NoStop: A Novel Configuration Optimization Scheme for Spark Streaming

Qianwen Ye, Wuji Liu, and Chase Wu

Department of Computer Science
Ying Wu College of Computing
New Jersey Institute of Technology
Outline

• Introduction and Motivation

• Related Work

• Problem Statement

• Design of NoStop

• Performance Evaluation
Spark Streaming Model

Spark Streaming receives real-time input data streams and divides the data into multiple batches before passing them to Spark processing engine.

System stability and end-to-end latency are among the utmost important performance metrics for streaming data processing.
The Goal

• For a given Spark Streaming application with a fluctuating input data rate, our goal is to dynamically and adaptively determine the most suitable system configuration to achieve minimum end-to-end delay while maintaining system stability.
Outline

- Introduction and Motivation
- Related Work
- Problem Statement
- Design of NoStop
- Performance Evaluation
Related Work

• Most of the existing efforts for performance optimization of Spark Streaming consider cases that require a constant input data rate or the ability to scale up cluster resources on demand.

• Some efforts apply machine learning-based approaches, which require a sufficient amount of historical data for training.
Outline

• Introduction and Motivation

• Related Work

• Problem Statement

• Design of NoStop

• Performance Evaluation
Batch Processing Time and Batch Interval

Batch Interval < Batch Processing Time: Unstable System
Batch Interval > Batch Processing Time: Long end-to-end latency
Batch Interval = Batch Processing Time: Ideal Case
Problem Description

Given a Spark Streaming application executed in a distributed computing environment and an input data stream arriving at a varying speed, we wish to find a proper setting for batch interval and number of executors in real time to achieve minimum end-to-end delay:

\[
\text{argmin} \quad \text{End-to-end Delay,}
\]

\[
\text{Batch Interval, Number of Executors}
\]

subject to the following constraint:

\[
\text{Batch Interval} \geq \text{Batch Processing Time},
\]

where the constraint guarantees system stability.
Outline

• Introduction and Motivation

• Related Work

• Problem Statement

• Design of NoStop

• Performance Evaluation
Design Goals

• Noise Tolerance: account for the randomness existing in the dynamics of streaming data processing in distributed environments

• Generality: be applicable to different types of Spark Streaming applications executed in different computing environments

• Efficiency: converge to the minimum end-to-end delay promptly for fluctuating input data with a negligible overhead

• Performance Guarantee: provide a theoretically-proved performance bound
NoStop Architecture

Kafka Broker

Timing Configuration

Report Status

Spark Streaming

Configuration

Listener

HDFS

Database
Simultaneous Perturbation Stochastic Approximation (SPSA)-Based Optimization Algorithm

• Rationale on the use of SPSA
  – Does not require an explicit analytical form of end-to-end delay
  – Does not require any additional information about system dynamics, applications and input distribution
  – Low time complexity
  – Does not require a large amount of historical data
  – Performance is theoretically guaranteed
SPSA-Based Optimization Algorithm

Let $\theta$ denotes the control parameters, $G(\theta)$ denotes the end-to-end delay.

The standard stochastic approximation form is given by

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \cdot \hat{g}_k(\hat{\theta}_k)$$

where $\hat{\theta}_k$ is the set of control parameter values in the k-th iteration, $a_k$ is a nonnegative gain coefficient.
SPSA-Based Optimization Algorithm

\( \hat{g}_k(\hat{\theta}_k) \) is the simultaneous perturbation estimate of \( G(\theta) \)'s gradient and is calculated as,

\[
\hat{g}_k(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k \Delta_k) - y(\hat{\theta}_k - c_k)}{2c_k \Delta_k} \begin{bmatrix} \Delta_k^{-1} \\ \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{bmatrix}
\]

where \( \Delta_k = [\Delta_k^{-1}, \Delta_{k1}^{-1}, \ldots, \Delta_{kp}^{-1}]^T \) is a random perturbation vector and \( c_k \) is a positive coefficient.
Outline

• Introduction and Motivation

• Related Work

• Problem Statement

• Design of NoStop

• Performance Evaluation
Performance Evaluation

• Experiment Setting
  – A heterogeneous cluster consisting of five compute nodes
  – Apache Hadoop 3.2.1, Apache Spark 3.0.0 and Apache Kafka 2.5.0.
  – Four workloads with varying input data rate.

(a) Logistic Regression  (b) Linear Regression
(c) WordCount        (d) Log Analyze
Experimental Results

Optimization evolution process
Experimental Results

Performance improvement over initial configurations set by default
Experimental Results

Comparison between SPSA and Bayesian optimization

(a) Comparison of steps
(b) Comparison of end-to-end delay
Thank You!