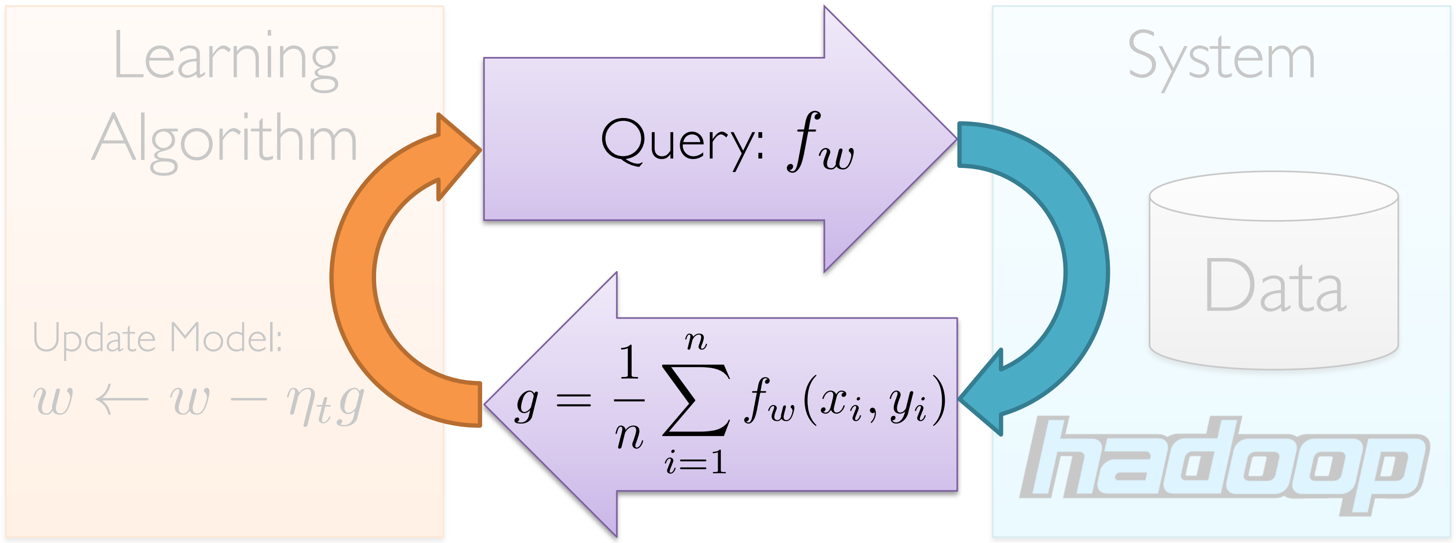
Gradient descent:

