Deriving a Realistic Workload Model to Simulate High-Volume Financial Data Feeds for Performance Benchmarking

Vladimir Sladojević
vladimir.sladojevic@mail.polimi.it
Politecnico di Milano
Italy

Sebastian Frischbier
sebastian.frischbier@infrontfinance.com
Infront Financial Technology GmbH
Germany

Alexander Echler
alexander.echler@infrontfinance.com
Infront Financial Technology GmbH
Germany

Mario Paic
Mario.Paic@infrontfinance.com
Infront Financial Technology GmbH
Germany

Alessandro Margara
alessandro.margara@polimi.it
Politecnico di Milano
Italy

ABSTRACT

Processing financial market data at scale and in real-time poses a set of unique challenges to event-driven architectures due to the volume, variety, velocity, and veracity of the enclosed information on top of other constraints. Reproducible stress tests at scale using configurable benchmarks are key to building and tuning suitable processing systems. Available benchmarks, however, lack realistic and configurable workload models for market data scenarios. In previous work we already addressed this gap by describing the specific challenges of processing financial data at scale and by introducing a modular open-source benchmarking framework. This paper makes two contributions to the ongoing challenge of building realistic benchmarks for the financial data processing domain by outlining: (a) a detailed statistical analysis of real-world financial market data feeds processed on a global scale by Infront Financial Technology GmbH; and (b) a simple workload model built on this analysis to simulate high-volume market data feeds with their distinctive characteristics to be used in benchmarks. We evaluate our model using the DEBS 2022 Grand Challenge data set *Trading Data*.

CCS CONCEPTS

• **Information systems → Information systems applications**; • **Computing methodologies → Modeling and simulation**; • **Applied computing → Event-driven architectures**.

KEYWORDS

Event-based systems, financial data, benchmark, workload model

ACM Reference Format:
Vladimir Sladojević, Sebastian Frischbier, Alexander Echler, Mario Paic, and Alessandro Margara. 2022. Deriving a Realistic Workload Model to Simulate High-Volume Financial Data Feeds for Performance Benchmarking. In *The 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), June 27-30, 2022, Copenhagen, Denmark*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3524860.3539653

DEBS '22, June 27-30, 2022, Copenhagen, Denmark

ACM ISBN 978-1-4503-9308-9/22/06…$15.00
https://doi.org/10.1145/3524860.3539653

1 INTRODUCTION

Data is the oil fuelling financial markets, enabling market participants to decide on investments, spot trends, and react to significant events. High-volume streams of fine-granular and time-sensitive event notifications, called *market data feeds*, need to be processed, analyzed, and enriched with low latency to generate actionable knowledge before it becomes outdated. As one of Europe's leading providers of financial data and regulatory solutions, Infront Financial Technology GmbH (part of Infront group, f.k.a. vwd Vereinigte Wirtschaftdienste GmbH) processes a daily average of 40 billion market data event notifications from 500+ sources. The core processing systems handle daily peak rates of 2+ million notifications per second by relying on publish-subscribe (PS) architectures in addition to applying event stream-processing (ESP) and complex event-processing (CEP).

As part of ongoing work, we stress-test modern open-source platforms continuously with real data and micro benchmarks. Our long-term goal is to provide the community with a configurable open-source benchmarking tool to assess and compare ESP systems in financial application scenarios regarding their performance with realistic, controllable, and repeatable workloads. In previous work, we already proposed a modular open-source benchmarking framework [3] to address the lack of suitable systems. The work presented here aims at stress-testing the processing capabilities of ESP systems regarding throughput and latency while reasoning capabilities of a downstream CEP engine (e.g., pattern detection) are out of scope. Consequently, we focus on the characteristics of market data that are necessary to derive realistic workload definitions from. The semantics of the data are out of scope.

In this paper, we focus on the workload data and provide two novel contributions: (i) a detailed statistical analysis of real-world financial market data feeds, and (ii) a simple yet accurate workload model derived from the analysis, which enables simulating high-volume market data feeds with their distinctive characteristics. Our analysis captures the core characteristics of financial data relevant for performance benchmarking and identifies five key parameters that influence such characteristics. The derived model builds on a small number of parameters and (1) enables to tune them to simulate different scenarios; (2) enables the implementation of

lightweight data generators; (3) only relies on parameters that can be easily understood and set by domain experts.

