Synergy: Quality of Service Support for Distributed Stream Processing Systems

Thomas Repantis   Vana Kalogeraki
Department of Computer Science & Engineering
University of California, Riverside
{trep,vana}@cs.ucr.edu

Xiaohui Gu
Department of Computer Science
North Carolina State University
{gu}@csc.ncsu.edu

http://synergy.cs.ucr.edu/

A significant number of emerging on-line data analysis applications require the processing of data that get updated continuously, to generate outputs of interest or to identify meaningful events: i) Analysis of the clicks or textual input generated by the visitors of web sites such as e-commerce stores or search engines, to determine appropriate purchase suggestions or advertisements. ii) Monitoring of network traffic to detect patterns that proclaim attacks or intrusions, to filter out traffic that violates security policies, or to discover trends that can help with network configuration. iii) Analysis of the data collected by sensors when monitoring wildlife, plants, traffic, or the environment for fire, earthquakes, or intruders.

We refer to this kind of continuously updated data as streams and the applications that process this kind of data in real-time as stream processing applications. Stream processing applications consist of software components that operate on a set of input streams to produce a set of output streams. The combined processing of the data streams by all the different components constitutes the execution of a stream processing application. To sustain the high load of processing streaming data in real-time, the stream processing components are often hosted by different machines, which communicate over a network. We refer to this type of distributed system as a distributed stream processing system.

Distributed stream processing applications have Quality of Service (QoS) requirements, expressed in terms such as end-to-end delay, throughput, miss rate, or availability. For example, an alert needs to be raised within a certain time frame after an intrusion. Adhering to such QoS requirements is crucial for the dependable operation of a distributed stream processing system. The first step towards satisfying the QoS requirements of stream processing application is taking them into account during the application composition. However, as the incoming data rates may increase at run-time, due to external events such as a network attack or a rapid popularity growth of some news event, an application execution may cease to adhere to the requested QoS. Under such dynamically changing conditions, providing application QoS is a challenging task. The problem is complicated further by the large scale and the distributed nature of a DSPS. Accurate centralized decisions are infeasible, due to the fact that the global state of a large-scale DSPS is changing much faster than it can be communicated to a single host.

We have designed and implemented Synergy, a distributed stream processing middleware. Synergy is a software running on every machine of a distributed system to offer distributed stream processing applications, under Quality of Service constraints, and while efficiently managing the system's resources. Unlike previously developed distributed stream processing systems, Synergy offers sharing-aware component composition. Sharing-aware component composition allows stream processing applications that are composed of individual components to utilize i) previously generated streams and ii) already deployed stream processing components. Synergy's component composition protocol takes the user-requested QoS into account when composing the application component graph. Furthermore, Synergy employs hot-spot prediction and alleviation techniques, to ensure that the application QoS requirements continue to be met at run-time. Finally, Synergy employs a component placement protocol that aims to maximize application availability.

The major research contributions of Synergy can be summarized as follows:

Synergy employes a decentralized light-weight composition algorithm that can discover streams and components at run-time and check whether any of the existing components or streams can satisfy a new application request. After the qualified candidate components have been identified, components and streams are selected and composed dynamically such that the application resource requirements are met and the workloads at different hosts are balanced.

Synergy integrates a QoS impact projection mechanism into the distributed component composition algorithm to evaluate the reusability of existing stream processing components according to the applications' QoS constraints.

Synergy encompasses a framework built on statistical forecasting methods, to accurately predict QoS violations at run-time and proactively identify application hot-spots. In order to achieve this, our prediction framework binds workload forecasting with execution time forecasting. To accomplish workload forecasting we predict rate fluctuations, exploiting auto-correlation in the rate of each component, and cross-correlation between the rates of different components of a distributed application. To accomplish execution time forecasting we use linear regression, an established statistical method, to accurately model the relationship of the application execution time and the entire workload of a node, while dynamically adapting to workload fluctuations.

Synergy enables nodes to react to predicted QoS violations and alleviate hot-spots by autonomously migrating the execution of stream processing components using a non-disruptive migration protocol. Candidate selection for migration is based on preserving QoS. We employ prediction again to ensure that migration decisions do not result to QoS violations of other executing applications. To drive migration decisions in a decentralized manner we build a load monitoring architecture on top of a Distributed Hash Table.

For initial component deployment Synergy employs a decentralized replica placement protocol that aims to maximize availability, while respecting resource constraints, and making performance-aware placement decisions.

We have implemented a prototype of Synergy and evaluated its performance on the PlanetLab wide-area network testbed using a real network monitoring application operating on traces of real TCP traffic. We have also conducted extensive simulations to compare Synergy's composition algorithm to existing alternative schemes. Our experimental results showed that Synergy achieves much better resource utilization and QoS provision than previously proposed schemes, by judiciously sharing streams and processing components during application composition. Our experimental evaluation has also demonstrated high load prediction accuracy, and substantial benefits in application QoS achieved by migration.

In conclusion, we have designed and implemented Synergy, a completely decentralized distributed stream processing middleware that incorporates mechanisms for sharing-aware component composition and hot-spot prediction and alleviation. More information on the Synergy middleware can be found at http://synergy.cs.ucr.edu