$$f_w(x,y) = \nabla \log L(y, h_w(x))$$

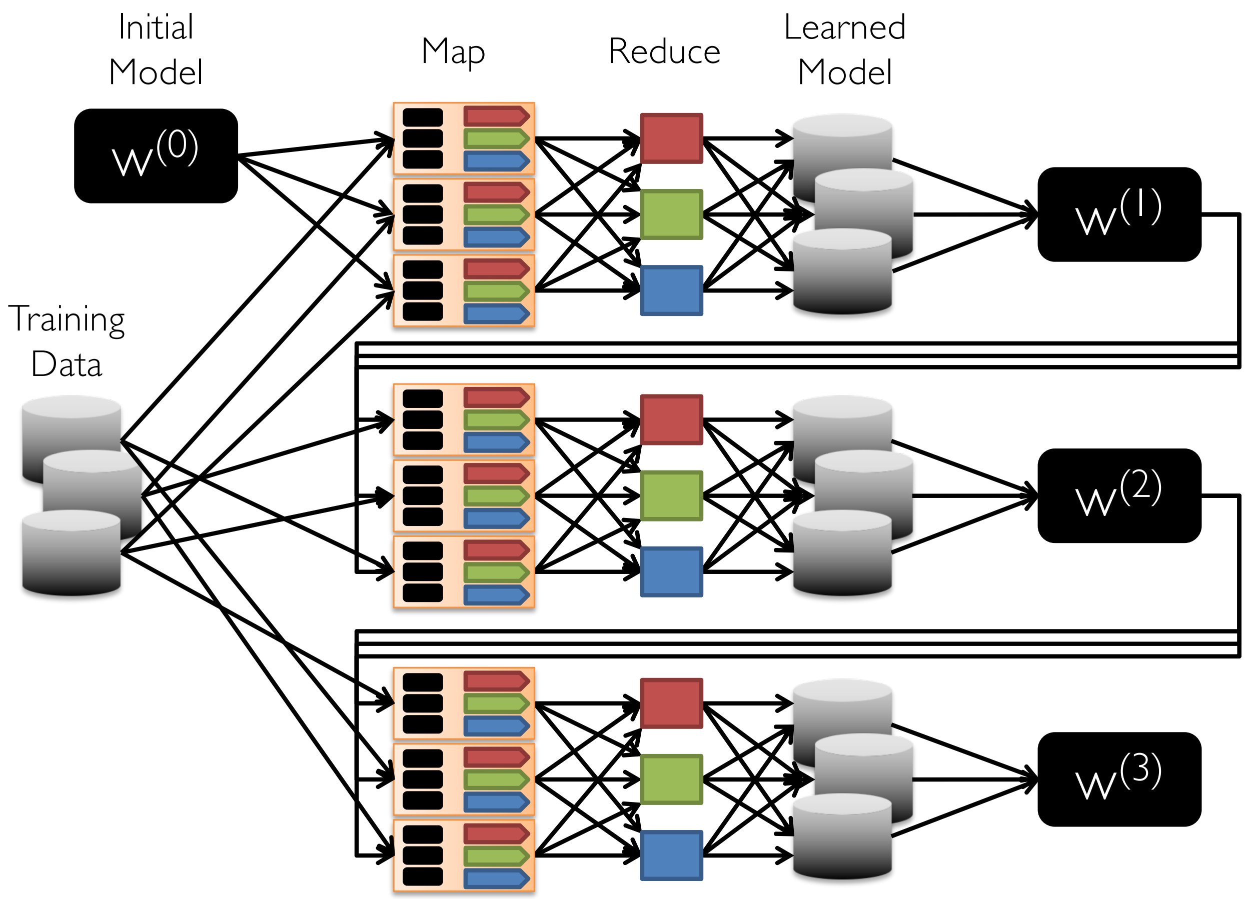


Learning Algorithm
Update Model:
$w \leftarrow w - \eta_t g$
Query: f_w
$g = \frac{1}{n}\sum_{i=1}^{n} f_w(x_i, y_i)$
System
Data
hadoop

