HFSP: Bringing Size-Based Scheduling To Hadoop

ABSTRACT:

Size-based scheduling with aging has been recognized as an effective approach to guarantee fairness and nearoptimal system response times. We present HFSP, a scheduler introducing this technique to a real, multi-server, complex and widely used system such as Hadoop. Size-based scheduling requires a priori job size information, which is not available in Hadoop: HFSP builds such knowledge by estimating it on-line during job execution. Our experiments, which are based on realistic workloads generated via a standard benchmarking suite, pinpoint a significant decrease in system response times with respect to the widely used Hadoop Fair scheduler, without impacting the fairness of the scheduler, and show that HFSP is largely tolerant to job size estimation errors.

INTRODUCTION

THE advent of large-scale data analytics, fostered by parallel frameworks such as Hadoop, Spark and Naiad has created the need to manage the resources of compute clusters operating in a shared, multi-tenant environment. Within the same company, many users share the same cluster because this avoids redundancy in physical deployments and in data storage, and may represent enormous cost savings. Initially designed for few very large batch processing jobs, data-intensive scalable computing frameworks such as MapReduce are nowadays used by many companies for production, recurrent and even experimental data analysis jobs. This heterogeneity is substantiated by recent studies that analyze a variety of production-level workloads.
EXISTING SYSTEM

In Existing System we address the problem of job scheduling, that is how to allocate the resources of a cluster to a number of concurrent jobs, and focus on Hadoop, the most widely adopted open-source implementation of MapReduce. Currently, there are mainly two different strategies used to schedule jobs in a cluster. The first strategy is to split the cluster resources equally among all the running jobs. A remarkable example of this strategy is the Hadoop Fair Scheduler. While this strategy preserves fairness among jobs, when the system is overloaded, it may increase the response times of the jobs. The second strategy is to serve one job at a time, thus avoiding the resource splitting. An example of this strategy is First-In-First-Out (FIFO), in which the job that arrived first is served first.

DisADVANTAGE OF Existing SYSTEM

✓ The scheduling choices lead inevitably to poor performance
✓ Small jobs may find large jobs in the queue, thus they may incur in response times that are disproportionate to their size.
✓ As a consequence, the interactivity is difficult to obtain.
✓ Both strategies have drawbacks that prevent them from being used directly in production without precautions.

PROPOSED SYSTEM

In Proposed System we presented an novel approach to the resource allocation problem, based on the idea of size-based scheduling. Our effort materialized in a full-fledged scheduler that we called HFSP, the Hadoop Fair Sojourn Protocol, which implements a size-based discipline that satisfies simultaneously system responsiveness and fairness requirements.

ADVANTAGE OF PROPOSED SYSTEM

✓ Evaluating job sizes online without wasting resources
Avoiding job starvation for both small and large jobs, and guaranteeing short response times despite estimation errors were the most noteworthy.

HFSP uses a simple and practical design: size estimation trades accuracy for speed, and starvation is largely alleviated, by introducing the mechanisms of virtual time and aging.

**HARDWARE REQUIREMENTS:**

- System: Pentium IV 2.4 GHz.
- Hard Disk: 40 GB.
- Floppy Drive: 44 Mb.
- Monitor: 15 VGA Colour.

**SOFTWARE REQUIREMENTS:**

- Coding Language: Java 1.7, Hadoop 0.8.1
- Database: MySql 5
- IDE: Eclipse