CS33001: DATA-INTENSIVE COMPUTING SYSTEMS SEMINAR

Today

- Presto discussion
- Data-intensive computing archetypes
- Data Parallel Data-intensive computing
  - Page Rank http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf
  - Map Reduce http://research.google.com/archive/mapreduce.html
- Example Data-intensive computing projects
- Systems (and RCC access)

Monday: Andrew Baptist, Cleversafe

- Julie Bellanca Cleversafe slides (Basics of AONT Security architecture)
- Jim Plank FAST05 tutorial “Erasure Codes and Storage”
  - What more you’d like to know/understand
- Short writeups on Data-intensive computing infrastructure

PRESTO/BLOCKUS

Summary – multicore+distributed parallelism to scale-up. Distributed, partitioned arrays as basis for parallelism. Focus was matrix operations, including graph problems formulated. Do it out of core. Scale to out of core.

3 Good

- **Fills need for large data sets in R (seems that all high level environments tend to note support scale well)**
- **Did get speedup; scaleup. And able to do dynamic load balance – but simple at this point.**
- **Use of shared memory good for space efficiency**

3 Bad

- **Took a long time to figure out the natural load balance.**
- **Ongoing overhead (lots of kinds). Garbage collection.**
- **Lots of other systems have introduced parallelism in this fashion**
- **Still dependence on the master node for scheduling (failures, scaling)**

DATA-INTENSIVE COMPUTING ARCHETYPES (PART I)

"Data Parallel"

- Large data set over which intensive computation happens
  - Similar to HPC, but Input data driven (not model driven), large input, small output
  - Examples: Netflix, Page rank, Walmart buying trends, etc.

"Tile, Sample, Sensor Integration"

- Large collection of smaller data samples, each of which requires processing and construction of a integrated view as a precursor to "data parallel"
  - Often partitionable into many tasks, executed over distributed data sets and resources, even samples over time. Image, spatial data processing.
  - Examples: Montage/EOSDIS, Google StreetView, Microsoft Streetside, Realtor websites, Traffic Maps

+ generated problems

+compositions of these

COMPUTING ARCHETYPES (PART II - TEMPORAL)

Data Parallel

Tile, Sample, Sensor integration

+ incremental update

+ real-time update

COMPUTING ARCHETYPES (PART III - CAPACITY)

Data Parallel

Tile, Sample, Sensor integration

+ incremental

+ real-time update

+ data set doesn't fit into memory

- Scaleout and Streaming versions
- Partition w/ database, in-database computation systems

DATA-INTENSIVE COMPUTING ARCHETYPES (PART IV)

Data Parallel

Tile, Sample, Sensor integration

+ incremental

+ real-time update

+ data set doesn't fit into memory

Scaleout and Streaming versions

Partition w/ database, in-database computation systems

+ naturally distributed, constrained to be distributed

TODAY'S READINGS

Page Rank http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

Map Reduce
http://research.google.com/archive/mapreduce.html

- programming model for distributed computing
- Map + reduce (design pattern), strict adherence
- => very large data size scaling, simple transparent fault tolerance, load balance.
- System pays for them, not the programmer

Classification? Data parallel + out of core

Summary?

3 GOOD

Pagerank

- **First major graph structure based ranking => required large-scale computation across the web graph**
- **Produced more robust results; elegance reduces to simple, well understood linear algebra problems**

Mapreduce

- **Transparent scaling, fault-tolerance, load imbalance**
- **Works well for embarrassingly parallel. But everything**
- **Tasks are idempotent (each phase is side-effect free)**
- **Centralized scheduler – good for scheduling, completion time, focuses where to invest for FT**

3 BAD

Pagerank

- **Unclear how they crawl? (sensitivity about completeness and "edges" – open graph)**
- **Unclear how to prevent manipulation (but better than keyword stuff)**
- **Popularity in links⇔ Rank (and that's not good)... Understand content, and search intent**

Mapreduce

- **Parallel elements can't communicate, awkward and inefficient in some cases**
- **Phase splitting makes complex data structures and algorithms very difficult to use**
- **Prevents any locality (streaming model to/from disk, a lot of work in every phase of the computation); can't easily carry forward partial results from phase to phase in a computation**
- **Centralized master – bad for scaling, bad for fault tolerance**

DISCUSSION

Data-intensive computing projects

- Netflix: Zach and Tanakorn
  - Recommenders - 2M reviews/day, 1-5 score, movie metadata
  - Matrix: subscribers x Movies, can derive new information about movies, but not users
  - Real-time update desirable; data small (10's of GB)
  - Computation is large N^3. Does additional data continue to help?
- EOSDIS: Aiman
  - 1TB/day, inherently distributed, search view based on indexing. Primary image data, metadata
  - Produce data products for further analysis; smooth fields
  - Global vs. Urban focus
- Graph Formulations of Tiling: Max
  - Tiling <-> graph algorithms, don't have locality properties
  - Grow exponentially; not a huge variaance in node degree
- Massively distributed data distribution: Yuan
  - Avalanche – bittorent improvement; coding is symmetric
  - Is computational effort to encode the critical bottleneck?
- Tbd: Matt

DICSYSTEMS PROJECT INFRASTRUCTURE

What might you need?

What do you have access to?

Cleversafe

LSSG cluster (100 cores)

RCC cluster (2000 cores)

Amazon EC2

SUMMARY

Data Intensive computing archetypes

Data-parallel – mapreduce and pagerank

Data-intensive computing projects

Next time: Erasure codes, Andrew Baptist. Come with interesting questions!

BACKUP

PROJECT ASSIGNMENT (MONDAY 4/15)

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/capabilty?) 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)

CANDIDATES

HBASE/H*, VoltDB
PIG/H*
HadoopDB/H*
Cassandra
SciDB
BLOOM/MR Online/?
MongoDB
Graphlab/Graphchi
Swift
?

Preference: something new

April 12, 2013
CS33001 Chien Spring 2013
15

ASSIGNMENT TURN-IN FORMAT (4/15)

Output: 4 slide summary
- Which & why
- What you did
- Answer to Q's

1-page writeup describing system and its capabilities

5-minute presentation in class using 4 slides – summarize capabilities and your experience with it (what you did)

- For each, we'll have a discussion on what its being used for
- What its good at
- What are its shortcomings
- What kinds of projects it might be suitable for

April 12, 2013
CS33001 Chien Spring 2013
16