# Doctoral Symposium: Dynamic Scaling of Distributed Data-flows under Uncertainty

Stuart Jamieson
s.jamieson3@newcastle.ac.uk
School of Computing, Newcastle University, United Kingdom, NE1 7RU
Supervised by Dr Matthew Forshaw

## ABSTRACT

Existing approaches to dynamic scaling of streaming applications often fail to incorporate uncertainty arising from performance variability of shared computing infrastructures, and rapid changes in offered load. We explore the definition and incorporation of risk and uncertainty, and advocate for risk-adjusted measures of performance and their application in improving the robustness of autonomic scaling of streaming systems.

## CCS CONCEPTS

**• Information systems → Data streaming**; **• Computer systems organization → Self-organizing autonomic computing**; **Real-time systems**; **• General and reference → Performance**.

## KEYWORDS

Streaming, Autonomic Scaling, Risk, Uncertainty, Performance

**ACM Reference Format:**
Stuart Jamieson. 2020. Doctoral Symposium: Dynamic Scaling of Distributed Data-flows under Uncertainty. In *The 14th ACM International Conference on Distributed and Event-based Systems (DEBS '20), July 13–17, 2020, Virtual Event, QC, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3401025.3406444

## 1 INTRODUCTION

Stream processing engines are a common execution platform for a variety of contemporary data-centre applications. A minor degradation in application performance can have a high penalty on cloud operators: Google reported 20% revenue loss, when an experiment caused an additional delay of 500 ms in response time. Amazon reported 1% sales decrease for an additional delay of 100 ms in search results. The impact of performance degradation is exacerbated in delay-sensitive applications e.g. streaming healthcare analytics.

Performance variability arises from rapid changes in offered load, workload skew [12] and performance interference observed when running atop public cloud infrastructures. State-of-the-art scaling controllers fail to model these interactions, instead making strong assumptions (e.g. linearity of performance under scaling) which compromise their effectiveness. We advocate for approaches capable of making robust operating decisions under uncertainty.

We outline key challenges in the autonomic scaling of streaming systems, through *'case studies'* drawing analogies from financial markets. Primarily we investigate methods to account for *'idiosyncratic'* risks as the direct result of one's decision-making versus *'systematic'* risks which originate outside of one's control.

Financial investment portfolio managers are faced daily with the need to identify, quantify and manage or mitigate the risks inherent in large, complex portfolios of assets. The need for a clear understanding of the associated risks is paramount; numerous models, methods and approaches have been developed over the past few decades, ranging from Modern Portfolio Theory [23], to the Credit Derivative risk models employed by large banks in the post Global Financial Crisis environment (with the failure of the Gaussian copula models used in the preceding period showing the potential downside of reliance on incorrect/unstable models [11]). We aim to adopt methods from this field and apply them to risk identification, quantification and mitigation in the stream processing arena.

Systematic trading strategy practitioners also face the problem of how to develop, optimise and evaluate new trading strategies. Evaluation of a live trading history, while far from being a simple procedure, is often less susceptible to misdiagnoses and Type I/II errors than a historical *back-test* of a proposed strategy. Algorithm design and historic *back-tested* (i.e. simulated) results evaluation must be carried out in such a manner as to avoid *biases* or *curve-fitting*. Optimisation attempts must also be able to account for numerous input parameters to be optimised, along with producing a robust and reliable result in an acceptable time-frame and display an acceptably low *out-of-sample* performance degradation [5].

## 2 REPORTING & MEASUREMENT METRICS

***Problem:*** With streaming systems, performance is often measured and recorded in a manner that fails to incorporate the notion of sensitivity or robustness to changing input parameters or operating environment [19]. Nor do we often find performance metrics that factor in the *maximum downside* experienced throughout the test (e.g. incorporating such a notion as some form of denominator with which we transform the performance metric into a *risk-adjusted* format). Also, seldomly are results gauged or compared versus the notion of what one would hope to expect from a certain set-up configuration choice, or specific system environment.