The document is structured as follows: Sec. 2 provides background information on the diversity of financial data and discusses related work; Sec. 3 details the statistical analysis of financial data as processed by Infront and describes the mathematical workload model derived from the analysis; Sec. 4 evaluates the fit of our model by comparing simulated data to the DEBS Grand Challenge 2022 data set *Trading Data*; Sec. 5 summarizes our contributions and presents ongoing work.

2 BACKGROUND AND RELATED WORK

This section introduces key aspects of financial market data relevant for the scope of this paper, distilling the detailed discussion presented in [5]. We map the domain-specific terms to the nomenclature of the community working on event-based systems and we highlight the key differences of financial data with respect to data from other domains by comparing with Internet of Things (IoT) scenarios. Then, we describe the data sets used for our analysis and discuss related work.

2.1 Market data processing in a nutshell

Definition. Market data generally refers to time-sensitive and fine-granular information about the latest trade activity regarding a specific instance of a *financial instrument* called a (ticker) *symbol*. Examples for financial instruments are equities, funds, and indices while the stock of IBM is an instance of an equity. Market data is being made available at various levels of granularity and up-to-dateness, ranging from aggregated summaries to fine-granular and up-to-date *ticks* published by a specific exchange. Publishers are identified by a global Market Identifier Code (MIC) and financial instruments by an exchange-specific alpha-numeric identifier; the latter can vary so that the same instrument instance is sometimes traded using different symbols depending on the exchange. In consolidated and normalized feeds such ambiguous identifiers are already mapped to a format specific to the feed provider.

Nomenclature. We map the key terms from the domain of financial market data to those used in the event-processing community. Financial information delivered at different levels of granularity and timeliness can be directly mapped to the three different levels of event complexity regarding detection and processing: raw data like ticks resemble *simple events* while aggregates such as moving and weighted averages translate to *composite events*; further condensed and enriched information maps to *complex events*. In this nomenclature, exchanges map to sources/publishers; feeds to event streams that are being subscribed to while instrument types and (ticker) symbols are part of the identifier/payload of an event notification. Order is preserved only per symbol by each exchange; as a feed can contain data from multiple exchanges, the overall order in a feed cannot be guaranteed.

How is financial market data different? We highlight some key distinguishing features of financial market data by comparing with data from IoT scenarios, which are wellknown by the community working on event-based systems. The two domains differ significantly in key characteristics that have a massive impact on the ESP's performance when processing notifications. These singularities must be reflected in a realistic workload model for market data benchmarks. Hence, we briefly compare market data processing to IoT data processing based on [1, 5, 13], grouping by *topology*, *source*, and *data*.

Topology (dynamics, coordination, adaptation, connectivity). Market data processing setups are static, stable, and not resource-constrained compared to IoT setups: sources, ESP, and subscribing systems communicate via redundant broadband connections that are centrally coordinated and secured. Any kind of churn is considered an incident in market data processing and not the norm as it is in IoT scenarios. Same is true for mobility aspects that are irrelevant for market data processing other than in IoT scenarios where mobility of the ESP, data sources or subscribers is a central element that needs to be considered by the system's architecture.

Sources (identity, semantics, syntax, alternatives). Market data sources are always known up-front regarding their identity, trustworthiness, location, syntax, and semantics from an ESP's perspective. Another key difference is the availability of alternative sources for the same data: in market data scenarios only a limited number of alternative sources offer the same information while the same type of sensor reading might be available from a multitude of sources in IoT scenarios.

Data (volume, variety, velocity, veracity, quality). In market data setups, the volume of data published by each source is usually very high compared to data source in IoT setups that need to take resource constraints into account. Same for the velocity and veracity that are always extreme in market data scenarios, even when not considering algorithmic trading. Load patterns are generally known in advance in market data setups (but not the amplitude) while mobility, churn, and other factors lead to unpredictable load patterns in IoT scenarios as a norm. Variety is high in both scenarios.

Summarizing, requirements for IoT data processing differ significantly from those relevant for market data. Hence, benchmarks tailored to IoT scenarios are not suited to stress-test ESPs in market data scenarios. Market data processing scenarios are characterized by a trusted and stable setup with resources being available in abundance. This allows to unburden the ESP from much unsupervised ad-hoc adaptation and safeguarding capabilities needed in more dynamic, untrusted, and resource-constrained IoT scenarios. Conversely, ESPs in market data scenarios must be able to deal with massive load spikes in addition to a high baseline of volume, variety, and veracity to be consumed from each source.