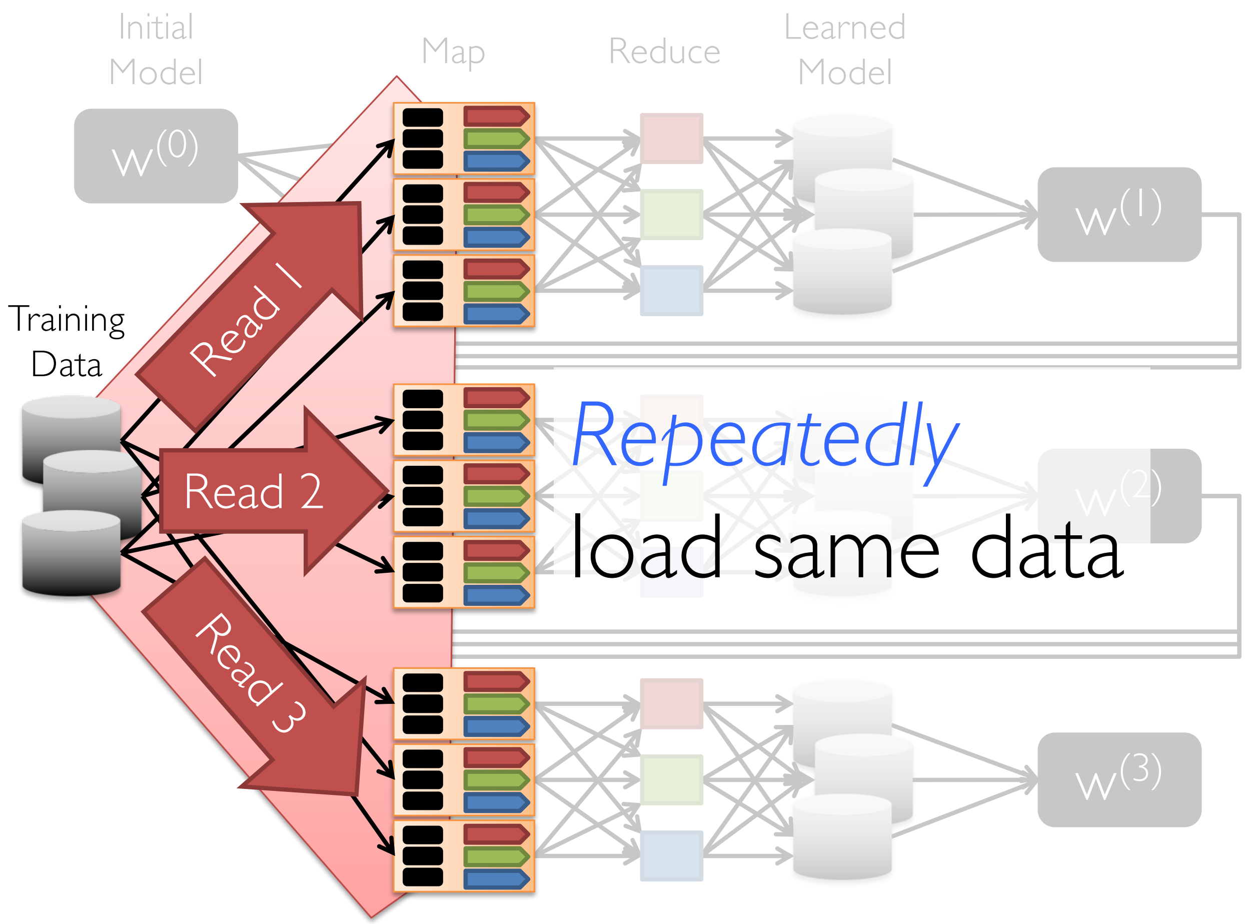
Map-Reduce is not optimized for iteration and multi-stage computation


Learning Algorithm
Update Model:
$w \leftarrow w - \eta_t g$
Query: f_w
$g = \frac{1}{n}\sum_{i=1}^{n} f_w(x_i, y_i)$
System
Data
hadoop