***State of the art:*** Streaming systems are typically evaluated on throughput and latency [20]. Other metrics have been proposed such as correctness, the capacity of adapting to stream load variations and uncertainty or fuzzy patterns compliance [24], along with memory consumption, maximum peak and post-peak latency variation ratio [25]. Bordin [9] identified the most frequently used metrics; latency, throughput, scalability, tuple loss, and resource usage. These metrics are framed as absolutes with no notion of accompanying adjustment for risks inherent in the specific deployment. It also should be ensured that any instrumentation used to record any relevant metrics or measurements does not itself perturb or alter the normal operation of the system under evaluation [26].

***Proposed approach:*** In finance the usual approach to ranking or selecting investments and asset purchases when following a *momentum* strategy has relied on evaluating the proposed assets' individual monthly returns over the ranking period. This realised cumulative return as a selection criterion is a simple measure which does not include the risk component of the asset behaviour. The fact that momentum strategies are far from *risk-less*, in addition to empirical evidence showing that individual asset returns exhibit non-normality, suggests it would be more reliable to use a measure that could account for these properties, such as the Sharpe ratio [2].

While a risk-adjusted measure of performance leads to superior ranking results, any metric will contain some estimation error itself which can play an important role in optimal portfolio choice [18, 27, 36]. Analytical results for the approximate bias and variance of the sample Sharpe ratio in terms of the underlying distribution parameters [6] may alleviate the problems of estimation risk.

I propose to adapt the principles behind several risk-adjusted return metrics used in the financial investment and trading domain and apply them to generating accurate and representative performance metrics for distributed data stream processing systems. Generally, three main metrics are used to measure portfolio risk-adjusted performance in a financial setting:

- Sharpe Ratio: excess return, per unit of standard deviation.
- Treynor Ratio: return, per unit of *systematic risk*.
- Jensen's Alpha: return over its *expected return*.

The three metrics are based on the same foundational concepts, but each differ in how the investor frames their paradigm of *risk* which is used as the denominator to transform the measure into a risk-adjusted value [15]. The Sharpe Ratio uses total risk (both *idiosyncratic* and *systematic*) as the denominator, while the Treynor includes only *systematic* risk. Mapped to the streaming context, this is analogous to including only risks that are outside of the streaming system operators' influence and control versus including risks that are a direct result of their actions.

We propose to generate a range of performance metrics which have been transformed into *risk-adjusted* measures through the use of *downside* or cost/risk metrics as normalisation parameters (e.g. throughput, memory consumption, latency, system-failure instances, load-shedding). This may allow much more meaningful and accurate comparisons between streaming systems which display relatively dissimilar characteristics or topologies.

## 3 SYSTEM OPTIMISATION & RE-BALANCING

***Problem:*** Streaming system parameter optimisation is highly non-trivial, displaying a non-convex, irregular solution space [33]. This challenge is exacerbated by the need to react in near real time, and to maintain pre-agreed service quality metrics. Lorido-Botrán et al. [22] identify several issues faced by an auto-scaler; *Under-provisioning*, *Over-provisioning*, and *Oscillation.* The matter is complicated further due to workloads and modern shared cluster environments exhibiting high variability and unpredictability.

***State of the art:*** Despite extensive research, current stream processing engines lack the ability to automatically grow and shrink to meet the needs of streaming applications; this includes Apache Storm and Heron, Apache Flink and Spark Streaming [1]. There have been numerous models and approaches developed to auto-tune distributed stream processing systems including threshold-based policies [30], reinforcement learning [35] and supervised learning models [14]. Several techniques are often applied to assist in model development, among them enhanced configuration sampling, metric dimensionality reduction and machine learning for both system configuration optimisation and for workload prediction [35].

Fischer et al. [13] found that using Bayesian Optimisation approaches resulted in significant gains over a parallel linear approach. Bilal and Canini [8] developed a hill-climbing algorithm that accounts for desirable initial configurations of stream processing applications along with a novel gray-box optimization algorithm based on a rule-based heuristic approach that provides comparable results while being two to five times faster

Much work has been carried out in the closely related area of QoS-aware service composition, the purpose of which is the selection of the best set of services to compose, meeting global QoS constraints imposed by the user [7]. All approaches to this problem to date have made a number of simplifying assumptions as the problem is known to be NP-hard [32] and can be modelled as a multidimensional, multi-objective, multi-choice knapsack problem (MMMKP) [16].