2.2 Related work

Building scalable and realistic benchmarks for event-processing systems remains an active area of research. In the context of this paper we categorize relevant benchmarks in three categories with regard to their workload model.

Application-specific benchmarks, such as the Pairs benchmark [10], Linear Road [2], or Sparkbench [9] allow to manually define workloads or use predefined data to fine-tune parameters of a specific target system. For ESPs, these benchmarks also tend to aim more on the correctness of a CEP engine's reasoning and hence focus on offering semantically rich and challenging workloads, with volume becoming a secondary concern.

Benchmarking frameworks, such as CityBench [1], RIoTBench [13], bench [14], BigDataBench [15], or the framework by Karimov et al. [8] are generally workload-agnostic. They focus on providing

scalable, versatile architectures to replay real data or to plug-in third-party workload generators.

Scenario-driven benchmarks, such as SpecJMS [12] or the TPC* benchmarks [11] use complex models to simulate semantically correct interactions between actors based on a reduced model of reality. These benchmarks scale by changing the actors' population or their interactions' intensity. Consequently, the resulting workload represents the effect of these interactions. Scenario-based benchmarks require a simplified yet representative model of the real-world scenario to be implemented. Unfortunately, none is available for market data processing.

We focus on performance benchmarking and not on the reasoning capabilities of a CEP. Consequently, benchmarks of the first category do not cover our requirements. In previous work [3] we already contributed to the second category of benchmarks by providing an open-source framework that allows to define simple workload patterns, plug-in workload generators or replay recorded data. Building a scenario-based benchmark for financial data feed systems similar to those of the third category is our long-term goal. However, defining a reduced yet accurate model of the interaction patterns of actors on the financial market with semantically correct cause-effect relationships is out of scope of this work. In sum, our work on a simple yet realistic workload model to simulate key characteristics of high-volume market data feeds as presented here is not yet covered by existing work.

3 A MODEL FOR FINANCIAL DATA

This section introduces the real-world data we base this work on (Sec. 3.1), the type of analysis we used to extract relevant features (Sec. 3.2), and the model we derived from the analysis (Sec. 3.3).

3.1 Data sets used for analysis

We use four complementary sets of real-world data made available by Infront Financial Technology GmbH (hereafter Infront) as shown in Fig. 1. Three data sets are used to derive the model while the fourth (recently published) data set is used to evaluate the model.

Consolidated feed sample [CFS]. This data set contains 308,290 events notifications published by 18 sources over the period of one minute and 3.760 seconds in 2021; the data set covers all event types processed at the highest available granularity for 30,918 symbols and seven instrument types (bonds, certificates, equities, exchange traded funds, indices, market depth). This data set is being used to analyze attributes and their completeness across instrument types, sources, and symbols.

Rate statistics sample in seconds [RSS]. This data set of 17,286 data points contains statistics on rates per second for six sources (exchanges Frankfurt, Tokyo, London, Sydney, Hong Kong, NASDAQ) captured over two average trading days. The data covers all event types for a representative mix of instrument types and symbols traded on these exchanges. This data is being used to identify rate patterns over time at a high granularity and per source in order to characterize typical rate patterns per source. Available at [4].

Infront daily rate statistics [DRS]. This data set of 451,072 data points contains daily statistics about all event types processed in the ticker plant of Infront from all 500+ global sources for the duration of 15 months covering 14 instrument types and 29 million symbols. This data is used to identify rate patterns over time per source and instrument type to characterize and categorize sources.

DEBS Grand Challenge 2022 Trading Data [GC22]. This data set contains 289 million event notifications of the highest granularity (tick data) from three sources covering the period of one week and two instrument types (equities and indices). This data set is used to verify our model. Available at [7].

