Anomaly Detection Framework with Chukwa

Introduction

We describe a general framework for implementing algorithms for detecting anomalies in systems (Hadoop or otherwise) being monitored by Chukwa, by using the data collected by the Chukwa framework, as well as for visualizing the outcomes of these algorithms. We envision that anomaly detection algorithms for the Chukwa-monitored clusters can be most naturally implemented as described here.

The types of operations that this framework would enable fall in these broad categories:

1. Performing anomaly detection on collected system data (metrics, logs) to identify system elements (nodes, jobs, tasks) that are anomalous,
2. Applying higher-level processing on collected system data to generate abstract views of the monitored system that synthesize multiple viewpoints, and
3. Applying higher-level processing on anomaly detection output to generate secondary anomaly detection,
4. Presenting and/or visualizing the outcomes of the above steps.

Design

The tasks described above will be performed in a PostProcess stage which occurs after the Demux. These tasks will take as their inputs the output of the Demux stage, and generate as their outputs (i) anomalous system elements, (ii) abstract system views, or (iii) visualizable data (e.g. raw datapoints to be fed into visualization widgets). These tasks will be MapReduce or Pig jobs, and Chukwa would manage these tasks by accepting a list of MapReduce and /or Pig jobs, and these jobs would form the anomaly detection workflow.

In keeping with the consistency of the Chukwa architecture, these jobs in the anomaly detection workflow would have to accept SequenceFiles of ChukwaRecords as their inputs, and would generate SequenceFiles of ChukwaRecords as their outputs.

Finally, the outputs of these tasks would be fed into HICC for visualization. The current approach would be to use the MDL (Metrics Data Loader) to load the data to an RDBMS of choice which can be read by HICC widgets.

Hence, the overall workflow of the anomaly detection would be as follows:

1. MapReduce/Pig job processes post-Demux output to generate abstract view and/or anomaly detection output "(these would be scheduled by the PostProcessor, the PostProcessor would serve as the entry-point to the execution logic of the anomaly detection framework)"
2. (Optional) Additional MapReduce/Pig job processes abstract views/anomaly detection output to generate secondary anomaly detection output
3. Data fed into HICC via an RDBMS
4. HICC widget loads anomaly detection/abstract view data from RDBMS for visualization

Implementation

Hadoop Anomaly Detection and Behavioral Visualization (Fingerpointing)

Current active developments for the Chukwa Anomaly Detection Framework are for detecting anomalies in Hadoop based on the following tools/concepts from the CMU Fingerpointing project:

1. SALSA for State-machine extraction of Hadoop's behavior from its logs
2. SALSA for Hadoop task-based anomaly detection using Hadoop's logs
3. Mochi for visualization of Hadoop's behavior (Swimlanes plots, MIROS heatmaps of aggregate data-flow) and extraction of causal job-centric data-flows (JCDF)
4. Ganesha for node-based anomaly detection using OS-collected black-box metrics

The `FSMBuilder` component implements SALSA state-machine extraction, and is a MapReduce job which reads SequenceFiles of ChukwaRecords and outputs SequenceFiles of ChukwaRecords, with each ChukwaRecord storing a single state. We describe the workflows for some of the tools below:

Swimlanes Visualization

This visualization shows the detailed task-level progress of MapReduce jobs across nodes in the cluster.

1. (`FSMBuilder` MapReduce job, available soon) SALSA is used to extract state-machine views of Hadoop's execution - uses post-Demux output; uses `JobData/JobHistory`
2. State-machine data from `FSMBuilder` is loaded into RDBMS using MDL

3. Raw state-machine views visualized using Swimlanes visualization HICC widget which reads data from RDBMS

MIROS (N x N heatmaps) Visualization

This visualization shows the aggregate data-flows across DataNodes in an HDFS instance.

1. (`FSMBuilder`, available soon) SALSA is used to extract state-machine views of Hadoop's execution - uses post-Demux output; uses `ClientTraceDetailed` (CHUKWA-282)
2. (MapReduce job to aggregate HDFS activity across states) Process states as seen from state-machine view to generate aggregate counts of HDFS activity
3. Aggregate activity data is loaded into RDBMS using MDL
4. Visualization of heatmaps using HICC widget

Task-based Anomaly Detection

1. (`FSMBuilder`, available soon) SALSA is used to extract state-machine views of Hadoop's execution - uses post-Demux output; uses `JobData/JobHistory`
2. (MapReduce job) Collect states from state-machine view and process them to generate list of anomalous nodes, possibly list of anomalous nodes per unit time for incremental/online diagnosis of anomalies (described in Section 8 of SALSA)
3. Load anomaly data into RDBMS using MDL
4. Visualization of heatmaps using HICC widget

Usage

Hadoop Anomaly Detection and Behavioral Visualization (Fingerpointing)

Swimlanes Visualization

This visualization is generated from the output of the Demux operation. The steps (mostly envisioned to be automated) involved in generating the visualization are:

1. Generate state-machine views using `FSMBuilder` (Currently unavailable, pending feature additions to the PostProcessor to support non-MDL tasks): Read post-Demux data (SequenceFiles of ChukwaRecords of JobData data) as input, write state-machine view as SequenceFiles of ChukwaRecords of states. (Unsupported at time of writing, but will be available soon)
2. Load states into database using MDL (CHUKWA-279)
3. Load Swimlanes widget in HICC (CHUKWA-279)