CS 744: SUMMARY

Shivaram Venkataraman

Fall 2019
• Midterm 2 on Tuesday

• Poster session Dec 13\textsuperscript{th}, 3-5pm details on Piazza

• Final report Dec 17\textsuperscript{th}

• AEFIS Course feedback form!
Scalable Storage Systems

Computational Engines

Resource Management

Applications

- Machine Learning
- SQL
- Streaming
- Graph

TensorFlow

Spark

hadoop

Open Compute Project

Datacenter Architecture
OUTLINE

Unification vs Specialization

Survey results, Discussion

Big data systems: Looking forward
SPECIALIZATION VS UNIFICATION
GENERALITY: “ONE SIZE FITS ALL” DBMS

1970s
- Research prototypes: SystemR and INGRES
- Main function: OLTP

From 1990s
- Rise of business intelligence workloads
- OLAP workloads need to be isolated from OLTP
- Solution: Scrape data into data warehouses.
DBMS IMPLEMENTATION

Common Top Parser

Transactions

OLTP Bottom

Query language

Warehouse Bottom

Analytics
STREAM PROCESSING?

Example: Financial feed processing (Bloomberg, Reuters)
EXAMPLE WORKLOAD

Goals: Maximize message processing throughput on single machine

Scenario: Stock tick is late is if it occurs more than X secs from previous tick

Performance comparison:
- 2.8 GHz, 512 MB memory, single SCSI disk
  - 160,000 messages per second with StreamBase
  - 900 messages per second with DBMS
WHY IS IT SLOW?

DBMS: “Outbound” processing model

1. Insert data
2. Index data, commit transaction
3. Process query, return results

Process after store

First store data and then process the data.
WHY IS IT SLOW?

“Inbound” data processing

1. Push inputs into system
2. Process query
3. Return results
4. Optionally store (async)

Only way to do this in DBMS: Triggers
Not performant
OUTBOUND

“Pull” records given query
Store data, run any query

INBOUND

“Push” records into query
Store queries, pass data through
Is it just streaming?

Sensor Networks: TinyDB

Text Search: GFS / MapReduce

Scientific databases: SciDB

Data warehouses
  Column stores, read-oriented vs. write oriented
BIG DATA SYSTEMS

Unified systems

Naïve - Timely

Pylot / MapReduce

Batch

Stream

Graph

Specialized systems

TPU → ML inference workloads

PS → very large models

Weld? → Ray? - RL applications

API

API [functions, actors, etc.]

Power graph

Power, Raw graphs

SciKit Learn
BENEFITS

Unified systems

- Developer ease of use
- No need to stitch things together
- Additional workloads
- Hard to build
  - Complexity
- Perform

Specialized systems

- Performance!
- Simple code
- Industry specific
- Vendor choice?
IS IT JUST A CYCLE?
WHERE ARE WE IN THE CYCLE?

Dryad
CIEL

Oracle
PostgreSQL
hadoop
Spark

Dryad

2004 - 2011

2011 - 2015

Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)

Apache Spark

Hadoop
Mahout

2015 - now

TensorFlow
PyTorch
BOOTSTRAPPING UNIFIED SYSTEMS?

1. Implement a system/app/functionality that is superior to what is out there
2. Rapidly build an ecosystem providing additional functionalities

Example:

- Tensorflow initially target SGD/deep learning
- Unifies number of other features
  - `tf.data` supporting `map`, `flat_map` etc.
  - `tf.linalg` implementing linear algebra
  - `tf.sparse` for sparse data / shallow models
SURVEY RESULTS
LEARNING OBJECTIVES

At the end of the course you will be able to

• Explain the design and architecture of big data systems
• Compare, contrast and evaluate research papers
• Develop and deploy applications on existing frameworks
• Design, articulate and report new research ideas
DISCUSSION

https://forms.gle/sQFiAKwiQfHEKkPd8
What were some of your goals when you started the course? (Think about the first survey.) Reflect on what part of your goals have been achieved and how.

- Arch design patterns
- Historical lineage of why we use what we use.
- What are the metrics which matter?
- Critically evaluate, compare, and identify shortcomings.
- How to build such a system.
In the class, we discussed one trend across systems of unification vs. specialization. What are some other trends you have noticed across the papers in the class?

- Eager vs. lazy execution
- Latency vs. Tput vs. Correctness
- Fault tolerance
- Hardware vs. Commodity

Optimizations

- Stateful vs. Stateless
- More computation vs. less memory

Trade-off

Single point of failure - Centralized

Sync vs. Async

ML vs. Graph

Specialized - TPUs (7ms)
Open Source vs. Closed Source

- Design evolves
- Influenced
  - Data Proc.
  - Spark
  - TensorFlow
  - PyTorch
  - Flink

Fixed Design

- Storage
  - S3
  - GFS

Mutability vs. State

- Immutability vs. lineage

API Design

- High-level vs. low-level
- TF
- Keras
- Driver

Managing Computation or Not
LOOKING FORWARD
NEXT-GENERATION BIG DATA SYSTEMS?

Workloads

Data Processing Systems

Hardware
TRENDS IN WORKLOADS

New functionalities
- Data science / AI
- Robotics

New data sources
- Bio-medical data
- Video streams
- IoT / edge devices

Workloads
- RL
- Infrastructure

DIVERSITY ?

Sequence genomes
- MRI
- richer

HD videos
- larger & richer
Fairness in ML?
How robust is your system?

Adversarial examples
WHAT CAN SYSTEMS RESEARCH DO?

More than performance?
   Latency, throughput, efficiency
   Ease of use

Some other goals to consider?
   Security, Privacy
   Robustness
   Data bias / ethics
COURSE SUMMARY

Large scale data analysis has changed the world
COURSE SUMMARY

Applications
- Machine Learning
- SQL
- Streaming
- Graph

Computational Engines

Scalable Storage Systems

Resource Management

Your System Here?