DS2 [17], offers an auto-scaler which leverages knowledge of the dataflow graph, the computational dependencies among operators, and estimates the operators' true processing and output rates. However it only targets workload changes on a timescale greater than its own convergence time; if too short it relies on using *back-pressure*, *buffering* or *load shedding* to generate the signals. This leads to a more stable result than dynamic scaling at a cost of increased latency or lost data as scaling too often over short horizons results in inefficient fluctuations. We propose that any scaling decision made based on fluctuations of current environment variables which prove to be of a relatively small magnitude may also lead to similar sub-optimal changes and therefore performance.

***Proposed approach:*** Our approach will frame the topic as two distinct, but inter-related problems; when we scale the system, and what the scaler chooses as the new system configuration.

In finance, portfolios can be re-balanced using the *percentage of portfolio re-balancing approach* whereby actions are triggered by changes in relative asset values, i.e. whenever any asset class moves far enough away from its optimal weight [29]. Tolerance bands are used to strike a balance between the adverse effects of allowing an allocation to stray too far from its optimal and the execution costs incurred through frequent re-balancing actions. Several factors impact the width of a tolerance band:

- Transaction costs (+ , risk $\uparrow$ , band width $\uparrow$)
- Risk tolerance (+ , risk $\uparrow$ , band width $\uparrow$)
- Correlation of asset returns (+ , risk $\uparrow$ , band width $\uparrow$)
- Volatility of the asset ($-$ , risk $\uparrow$ , band width $\downarrow$)

- Volatility of other portfolio assets (− , risk ↑ , band width ⇃)

Different re-balancing strategies tend to perform better in different market regimes, whether trending (up or down), or oscillating.

The inefficient fluctuations in streaming systems can be seen as analogous to re-balancing a portfolio of financial assets before the tolerance band has been breached. The asset weights would be analogous to the magnitude of latency, load shedding, system downtime, or other specified cost/risk. These can be considered in terms of their relative position within the overall portfolio that makes up our operating environment. The width of a tolerance band in the instance of a streaming system would be influenced by:

- Individual cost/risk tolerance (+ , risk ↑ , band width ↾)
- Correlation between costs/risks (+ , risk ↑ , band width ↾)
- Volatility of the costs/risks (− , risk ↑ , band width ⇃)
- Volatility of other costs/risks (− , risk ↑ , band width ⇃)

We propose to quantify the leading constituent cost/risk factors and incorporate the concept of tolerance bands into the scaling decision process. This may allow a reduction in scaling decisions being enacted which result in an inefficient fluctuation when considering the benefit gained versus the costs incurred.

Once the auto-scaler has decided to enact a scaling or re-balancing of the system, it is necessary to identify a set of new system configuration and parameter values to implement. These new parameters should represent what our model perceives as the optimal configuration, and need to be calculated and produced in near real-time.

In optimising algorithmic trading strategy parameters or financial asset portfolio constituents, key problems relate to the sensitivity to small parameter value changes and numerous local extrema, distributed over the solution space in an irregular way. Certain methods were designed to significantly reduce optimisation computation time, without a substantial loss of strategy quality. These include Differential Evolution and Genetic Algorithms, which join the exploitation of past results with exploration of the search space [10].

Differential evolution has been shown to outperform genetic algorithms for numerical multi-objective optimisation [34]. Differential Evolution [31] was proposed for solving problems with an irregular solution space; it was shown to successfully minimise Conditional Value at Risk (CVaR) for large-scale portfolios [3].

Streaming systems can vary significantly across varying use cases, and domains of application. Characteristics include the complexity of the computation, the volume and volatility of the incoming data stream, along considerations of the system provider, such as meeting service level agreements. It is not unusual to find that a system's quantity of variables that need to be optimised grows relatively large, often each with a relatively large possible range of values, creating an unmanageable number of permutations to test across using a simple brute force approach; this is where the development of an optimisation model and process that allows the computation time to be dramatically reduced without experiencing a corresponding degradation in performance is highly desired.