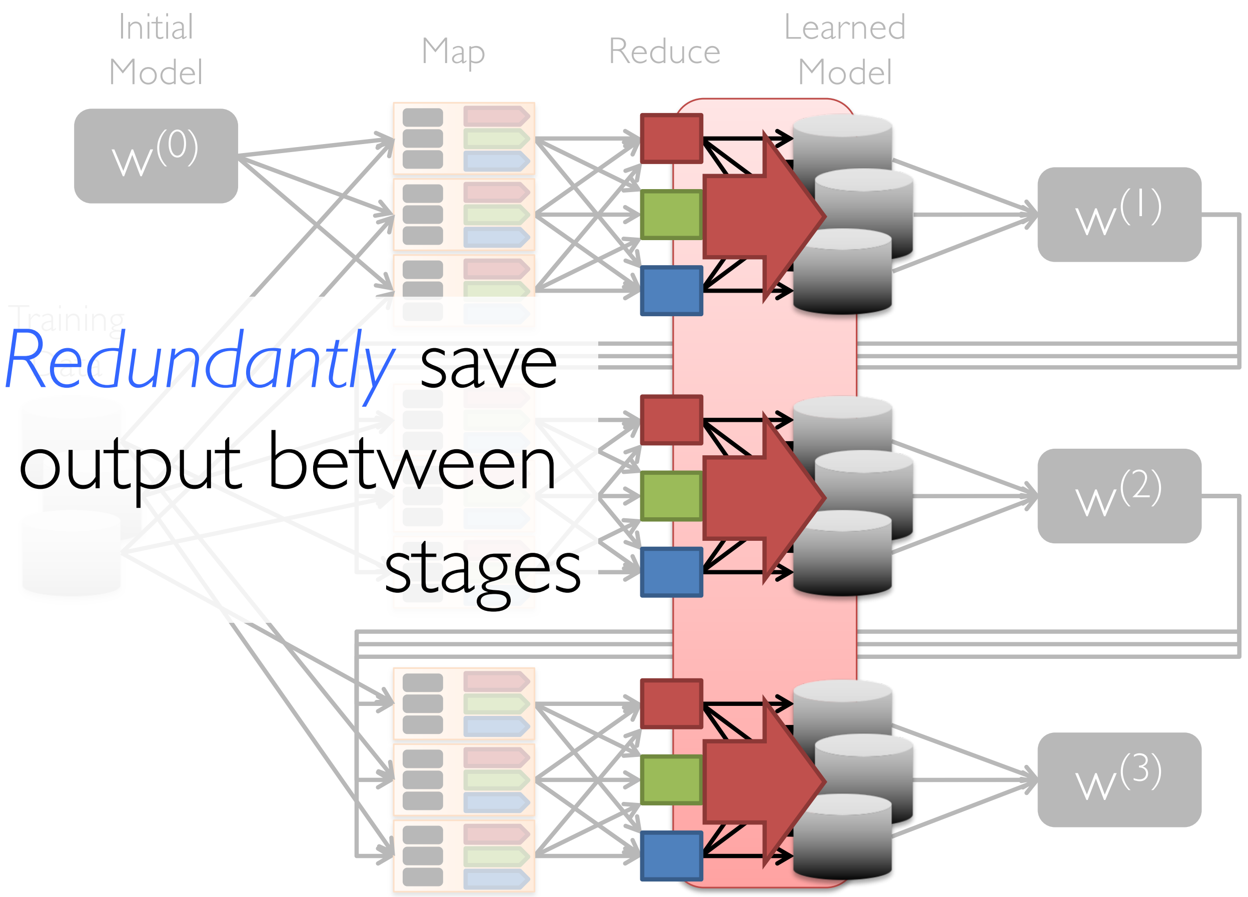
Iteration in Map-Reduce



Cost of Iteration in Map-Reduce



Cost of Iteration in Map-Reduce



Redundantly save output between stages

Spark

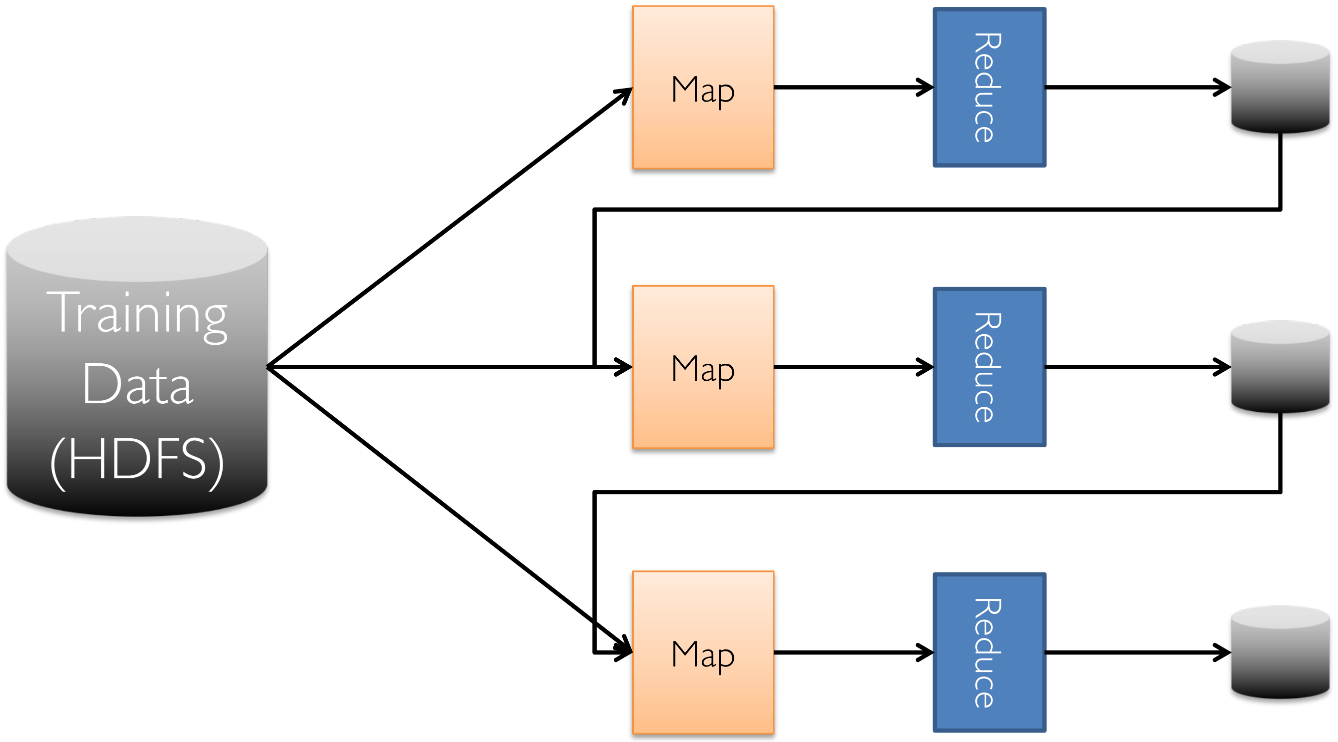
