Debugging of Hadoop Production Errors Using Open Source Flow Analyzer – Semiconductor Industry Case Study

Reach the community at users@collaborate.jumbune.org

Download Jumbune from http://www.jumbune.org

An Open Source initiative LGPLv3 licensed

Table of Contents

Overview

A renowned manufacturer of microprocessors and GPUs, depends on data generated during manufacturing to improve upon its processors, manufacturing engineering processes, detect flaws during fabrication and also determine when a specific step in one of its manufacturing processes starts to deviate from normal tolerances. This data can include logs from the fabrication which contains the temperature, stage and wafer details and also logs from the multi-tier tests that are run during each phase of the manufacturing process.

Business Challenge and Future Proofing

The company traditionally used RDBMS to store all the current and historical data. As the data grew, as the costs associated with running analytics and maintenance of the database started multiplying the company decided to migrate to Apache Hadoop system to create an enterprise data lake on HDFS (Hadoop Distributed File System) for faster analytics. All the data from the Wafer processing, Die preparation to IC packaging and IC testing data would be pushed on to the HDFS enterprise data lake. One of the initial challenge faced by the data engineering team was partitioning data into their respective departments and/ sections for faster querying and accessibility. The team decided to go ahead with MapReduce for implementing the partitioning system.

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Use Case

The various environments in the organization have heterogeneous Hadoop setups, Development and QA use Hadoop 2.4.1 Yarn, whereas Production environment has a commercial distribution of Hadoop. A system was put in place to partition data on the HDFS according to the source, type and ingestion period and was running smoothly till the QA team found some inconsistencies in the partitioned data. During the semiconductor device fabrication once the front-end process has been completed, the semiconductor devices are subjected to a variety of electrical tests to determine if they function properly. The fab tests the chips on the wafer with an electronic tester that presses tiny probes against the chip. The machine marks each bad chip with a drop of dye and is logged into a central repository and chips are sorted into virtual bins according to predetermined test limits. The resulting data can be graphed, or logged, on a wafer map to trace manufacturing defects and mark bad chips. This map can also be used during wafer assembly and packaging. Chips are also tested again after packaging and logged into the repository again. All the respective log and result data were dumped onto the HDFS data lake for further analysis.

The data from the test phases were categorized according to their source, result and environment for pattern recognition. The QA team found that there were a lot of inconsistencies in the test data from the wafer phase and the packaging phase. Large number of the wafers that had passed the failure threshold during fabrication was found to be unresponsive during the later phases, this lead to a lot of effort being spent on finding out the source of the issue. Multiple failure points such as the test probe, test bed, the MapReduce system that sorts the data and the final quality checks were

analyzed to zero in on the source of the discrepancy. After spending lot of man hours testing the test equipment and the different phases of manufacturing it was concluded that the discrepancies were instigated during the MapReduce phase.

MapReduce Debugging using Jumbune's Debugger

MapReduce debugging is a tiresome but unavoidable routine for any Hadoop developer. An effective debugger can make programmers more productive by allowing them to dedicate more time in implementing use cases. Debugger should allow the programmer to inspect the code at the site of an anomaly and trace back in the program stack to derive more clues about the cause of the problem.

- A successful debugger should collect data about billions of key value pairs that go through each component, phase and control block of the MapReduce application.
- Such an extensive system should be frugal, scalable and yet detailed.
    - It must be frugal that so that the instrumentation does not escalate the execution time of the code.
    - It must also collect information that is detailed enough to let the developer understand the bottlenecks and faults in their program.
    - The debugging system must also scale to large data sets which would accurately mimic all the use cases that the developers have in mind.

Jumbune's MapReduce debugger is a unique solution that gives the cluster level correlation of data with execution flow inside the code. It can bring down hours of execution logic debugging trials by identification of root causes within minutes. User may apply regex validations or user defined validation classes. As per the applied validation, Flow Debugger checks the flow of input data tuples essentially <*key, value*> pair data for each mapper and reducer in the submitted job.

- Jumbune provides a comprehensive table/chart view depicting the flow of input records through the job.
- The flow is displayed at job level, MapReduce level, and instance level.
- Unmatched keys/values represent the number of unexpected flow of key/value data through the job.
- Debugger drills down into the code to examine the flow of data for various counters like loops and if-conditions, else-if, etc.

The team with the help of Jumbune's flow debugger team created custom validation classes in java for each test and with this they were able to correlate the number of tests that passed and to the actual result, was able to detect that during the partitioning of data in the MapReduce phase, the pass threshold for a certain critical test was interpreted in a wrong manner which lead to a lot of valid wafer data being diagnosed as faulty.

Impact: Zero issues in Production

Production errors and failures results in significant loss in revenue and time, enterprise soliutions have zero tolerence to them. Jumbune's Debugger has helped this organisation to get rid of production issues all tougether.