Our efforts will attempt to improve on current state-of-the-art optimisation models, balancing the trade-off between computational cost, robustness and performance of the resulting optimised model. Initial efforts will focus on Genetic Algorithms and Differential Evolution approaches and will aim to successfully manage the *exploration versus exploitation* trade-off.

## 4 STRESS TESTING & SCENARIO ANALYSIS

***Problem:*** It is important for operators to have confidence in how their proposed system is expected to perform under differing situations and operating environments. Operators must be cognisant of the implications of a wide range of possible eventualities ranging from probable but low-impact to improbable but severe situations.

***State of the art:*** Previous works investigate stress testing and scenario analysis on streaming systems [4]. Recent approaches explore coordinated failure scenarios which propagate across a streaming system [26]. Attempts have been made to identify and categorise testing methods, focusing on varying areas and characteristics of potential sources or points of system weakness [21]. Three criteria are often used to evaluate fault-injection approaches; representativeness, usability and efficiency [28].

***Proposed approach:*** Enterprise *stress testing*, *reverse stress testing* and *scenario analysis*, the process whereby banks assess their financial resilience to macro-economic or market-driven scenarios, has changed over the last decade or so from what used to be a simple, top-down process into a complex, bottom-up modelling exercise, involving almost every function within the bank.

Even at the individual financial asset portfolio level, equity, fixed-income, and options positions can be characterised by a number of exposure measures that reflect the sensitivities of these positions to movements in underlying risk factors. Sensitivity measures examine how performance responds to a single change in an underlying risk factor. Understanding and measuring how portfolio positions respond to the underlying sources of risk is a primary objective in managing this risk. A scenario risk measure estimates the portfolio return that would result from a hypothetical change in markets (a hypothetical scenario) or a repeat of an historical event (a historical scenario). One of the major areas of benefit that results from carrying out these tests and scenarios is in the identification and quantification of how individual assets in a portfolio respond to various stresses and events. Moreover, it also reveals how one may expect assets to respond to these shocks, in relation to the other assets held. This can then be used to explore and measure a comprehensive range of more nuanced and focused interactions and metrics such as *Marginal VaR* and *Conditional VaR* of individual assets. These measure the additional risk added to your portfolio by adding $1 worth of the asset in question ( *MVaR*), whereas the *CVaR* measures the proportion of overall risk which can be attributed to an individual asset's entire allocation.

We propose the development of a framework which aids relevant parties to gauge not only the magnitude of risk inherent to their system design choices, but also understand their exposure to certain events or environment changes. Attempts will be made to incorporate what are deemed to be the most important collection of factors, much in the same way financial models concentrate on capturing the major proportion of causality while making a number of conscious design choices to avoid unnecessary complexity. Table 1 maps financial risk categories onto the streaming domain.

The model will adopt both parametric and Monte Carlo methods to allow more accurate quantification of the magnitude of risk exposures. It will also capture differing rates of change caused by the interaction of non-linear relationships (much like financial options contracts display non-linear payout structures).

**Table 1: Mapping of financial risk to streaming applications**

| Financial Risk | Description | Streaming Risk Map |
|---|---|---|
| Market risk | The risk of losses caused by adverse price movements | Concept drift and volatility shifts in input stream/arrival rates |
| Credit risk | The risk of a loss resulting from a borrower's failure to repay a loan or meet contractual obligations | A company's risk of a loss resulting from a 3rd party's failure to remain solvent and a "going concern" |
| Liquidity risk | The risk that a company or bank may be unable to meet short term financial demands | The risk that a company may face a lack of access to urgently needed resources in sufficient quantities |
| Counter-party risk | The risk that one of those involved in a transaction might default on its contractual obligation | The risk resulting from the effects as a direct result of reliance on a 3rd party which experiences a negative event |
| Model or Estimation risk | The risk that a financial model used to measure quantitative information and affects the outcome of financial securities valuations fails or performs inadequately | The risk that a model used to optimise stream processing system parameters or calculate/interpret metrics fails or performs inadequately |
| Operational risk | The risk arising from inadequate or failed internal processes, people and systems, or from external events | The risk arising from the failure or incorrect management of systems and processes considered external or exogenous to the streaming engine |

## 5 CONCLUSIONS & FUTURE WORK

We have explored several opportunities where risk and uncertainty can be incorporated into dynamic scaling decisions in streaming systems. Through participation in the Doctoral Symposium, we seek feedback from the community to develop our research ideas further, and to stimulate ongoing collaboration.