Iteration and Multi-stage computation

In-Memory Dataflow System

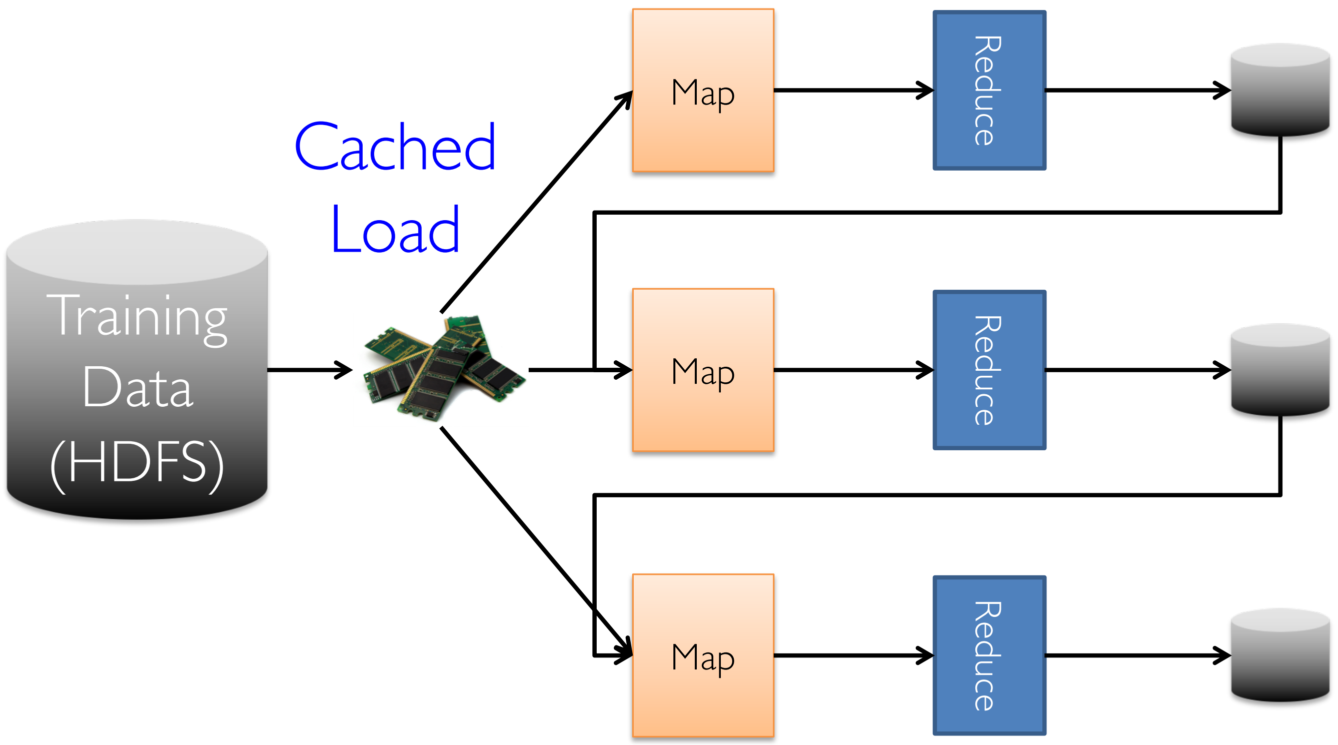
M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. *Spark: cluster computing with working sets.* HotCloud'10