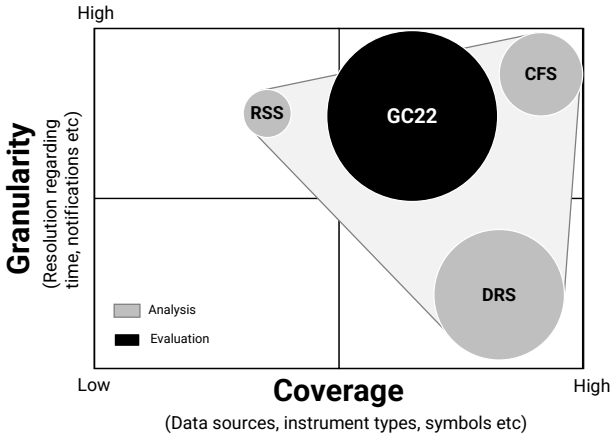


Figure 1: Complementary data used for analysis (gray) and evaluation (black); diameters indicate size of set (not in scale).

3.2 Data analysis

We analyzed the data in Sec. 3.1 to derive prominent characteristics. The analysis has been verified by interviews with domain experts.

Exchange categorization. Exchanges are the sources of financial data and different exchanges exhibit different behaviors in terms of the data they produce. The first step of our analysis consists in identifying classes of exchanges that present similar profiles in terms of data generation. The rationale is that the class of exchange may affect any subsequent analysis of data: by identifying classes of exchanges with homogeneous behavior, we can analyze each class in isolation and improve the accuracy of the results we derive.

To identify classes of exchanges, we adopt a clustering algorithm. We observe that the main distinguishing factor that characterizes exchanges is the distribution of instrument types for the events they generate. Accordingly, we use the frequencies of occurrence of events belonging to each instrument type as features. As the data sets contain 14 instrument types, each exchange is identified by a point in a 14-dimensional feature space, and the goal of the clustering algorithm is to find groups of exchanges that are close to each other in the feature space. We adopt a k-means clustering algorithm: the algorithm takes in input a positive number k and returns (i) k points in the feature space that identify the center of k groups (*centroids*); (ii) association of each element in the data set (exchange in our case) to one group. The algorithm is iterative: it starts from an initial set of centroids and refines them at each iteration. As we do not know the number of classes upfront, we apply the algorithm for different values of k; for each value, we compute the quality of the solution

based on a measure of similarity of the elements within each cluster (the sum of square distance between each element and the centroid of its cluster); we select the smallest number of k after which the quality of the solution does not improve any further. As the final solution depends on the initial selection of centroids, for each value of k we apply the algorithm 5 times starting from different centroids, and we evaluate the quality as the average over the 5 executions.

As a result, we identified 12 classes of exchanges: 3 of them only produce instruments of one type; 4 of them produce 2 types of instruments (one type accounts for over 90% of the events); 2 of them produce 3 types of instruments (one type accounts for over 80% of the events); the remaining classes produce between 4 and 8 different types of instruments.

Distribution of symbols. Data in the financial market is highly skewed. Certain symbols appear much more frequently than others, for instance due to a very high number of daily transactions involving a small number of large companies. Trying to identify classes of symbols that present similar behaviors (as we did for the exchanges) (i) is difficult, as individual symbols may change their behavior over time in an unpredictable way due to external conditions that are not captured in the data sets and hard to identify; (ii) is error-prone, as a wrong classification may lead to an imprecise characterization of the workload; (iii) may lead to an overly complicated definition of the workload, which is not in line with the principle of simplicity we are following in our work.

Accordingly, in our analysis we proceed as follows. (A) We compute the distribution of symbols according to their frequency at which they appear in the data sets. We consider the set of symbols that appear in different instrument types as disjoint. (B) We observe that the distribution can be well approximated by a *Pareto distribution*. (C) Pursuing the idea of simplicity, we derive a coarse grained discretization of the distribution, identifying n classes of frequencies and assuming that all symbols that belong to the same class appear with the same frequency. Specifically, to build our model we considered $n = 4$ classes of symbols, with each class accounting for 25% of input events. The first class contains very few symbols that appear very frequently, down to the last class that includes many symbols, each of them appearing in only few events. As we will discuss in Sec. 3.3, the above discretization enables the definition of a simple model that can generate realistic workloads with low computational complexity.

Temporal distribution. The temporal distribution of events plays a primary role in ESP systems and needs to be carefully modeled. For instance, handling a sudden increase in the input rate of events may be problematic for some systems and may lead to undesirable consequences such as an increased response time. We analyze temporal distribution by looking at two levels of granularity: at a coarse granularity we consider the *daily profile* of exchanges, while at a fine granularity we consider the *timestamp distribution*.