## ACKNOWLEDGMENTS

This work was supported by the Engineering and Physical Sciences Research Council, Centre for Doctoral Training in Cloud Computing for Big Data [grant number EP/L015358/1].

## REFERENCES

[1] Tarek M. Ahmed, Farhana H. Zulkernine, and James R. Cordy. 2016. Proactive Auto-Scaling of Resources for Stream Processing Engines in the Cloud *(CASCON)*.

[2] Dong-Hyun Ahn, Jennifer Conrad, and Robert Dittmar. 2003. Risk Adjustment and Trading Strategies. *Review of Financial Studies* 16, 2 (2003), 459–485.

[3] David Ardia, Kris Boudt, Peter Carl, Katharine Mullen, and Brian Peterson. 2010. Differential Evolution with DEoptim: An Application to Non-Convex Portfolio Optimization. *The R Journal* 3 (04 2010).

[4] Marcos Assuncao, Alexandre Veith, and Rajkumar Buyya. 2017. Distributed Data Stream Processing and Edge Computing: A Survey on Resource Elasticity and Future Directions. *Journal of Network and Computer Applications* 103 (12 2017).

[5] David Bailey, Jonathan (Jon) Borwein, Marcos Lopez de Prado, and Qiji Zhu. 2014. Pseudo-Mathematics and Financial Charlatanism: The Effects of Backtest Overfitting on Out-of-Sample Performance. *Notices of the American Mathematical Society* 61 (05 2014), 458. https://doi.org/10.1090/noti1105

[6] Yong Bao. 2009. Estimation Risk-Adjusted Sharpe Ratio and Fund Performance Ranking under a General Return Distribution. *Journal of Financial Econometrics* 7, 2 (Spring 2009), 152–173.

[7] Nebil Ben Mabrouk, Sandrine Beauche, Elena Kuznetsova, Nikolaos Georgantas, and Valérie Issarny. 2009. QoS-Aware Service Composition in Dynamic Service Oriented Environments. In *Middleware 2009*. Berlin, Heidelberg, 123–142.

[8] Muhammad Bilal and Marco Canini. 2017. Towards Automatic Parameter Tuning of Stream Processing Systems *(SoCC '17)*. 189–200.

[9] Maycon Viana Bordin. 2017. A benchmark suite for distributed stream processing systems.

[10] Eric Conrad, Seth Misenar, and Joshua Feldman. 2016. *Domain 8: Software Development Security (Understanding, Applying, and Enforcing Software Security)*. 429–477.

[11] Sanjiv R. Das, Darrel Duffie, Nikunj Kapadia, and Leandro Saita. 2007. Common Failings: How Corporate Defaults Are Correlated. *J Finance* 62, 1 (2007), 93–117.

[12] Junhua Fang, Rong Zhang, Tom Z.J. Fu, Zhenjie Zhang, Aoying Zhou, and Junhua Zhu. 2017. Parallel Stream Processing Against Workload Skewness and Variance *(HPDC '17)*. 15–26.

[13] Lorenz Fischer, Shen Gao, and Abraham Bernstein. 2015. Machines Tuning Machines: Configuring Distributed Stream Processors with Bayesian Optimization *(CLUSTER '15)*. USA, 22–31. https://doi.org/10.1109/CLUSTER.2015.13

[14] Avrilia Floratou, Ashvin Agrawal, Bill Graham, Sriram Rao, and Karthik Ramasamy. 2017. Dhalion: Self-Regulating Stream Processing in Heron. *Proc. VLDB Endow.* 10, 12 (Aug. 2017), 1825–1836. https://doi.org/10.14778/3137765.3137786

[15] Ken Hung, Chin-Wei Yang, and Dwight B. Means. 2006. *A note on the relationship among the portfolio performance indices under rank transformation.* 470–476.

