CS33001: DATA-INTENSIVE COMPUTING SYSTEMS

Today
- Organizational Meeting
- Topics, Structure, Expectations
- Expected Coursework
- Ground Rules

Announcements
- Assignments for Next meeting (Friday)

LOGISTICS

Meetings:
- MF 130-250pm, Ry 277
- Format: Interactive Discussion (invited speakers, fostered paper discussions, project brainstorming/presentations/reviews)

Grading:
- In-class Participation (25%)
- Writeups (25%)
- Projects (50%)

Course Web:
OBJECTIVES

- Explore the technical challenges of data-intensive computing systems,
- including canonical driving problems, research systems, and emerging technologies.
- Develop a broad familiarity with the state of the art, including leading edge research in the area, and
- Hands-on experience with a range of systems which together provide a solid preparation for research in the area.

TOPICS: DATA-INTENSIVE COMPUTING SYSTEMS

- Application Archetypes and Major Infrastructures
- Storage, Traditional Filesystems, and Databases
- Big Data Computing Middleware
- Key-value Stores
- NoSQL Databases
- Background on Novel Storage Technologies (storage-class memory)
- Systems that integrate such Novel Tech into System Architectures
- Systems that integrate computing into the storage hierarchy
  + invited speakers
  + project discussions
COURSEWORK

Weekly:
• Read a set of technical papers, write summaries of the papers (3 paragraphs), write critiques of the papers – 3 good things and why, 3 weakness and why
• Lead a discussion of 1/Nth of the papers (N=# of students in class)
• Participate in a discussion of ALL of the papers
• Participate in a discussion of the larger research issues/ opportunities/challenges for that topic

Project:
• Using one of the identified DI-computing systems infrastructures, define and complete an innovative data-intensive computing systems project
  • "may (but need not) give an insight for an application, but must shed light on a data-intensive computing systems question"
  • Present and Document the Project in Compelling Fashion

DATA-INTENSIVE COMPUTING INFRASTRUCTURES

Encouraged:
• Presto/Blockus (my group w/ HP)
• Cleversafe (systems group + local startup)
• Graphlab/GraphChi (low-level graph computing engine)

Other possibilities:
• Staples of Data-intensive computing: Hadoop, VoltDB/HadoopDB, Cassandra, Memcached, MongoDB
• Other interesting infrastructures?

Challenge: Identify early and qualify robustness and capabilities
**BLOCKUS: SCALABLE “BIG DATA” ANALYTICS IN R**

Leverage rich, widely-adopted R Programming environment

- + simple parallelism extensions: Data parallel, versions, change-triggered execution

**Scale-out: demonstrated scalable performance on clusters (Presto)**

![Hadoop PageRank](image)

(See Eurosys ‘13)

**BLOCKUS: SCALING TO BIG COMPUTATIONS WITH NVRAM**

Idea: Use parallel R programs (Presto), and resulting partial execution order to scale up on data sets larger than the physical memory (i.e. 500GB data set on computer with 10GB memory)

- Exploit high bandwidth and low-latency of NVRAM
- Exploit workload and algorithm information
- Exploit structured parallelism

**Pursuing 10-100x improvements, on a large class of programs (Data Mining, Bioinformatics, Large graphs)**

- Automatic data movement;

Completing 1st generation NVRAM experiments
Cleversafe Company Info

- Privately held company, founded in 2004 at IIT by Chris Gladwin
- Sell software and/or hardware solutions to store data
- Based on the idea of Forward Error Correction applied to storage
- Customers needing to reliably store between 1 PB - 10 EB of storage at low cost
  - Example: Shutterfly - currently stores 30 PB
- Customers needing a very secure storage solution
  - Example: Intelligence Agencies foreign unsecure data centers
- Unique capability for multi-site deployments
The Math behind Information Dispersal

Reed-Solomon is Linear Algebra - solving a system of equations
Based on forward error correction
Compute some redundant information that can correct a future loss
Reed-Solomon as an Forward Error Correction (FEC) code
Perfectly efficient in storage space
Supports any desired fault tolerance
Example encoding, \( k=5, n=8 \)

\[
\begin{align*}
1a - 3b + 8c + 2d - 5e &= s_1 \\
4a + 1b - 9c + 6d - 2e &= s_2 \\
6a - 7b - 4c + 2d + 7e &= s_3 \\
2a + 2b - 3c + 1d - 6e &= s_4 \\
8a - 5b + 1c - 6d + 1e &= s_5 \\
5a - 6b + 5c - 2d + 2e &= s_6 \\
7a - 8b + 6c - 7d + 4e &= s_7 \\
2a - 2b + 3c - 4d + 3e &= s_8
\end{align*}
\]