M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica. *Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing*, NSDI 2012

Dataflow View

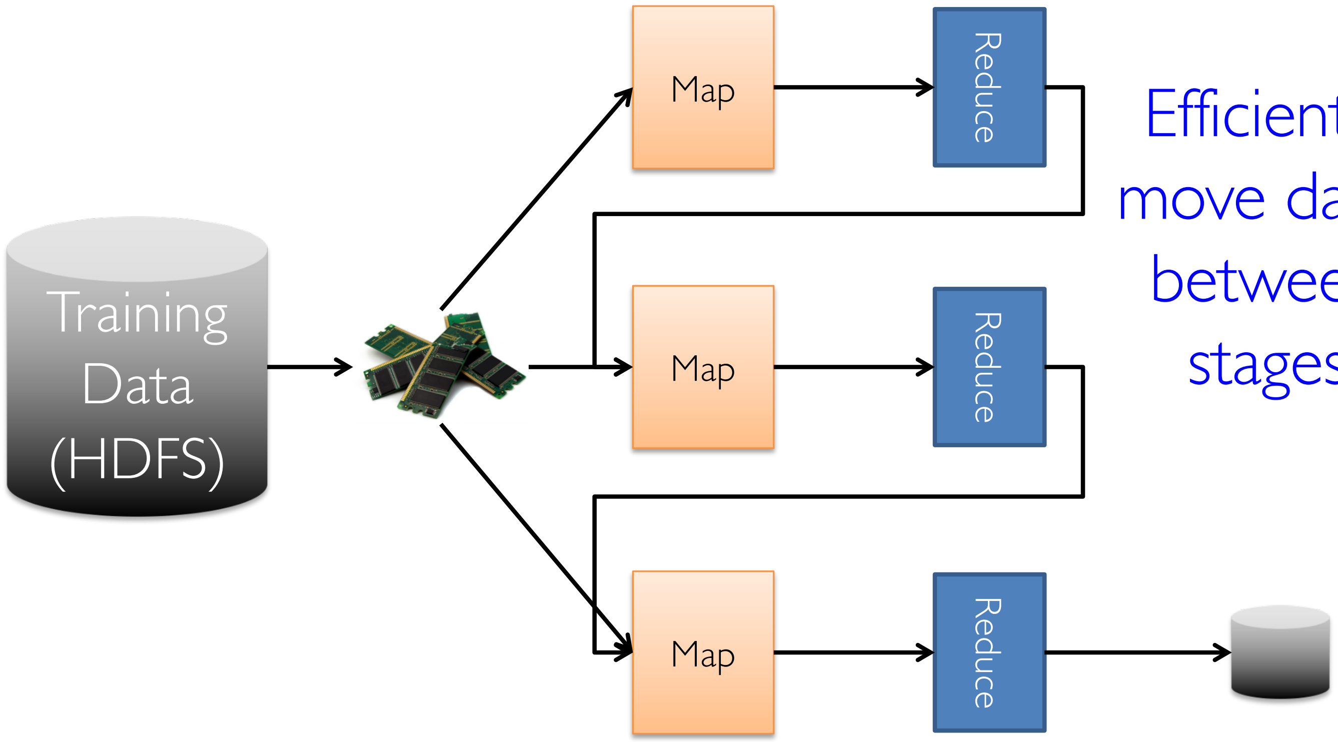


Memory Opt. Dataflow



10-100× faster than network and disk

Memory Opt. Dataflow View



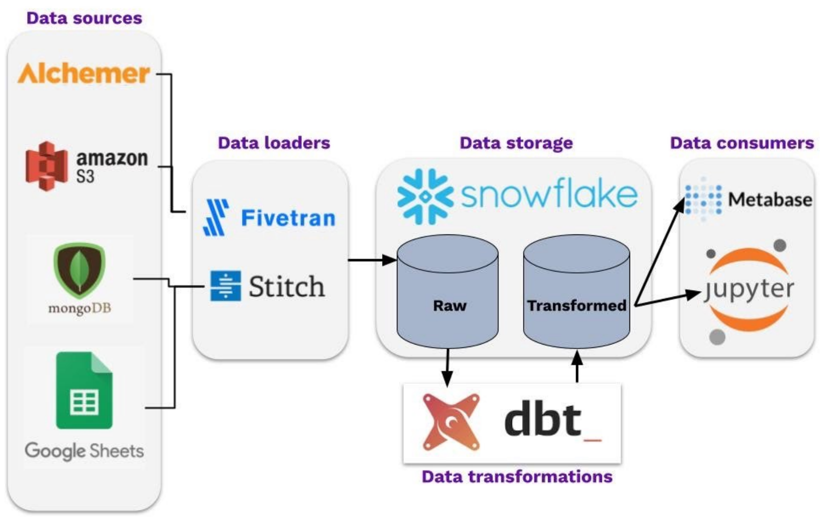
Efficiently move data between stages

Databricks Company

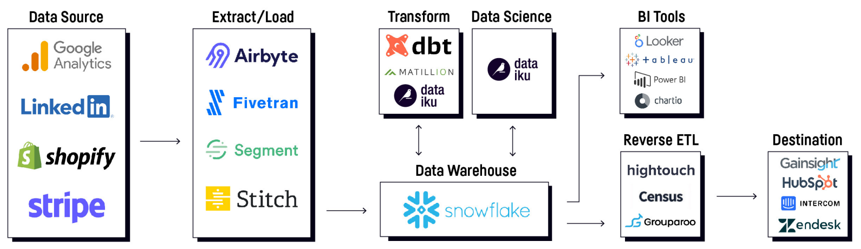
- Launched out of Berkeley and Spark Project
- Focused on data lake technology
- Competing with data warehouses:
  - **AWS Redshift:** probably most successful AWS service but structured and priced more like a traditional data warehouse