[16] M. C. Jaeger, G. Muhl, and S. Golze. 2005. QoS-aware composition of Web services: a look at selection algorithms. 808. https://doi.org/10.1109/ICWS.2005.95

[17] Vasiliki Kalavri, John Liagouris, Moritz Hoffmann, Desislava Dimitrova, Matthew Forshaw, and Timothy Roscoe. 2018. Three steps is all you need: fast, accurate, automatic scaling decisions for distributed streaming dataflows. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 783–798.

[18] Raymond Kan and Guofu Zhou. 2007. Optimal Portfolio Choice with Parameter Uncertainty. *J. Fin. and Quant. Analysis* 42, 3 (2007), 621–656.

[19] Jeyhun Karimov, Tilmann Rabl, Asterios Katsifodimos, Roman Samarev, Henri Heiskanen, and Volker Markl. 2018. Benchmarking Distributed Stream Data Processing Systems. 1507–1518. https://doi.org/10.1109/ICDE.2018.00169

[20] Jeyhun Karimov, Tilmann Rabl, Asterios Katsifodimos, Roman Samarev, Henri Heiskanen, and Volker Markl. 2018. Benchmarking Distributed Stream Processing Engines. (2018).

[21] Francesca Lonetti and Eda Marchetti. 2018. Chapter Three - Emerging Software Testing Technologies. Advances in Computers, Vol. 108. Elsevier, 91 – 143.

[22] Tania Lorido-Botrán, Jose Miguel-Alonso, and Jose Lozano. 2014. A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *Journal of Grid Computing* 12 (12 2014). https://doi.org/10.1007/s10723-014-9314-7

[23] Harry Markowitz. 1952. Portfolio Selection. *J Finance* 7, 1 (1952), 77–91.

[24] Marcelo Mendes, Pedro Bizarro, and Paulo Marques. 2008. A framework for performance evaluation of complex event processing systems. 313–316.

[25] Marcelo R. N. Mendes, Pedro Bizarro, and Paulo Marques. 2009. A Performance Study of Event Processing Systems. In *TPCTC 2009*. 221–236.

[26] Saleh Mohamed, Matthew Forshaw, Nigel Thomas, and Andrew Dinn. 2017. Performance and Dependability evaluation of distributed event-based systems: a dynamic code-injection approach. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*. 349–352.

[27] Matthew Morey and Hrishikesh Vinod. 2001. Estimation Risk in Mutual Fund Ratings: The Case of Morningstar. *SSRN Electronic Journal* (05 2001).

[28] Roberto Natella, Domenico Cotroneo, and Henrique S. Madeira. 2016. Assessing Dependability with Software Fault Injection: A Survey. *ACM CSUR* 48, 3 (2016).

[29] Andre F. Perold and William F. Sharpe. 1988. Dynamic Strategies for Asset Allocation. *Financial Analysts Journal* 44, 1 (1988), 16–27.

[30] Ambalavanar Senthuran and Saman Hettiarachchi. 2020. A Review of Dynamic Scalability and Dynamic Scheduling in Cloud-Native Distributed Stream Processing Systems. In *ICDSMLA 2019*. 1539–1553.

[31] Rainer Storn and Kenneth Price. 1997. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. of Global Optimization* 11, 4 (Dec. 1997), 341–359. https://doi.org/10.1023/A:1008202821328

[32] A. Strunk. 2010. QoS-Aware Service Composition: A Survey. In *2010 Eighth IEEE European Conference on Web Services*. 67–74.

[33] Bart Theeten, Ivan Bedini, Peter Cogan, Alessandra Sala, and Tommaso Cucinotta. 2014. Towards the Optimization of a Parallel Streaming Engine for Telco Applications. *Bell Labs Technical Journal* 18 (03 2014), 181–197.

[34] Tea Tusar and Bogdan Filipic. 2007. Differential Evolution versus Genetic Algorithms in Multiobjective Optimization, Vol. 4403. 257–271.

[35] Luis Vaquero and Félix Cuadrado. 2018. Auto-tuning Distributed Stream Processing Systems using Reinforcement Learning.

[36] Yihong Xia. 2000. Learning about Predictability: The Effects of Parameter Uncertainty on Dynamic Asset Allocation. *The Journal of Finance* 56 (06 2000).