MOCHI: VISUAL LOG-ANALYSIS BASED TOOLS FOR DEBUGGING HADOOP
Jiaqi Tan, Xinghao Pan, Saida Kavulya, Rajeev Gandhi, Priya Narasimhan (Carnegie Mellon University)

INTRODUCTION

- MapReduce: Framework for distributed, parallel programming on commodity clusters
  - Programs defined as Map and Reduce functions
  - Multiple copies run in parallel on data segments
- Current tools for performance debugging:
  - Designed for non-distributed programs
  - Too much information when debugging MapReduce programs
  - Do not expose MapReduce abstractions, automated framework behaviors
- System Logs
  - Ubiquitous but hard to process automatically
- Need to correlate across nodes for MapReduce

APPROACH

- Extract state-machine view of each node’s execution from its logs [Tan et al., USENIX WASL 08]
  - Distributed control-flow view (MapReduce execution layer)
  - Distributed data-flow view (Distributed Filesystem layer)
- Correlate state-machine views of each node
  - Across nodes
  - Across execution and distributed filesystem layers
- Extract conjoined data + control causal flows (Realized Execution Paths)

VISUALIZATIONS OF MAPREDUCE

- “Swimlanes”: Task-execution in time, across nodes
  - Horizontal-axis: Time elapsed;
  - Vertical-axis: One tick for each task
  - Each task: horizontal line for its duration
  - Blue: Maps; Green: Reduces
- Top graph sorted by node; Bottom by start-time
- MIROS (Map Inputs, Reduce Outputs, Shuffles)
- Data transfers to/from hosts during phases (Maps, Shuffles, Reduces)
- Aggregate data volumes transferred across all tasks on each node for entire job
- Color intensity shows sizes of data transfers

CASE STUDY: YAHOO! M45 USER JOBS

- Performance debugging case-studies on CMU user workloads on Yahoo! M45 cluster
  - Matrix-Vector Multiplication (left) [sorted by node]
    - Before: Some nodes idle during reduce
    - After: All nodes ran reducers
  - Sleep/No-op Benchmark (right) [sorted by start-time]
    - Before: Some reduces – unusually long durations, due to JVM-IPv6 miconfiguration
    - After: Completed 50% faster

Conjoined Control-flow + Data-flow:
Job-Centric Data-Flow (JCDF)
(across all nodes)

Causal flows of data and processing:
Realized Execution Paths (REP)

Execution (MapReduce) layer state-machine view (per-node)
Distributed FileSystem (HDFS) state-machine view (per-node)
Space-time View
Time-volume View
Space-volume View
Time-volume View