- Solving for \( k \) variables
- Requires knowing any \( k \) results
- Therefore we can lose \((n-k)\) results
- Data overhead on the disk and on the wire is \((n/k)\)

How Dispersed Storage Technology Works

1. Source data is divided into slices
2. Slices are distributed to separate disks, storage nodes and geographic locations
3. A threshold number of slices are retrieved and the original content is regenerated
Security - Cleversafe

All or Nothing Transform applied to dispersal

Confidentiality – individual slices are useless
Integrity – data can be verified after reconstruction
Availability – threshold of slices reconstruct original

Scalability / Performance

• Allow building exabyte scale systems
• Achieved by assigning a namespace range to each server - no central index of data
• Each server stores data assigned to it on multiple disks
• 12 disks x 3 TB = 36 TB
• 84 disks x 4 TB = 336 TB
• Typical system has 100+ server nodes
• Related to scalability is performance – the larger the system, the higher the performance requirement
  – System scales horizontally – so as size grows, performance grows linearly
  – Requires no single point of contention for IO
Distributed Graph-Parallel Computation on Natural Graphs

Existing distributed graph computation systems perform poorly on Graphs, Natural Graphs, IO Systems.

Carnegie Mellon University
The **Graph-Parallel** Abstraction

- A user-defined **Vertex-Program** runs on each vertex
- **Graph** constrains **interaction** along edges
  - Using **messages** (e.g., Pregel [PODC’09, SIGMOD’10])
  - Through **shared state** (e.g., GraphLab [UAI’10, VLDB’12])
- **Parallelism**: run multiple vertex programs simultaneously

---

**PowerGraph**

**Program For This**

**Run on This**

- Split High-Degree vertices
- **New Abstraction** → **Equivalence** on Split Vertices
BACKGROUND READINGS

- Presto: Distributed Machine Learning and Graph Processing with Sparse Matrices, Eurosys 2013, April 2013.

READINGS

For Friday, 4/4


For Monday, 4/9 (Erik Bodzsar, Guest lecture)
Presto: https://sites.google.com/site/uchicagolssg/lssg/research/blockus

For Friday, 4/12
Data-Parallel
  - Map Reduce: http://research.google.com/archive/mapreduce.html

For Monday, 4/15 (Andrew Baptist, Guest Lecture)
Cleversafe
PROJECT ASSIGNMENT FOR NEXT WEEK (FRIDAY 4/12)

Identify a challenging data-intensive computing project and read up on it

- What defines it as a data-intensive computing project? (as opposed to something-else intensive)
- What are some of the unique technical challenges it represents? Systems challenges?
- What is the value of having all that data? Summaries? (there’s clearly a cost)
- What are some unique opportunities it represents? Where do the timeliness/quality/yield requirements come from?
- If significant improvements were possible? (speed/quality/cost) What if any new opportunities would it unlock?
- What computing infrastructure are they using? Is it efficient? Is it accessible?

Download, install, and run a data-intensive computing infrastructure

- A widely used one? (MongoDB, Hbase/H*, Graphlab, Cassandra)
- Or get started with Presto/Blockus or Cleversafe
- What is it capable of?
- What types of problems is it particularly well suited to? Intended workload?
- Does it scales? (in data? In speed/capability?) does it scale down?
- Robustness/Resilience of the system – hw/sw, operating point/usage, does it degrade or collapse?
- Recovery and Diagnosis – what can you recover in a failure? And what can you deduce about the cause of the failure?
- What kind of hardware was designed for? (clusters, HPC) – communication, reliability, system balance issues. Distribution?
- Is it efficient? (cost, energy, algorithmically, human effort)

GROUND RULES FOR THE COURSE

No “tourists” – come and come regularly

Active participation – come prepared, and come with something to say, and with questions to be answered

Push the envelope – beyond the questions framed in the papers, ideas in projects, to their logical extreme or conclusion

No “sacred cows” – any and all technical (and even ecosystem) topics can be opened and discussed (Andrew will shape discussion based on “productivity”)

April 1, 2013
CS33001 Chien Spring 2013