HiTune: Dataflow-Based Performance Analysis for Big Data Cloud

Bo Huang
Principal Engineer
Intel APAC R&D Ltd.
Aug 16, 2011
Big Data

• “Industrial Revolution of Data”
  – The heartbeat of mobile, cloud and social computing
  – Expanding faster than Moore’s law
    • E.g., Internet of Things

• What is Big Data?
  – Too large to work with using traditional tools
  – Require a new architecture
    • Massively parallel software running on 100s~1000s of servers
Dataflow Model for Big Data Analytics

- **Users**
  - Applications modeled as dataflow graphs
  - Write subroutines running on the vertices
  - Abstracted away from messy details of distributed computing

- **System runtime**
  - Dynamically map dataflow graphs to the cluster
  - Handles all the low level details
    - Data partitioning, task distribution, load balancing, node communications, fault tolerance, ...
What Worked

- **Parallel programming is hard**
  - Distributed programming is harder
  - Dataflow model makes it a lot easier

- **An appropriately high level of abstraction**
  - User required to consider data parallelisms exposed by the dataflow
  - Runtime distributes executions of subroutines by exploiting data dependencies encoded in the dataflow

---

Nontrivial software written with *threads*, *semaphores*, and *mutexes* are incomprehensible to humans.

Edward A. Lee
CGO 2007, March 2007

Auto-Partitioning Compiler for Intel Network Processor (IXP)
What Didn’t Work

• **Dataflow abstraction makes Big Data system appear as a “black box”**
  – Very difficult for a user to understand runtime behaviors
  – Performance analysis & tuning remains a big challenge

• **Key challenges of performance analysis for Big Data**
  – Massively distributed system
    • How to correlate concurrent performance activities (across 10000s of programs and machines)?
  – High level dataflow abstraction
    • How to relate low level performance activities to high level dataflow model?
HiTune: “Vtune for Hadoop”

- **Distributed instrumentations**
  - Lightweight sampling using binary instrumentation
    - No source code modifications
  - Implemented using Java programming language agents
    - Generic sampling information collected

- **Dataflow-driven analysis**
  - Re-constructing dataflow execution process using low level sampling information
    - Based on a dataflow specification
  - Implemented as several Hadoop jobs
HiTune 0.9

Status

- Used intensively both inside Intel and by several external customers
- Open sourced under Apache License 2.0
- Available at https://github.com/hitune/hitune

```
Instrumentation

Local HiTune data
Adaptor
Adaptor
Adaptor
Chukwa Agent
Chukwa Collector
Chukwa Collector
Chukwa Collector

Aggregation

HDFS
HiTune Report (.csv)
PostProcess
Chukwa Demux
HiTune Paser
HiTune Paser

Analysis

Hadoop Job
Hadoop Job
Hive Query
HiveQL
Hive
HiTune Report (.csv)
Excel Spreadsheet
Visual Report Samples (.xlsm)
```
Overhead

- Ratio of instrumented vs. uninstrumented clusters
  - Less than 2% runtime overhead due to instrumentation
The Hadoop Dataflow Model

Partitioned Input

Map Tasks

Reduce Tasks

Streaming dataflow

Sequential dataflow
Case Study: Limitation of Traditional Tools

• Sorting many small files (3200 500KB-sized files) using Hadoop 0.20.1
  – *Cluster very lightly utilized (extremely low CPU, disk I/O and network utilization)*
  – No obvious bottlenecks or hotspots in the cluster
  – Traditional tools (e.g., system monitors and program profilers) fail to reveal the root cause
Case Study: Limitation of Traditional Tools

- HiTune results (dataflow execution) reveal the root cause
  - Upgrading to “Fair Scheduler 2.0” fixes the issue

**Dataflow Execution Chart**

- **Map/Reduce Tasks**
  - bootstrap
  - map
  - shuffle
  - sort
  - reduce
  - idle

**The Low Utilization Issue**

**The Fix**
Case Study: Limitation of Hadoop Logs

• **TeraSort**
  - *Large gap between end of map and end of shuffle*
    - None of CPU, disk I/O and network bandwidth are bottlenecked during the gap
  - “*Shuffle Fetchers Busy Percent*” metric reported by Hadoop is always 100%
    - Increasing the number of copier threads brings no improvement
  - Traditional tools or Hadoop logs fail to reveal the root cause
Case Study: Limitation of Hadoop Logs

- HiTune results (dataflow-based hotspot breakdown) reveal the root cause
  - *Copier* threads idle 80% of the time, waiting for *memory merge* thread
  - *memory merge* thread busy mostly due to compression

- Changing compression codec to LZO fixes this issue
Case Study: Extensibility

- Easily extended to support Hive
  - Simply changing the dataflow specification

- Aggregation query in Hive performance benchmarks
  - 68% of time spent on data input/output, Hadoop/Hive initialization & cleanup
  - Critical to reduce intermediate results, improve data input/output, and reduce Hadoop/Hive overheads
Summary

• HiTune - “VTune for Hadoop”
  – Better insights on Hadoop runtime behaviors
    • Dataflow-based analysis
  – Extremely low runtime overheads
  – Very good scalability & extensibility
  – v0.9 open sourced under Apache License 2.0
    • See https://github.com/hitune/hitune

* Refer to our USENIX ATC’11 paper for more details
Amazing things happen with Intel inside®