The daily profile looks at the average rate of data generated by a given exchange for each minute within an entire day. All the exchanges we analyzed present recurring temporal patterns: (i) they only produce traffic during working hours; (ii) they present two spikes at the beginning and at the end of the working day. Based on this observation, we generated a *generalized daily profile*, presented in Fig. 2, by aligning and averaging (with a granularity of 10 minutes) the daily profiles of different exchanges.

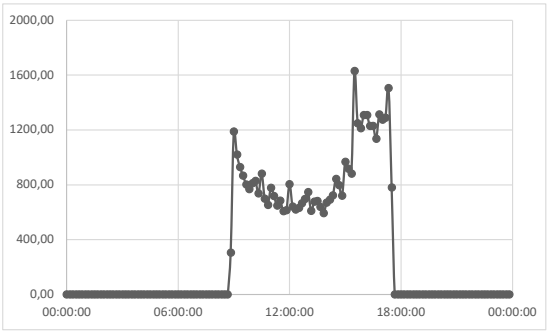


Figure 2: Generalized daily profile.

Individual exchanges may differ from the generalized daily profile with respect to three axis: (i) beginning of the working day (e.g., due to different time zones); (ii) duration of the working day; (iii) absolute rate. As we detail in Sec. 3.3, our model introduces parameters to control the three axis by (1) shifting, (2) scaling horizontally and (3) vertically the generalized daily profile.

The daily profile defines the average rate of data produced by a given exchange with a granularity of 10 minutes, that is, it captures rate fluctuations that occur daily. However, we observe that the distribution of events also presents fluctuations at a much smaller scale (millisecond) that affect the distribution of events. We capture these fluctuations by studying the distribution of the distance in time (timestamp difference) between two consecutive events coming from the same exchange. Our analysis shows that the distribution is very skewed and can be well approximated by an exponential function with rate $\lambda = 1/\mu$, where μ is the average data rate (as defined by the daily profile of the exchange).

Completeness. The quality of data can play an important role for an ESP system, as it may affect the way in which individual events are processed (or maybe discarded) [6]. Accordingly, we conduct a detailed analysis for the completeness of event attributes as follow. First, we split the data sets by exchange type, as we observed that for any given exchange type some attributes are mandatory (always present) and some attributes are not compatible (always absent). Then, for any remaining attribute, we compute the frequency of occurrence in a given exchange type. We use these simple statistics in our model to determine the probability that a given attribute is empty. As a final remark, we also studied the correlation of missing values, to understand if there were pairs or sets of attributes having a high probability of being all present or all absent. However, this study did not bring conclusive results and we opted for treating each attribute independently from the others.

3.3 Model definition

Our model directly derives from the analysis in Sec. 3.2 and produces a realistic workload by simulating multiple sources (exchanges).

Let us start from the characterization of individual exchanges. An exchange e can be fully specified through 5 parameters.

(1) *Opening time* t_e^0: the UTC time at which the working day starts for e. Different exchanges opens at different times depending on their time zone. This parameter is used to shift the daily profile curve.

(2) *Day length* ℓ_e: the length of the working day for e. This parameter is used to scale the daily profile curve horizontally (shrink it for shorter working days or expand it for longer working days).
(3) *Maximum rate* α_e: the maximum input rate of events for e. This is used to scale the daily profile curve vertically.
(4) *Category* c_e: the category of e, among the 12 categories we identifies with our cluster analysis. The category influences the behavior of the exchange in terms of data generation.
(5) *Symbols* S_e: the set of symbols produced by e. This parameter allows the developers to control how the events produced by different exchanges overlap in terms of symbols.

As discussed in Sec. 2.1, exchanges are organized into feeds and a feed publishes events coming from one or more exchanges. Our model lets users define multiple exchanges (compiling the 5 parameters above for each of them) and associate them to feeds. Thus, we consider three additional parameters:

(6) *Feeds* F: set of feeds to be considered.
(7) *Feed exchanges* E_f: for each feed $f \in F$, the set of exchanges associated to that feed.
(8) *Omission rate* ρ_f: for each feed $f \in F$, the probability that an entire event coming from that feed is lost.
(9) *Latency* λ_e^s: for each $e \in E_f$ and for each symbol $s \in S_e$, the latency of communication for symbol s between the exchange e and the feed f (mean and standard deviation). In other words, the distance in time between the instant when the event is generated (as defined by its timestamp) and the instant when the event is appended to the feed. With this parameter, users can introduce a degree of reordering between events of the same symbol received by an ESP system.