  - **Google BigQuery:** Very successful (and flexible) data warehouse system which separates storage from compute
  - **Snowflake:** Similar to BigQuery but cross cloud and slightly different pricing models

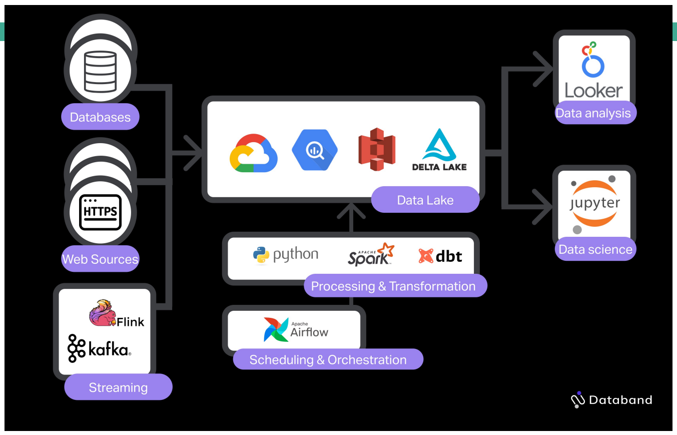
Modern Data Stack

- Great Blog Post: https://blog.getdbt.com/future-of-the-modern-data-stack/



The Modern Data Stack in the AI Era





Thoughts for Reading this Week?

- Older papers → What has changed in ML and Systems since these were written.
- Would you use any of these systems:
  - To support your research?
  - To build new systems?
- What are some biases in the author's perspectives?