Procedia Computer Science

Procedia Computer Science 101, 2016, Pages 323 – 332

YSC 2016. 5th International Young Scientist Conference on Computational Science

Quality-based workload scaling for real-time streaming systems

Pavel A. Smirnov, Denis Nasonov
ITMO University, St.Petersburg, Russia
{smirnp, denis_nasonov}@niuitmo.ru

Abstract

In this paper we propose an idea to scale workload via elastic quality of solution provided by the particular streaming applications. The contribution of this paper consists of quality-based workload scaling model, implementation details for quality assessment mechanism implemented at the top of Apache Storm and experimental evaluation of the proposed model on a synthetic and real-world (medical) examples.

Keywords: scaling model, data streaming, elastic workload, quality of service, apache storm, big data

1 Introduction

Nowadays huge amounts of data are produced in real time manner. Autonomous vehicles, physical sensors, social-network activities, stock-exchange markets are typical producers of streaming data, which require immediate real-time processing to be actual in a short period of time.

Performance optimization for real-time streaming applications is a complex non-trivial process. The goal of profiliration and optimization process is to find an optimal tradeoff between two states of an application: underutilization and overloading. A result of non-optimal performance occurs queues of tuples behind the overloaded operators. Open-source streaming engines like Apache Storm* are smart enough to handle overloads by pausing an emission if downside operators cannot serve it (this feature is called “backpressure’). But if amounts of data from third-party sources remain unchangeable, an overloaded application will accumulate an endless queues of data, which may be lost as a result of crash. While there are at least 5 ways to scale application performance in general (Ahluwalia, 2007), only two of them become popular in field of data streaming: to increase operator’s parallelism and to add extra hardware resources. The literature overview in field of scaling streaming applications outlined that these two patterns are usually applied simultaneously: parallelism increase is reached via allocation of new operators on a newly-added hardware. From economic point of view new hardware resources cause additional costs, so it makes sense for more effective operators’ placement (called “scheduling”), which also impacts on application’s service rate.

* http://storm.apache.org/

Peer-review under responsibility of organizing committee of the scientific committee of the 5th International Young Scientist Conference on Computational Science

doi:10.1016/j.procs.2016.11.038

In contrast with scheduling and resource scaling in this paper we propose the idea to scale workload via elastic quality of solution, which a streaming application provides. The contribution of this paper consists of: the quality-based workload scaling model for streaming applications; the implementation details of quality assessment mechanism implemented at the top of Apache Storm; experimental evaluation of the proposed model on the synthetic and real-world (medical) applications. The remainder of the paper is organized as follows: section II presents the literature study; the formal definition of the proposed model is presented in section III; the implementation details of quality assessment mechanism are described in section IV; section V presents the results of experimental evaluation; section VI contains conclusion and plans for the future work with potential use-cases regarding quality-based mass-serving applications.

2 Related works

Today a workload scaling patterns for online streaming applications is the actual state-of-art problem. The widely spread open-source streaming engines like Apache Storm[†], Samza[‡], Flink[§], Twitter Heron[**] and others cannot scale workload automatically, but may do it according to some signals from user. This lead to appearance of several simultaneous research efforts devoted to active and proactive workload scaling techniques.

Paper (Xu, Peng, & Gupta, 2016) describes system called Stela, which provides active workload scaling via regular calculation of service rates (authors call it a congestion or ETP-metric) for all operators and increase it for the most ETP-slowest operators by allocating extra instances on newly allocated hardware. The Stela is implemented on top of the Apache Storm platform and automatically captures performance statistics from Storm API. Stela periodically refreshes operators' ETPs and automatically scales-in and scales-out topologies in Apache Storm.

The research (Heinze et al., 2015) proposes online optimization approach, which automatically detects changes in workload pattern, chooses and applies a scaling strategy to minimize number of hosts used for current workload characteristics. Changes of workload patterns are detected by an adaptive windows approach. The approach is implemented on top of FUGU (Heinze et al., n.d.), which is elastic data stream processing engine developed by the authors during previous research efforts. For overloaded hosts the system automatically decides which operators should be unchanged and which should be moved to a newly added resources. The decisions are made according to local and global threshold-based rules (Heinze, Pappalardo, Jerzak, & Fetzer, 2014), which deal with operators' performance metrics. Authors also propose a QoS-based (Heinze, Jerzak, Hackenbroich, & Fetzer, 2014) approach for workload scaling, where latency is the main QoS constraint, usually strictly declared in service level agreement.

In (De Matteis & Mencagli, 2016) authors propose latency latency-aware and energy-efficient scaling strategies with predictive capabilities. The strategies are made via adaptation of the Model Predictive Control – a technique for searching optimal applications' configurations along a limited prediction horizon. The paper presents models, which describe dependency between QoS variables (latency, energy) and a configuration of the system (parallelism, CPU frequency and etc.). The result of optimization is a reconfiguration trajectory within a strictly-limited prediction horizon.

Paper (Hidalgo, Wladdimiro, & Rosas, 2016) is devoted to reactive and predictive resource scaling according to workload demands. Authors apply optimization approach called fission to increase the amount of replicas for fully-loaded operators. The two algorithms are proposed to determine an operator's state: a short-term and a mid-term state for peaks and patterns detection respectively. The

[†] http://storm.apache.org/
[‡] http://samza.apache.org/
[§] http://flink.apache.org/
[**] http://twitter.github.io/heron/