Auto-Scaling Techniques for Spark Streaming

Master-Thesis von Seyedmajid Azimi Gehraz
Tag der Einreichung:

1. Gutachten: Prof. Dr. rer. nat. Carsten Binnig
2. Gutachten: Dr. Thomas Heinze

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Data Management

Auto-Scaling Techniques for Spark Streaming

Vorgelegte Master-Thesis von Seyedmajid Azimi Gehraz

1. Gutachten: Prof. Dr. rer. nat. Carsten Binnig
2. Gutachten: Dr. Thomas Heinze

Tag der Einreichung:

Contents

List of Figures

List of Tables

1 Abstract

3 Spark Streaming

6 Implementation Detail

Dynamic resource scaling in cloud environments has been studied extensively in literature. As a naive implementation, there are two thresholds, namely *upper bound* and *lower bound*. However, such an implementation suffers from *oscillating* decisions. In order to remedy this issue, *grace period* shall be enforced. During this period, no scaling decision is made.

Hasan et al. [2] introduced four thresholds and two time periods. `ThrUpper` defines upper bound. `ThrBelowUpper` is slightly below `ThrUpper`. Similarly, `ThrLower` defines lower bound and `ThrAboveLower` is slightly above the lower bound. In case, system utilization stays between `ThrUpper` and `ThrBelowUpper` for a specific duration, then cluster controller decides to take a scale-out action, by adding resources. On the other hand, if system utilization stays between `ThrLower` and `ThrAboveLower` for a specified duration, then controller decides to take scale-in action. Defining two levels of thresholds helps to detect workload *persistency* and avoid making immature scaling decision. However, defining thresholds is a tricky and manual process, and need to be carefully done[1]. It should be noted that, computation overhead of this approach is very low.

RightScale applies voting algorithm[6] among nodes to make scaling decision. In order for a specific action to be decided, majority of nodes should vote in favour of that action. Otherwise, no action is selected as a default action. Afterwards, nodes apply grace period to stabilize the cluster. The complexity of the voting process in trusted environments is in the order of $O(n^2)$, which leads to heavy network traffic among participants when cluster size grows. This approach is also categorized in threshold based approaches. Thus, it suffers from the same issue as mentioned above.

Herbst et al. [3] surveys many different auto-scaling techniques based on time-series analysis in order to forecast *trends* and *seasons*. *Moving Average Method* takes the average over a sliding window and smooths out minor noise level. Its computational overhead is proportional to size of the window. *Simple Exponential Smoothing*(SES) goes further than just taking average. It gives more weight to more recent values in sliding window by an exponential factor. Although it is more computationally intensive compared to moving average, it is still negligible. SES is capable of detecting short-term trends but fails at predicting seasons. These approaches are more specific instances of *ARIMA* (Auto-Regressive Integrated Moving Average) which is a general purpose framework to calculate moving averages. However, time-series analysis is only suitable for stationary problems consist of recurring workload patterns such as web applications. In case of streaming (specially in multi-tenant environments), in which the workload is highly unpredictable, time-series analysis tends to be less useful. Additionally, more advanced forms of time-series analysis which are capable of forecasting seasons (such as *tBATS Innovation State Space Modelling Framework*[5], *ARIMA Stochastic Process Modelling Framework*[4]) are computationally infeasible for streaming workloads.

[1] X. Dutreilh, A. Moreau, J. Malenfant, N. Rivierre, and I. Truck. “From Data Center Resource Allocation to Control Theory and Back”. In: *2010 IEEE 3rd International Conference on Cloud Computing*. 2010, pp. 410–417. DOI: 10.1109/CLOUD.2010.55.

[2] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, and S. L. D. Gudreddi. “Integrated and autonomic cloud resource scaling”. In: *2012 IEEE Network Operations and Management Symposium* (2012), pp. 1327–1334.

[3] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn. “Self-adaptive Workload Classification and Forecasting for Proactive Resource Provisioning”. In: *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*. ICPE ’13. Prague, Czech Republic: ACM, 2013, pp. 187–198. ISBN: 978-1-4503-1636-1. DOI: 10.1145/2479871.2479899. URL: http://doi.acm.org/10.1145/2479871.2479899.

[4] R. Hyndman and Y. Khandakar. “Automatic Time Series Forecasting: The forecast Package for R”. In: *Journal of Statistical Software, Articles* 27.3 (2008), pp. 1–22. ISSN: 1548-7660. DOI: 10.18637/jss.v027.i03. URL: https://www.jstatsoft.org/v027/i03.

[5] A. M. D. Livera, R. J. Hyndman, and R. D. Snyder. “Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing”. In: *Journal of the American Statistical Association* 106.496 (2011), pp. 1513–1527. DOI: 10.1198/jasa.2011.tm09771. URL: https://doi.org/10.1198/jasa.2011.tm09771.

[6] RightScale. *Set up Autoscaling using Voting Tags*. Accessed June 20, 2018. 2018. URL: http://support.rightscale.com/12-Guides/Dashboard_Users_Guide/Manage/Arrays/Actions/Set_up_Autoscaling_using_Voting_Tags/.