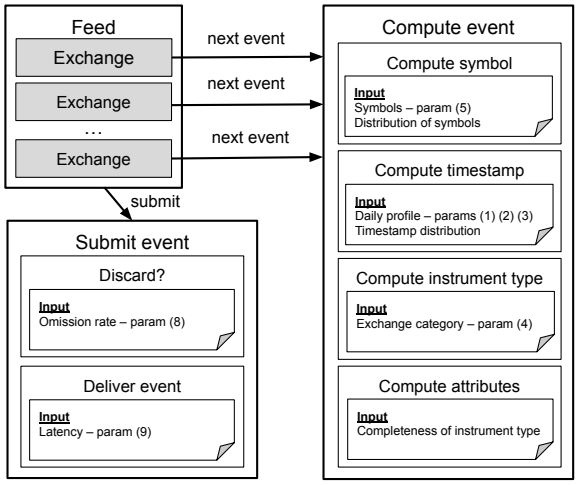


Figure 3: Workflow of the model

Starting from this small number of parameters, the model defines the events to be produced following the workflow presented in Fig. 3 and discussed below.

(A) At any given point in time of the simulation, the model determines the average rate of each exchange e based on its daily profile, based on the generalized daily profile and t_e^0, ℓ_e, α_e.
(B) For a given exchange e, the model determines the symbol of the next event using the list of symbols – parameter S_e and the distribution of symbols – organized into discrete classes of frequencies, as discussed in Sec. 3.2.
(C) For a given exchange, the model determines the instrument type of the next event by using the exchange category – parameter c_e.
(D) For a given exchange e, the model determines the timestamp of the next event using the distribution of time differences – exponential distribution, as discussed in Sec. 3.2.
(E) The model determines if an event has missing attributes bases on the completeness characteristics of its instrument type, as discussed in Sec. 3.2.
(F) The model determines if an event has to be discarded based on the omission rate of the feed f producing it – parameter ρ_f.
(G) The process is applied to each feed $f \in F$ and for each exchange $e \in E_f$. The model produces events for a given symbol s within an exchange e in strict timestamp order, while events from different symbols may be received out of order, depending on the latency of communication – parameter λ_e^s.

4 EVALUATION

We evaluate the fit of our model qualitatively by comparing some features of generated data with the data set *Trading Data* [7] that has been published as part of the DEBS Grand Challenge 2022 (hereafter [GC22]). As outlined in Sec. 3.1, [GC22] complements the combined data used in this work for analysis and model derivation. Quantitative analysis of the generated workload and its impact on state-of-the-art ESPs is ongoing research work. With respect to the four aspects of our analysis discussed in Sec. 3.2, we focus on *symbol distribution* and *temporal distribution* for this evaluation. We could not replicate the study of *exchange categorization* due to the limited number of exchanges available in [GC22] (only three). Same for the study of *completeness* because the attributes contained in [GC22] have been purged to a very small subset (only 39) of the 2,500+ attributes available to us in [CFS]. Hence, we focus on comparing data generated by our model with the [GC22] data set in regard to (i) daily profile pattern, (ii) distribution per symbol, and (iii) orders of magnitude for update rates per exchange type as Frankfurt/ETR in [GC22] resembles a small, Amsterdam/NL a medium, and Paris/FR a large exchange regarding the overall traffic they contribute to the data set.

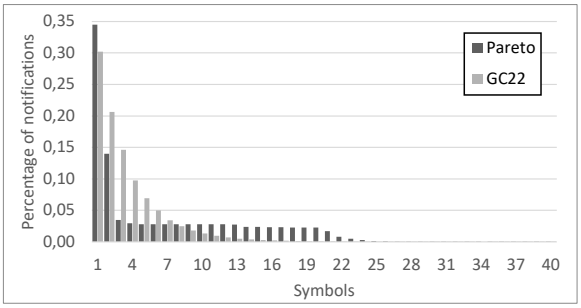
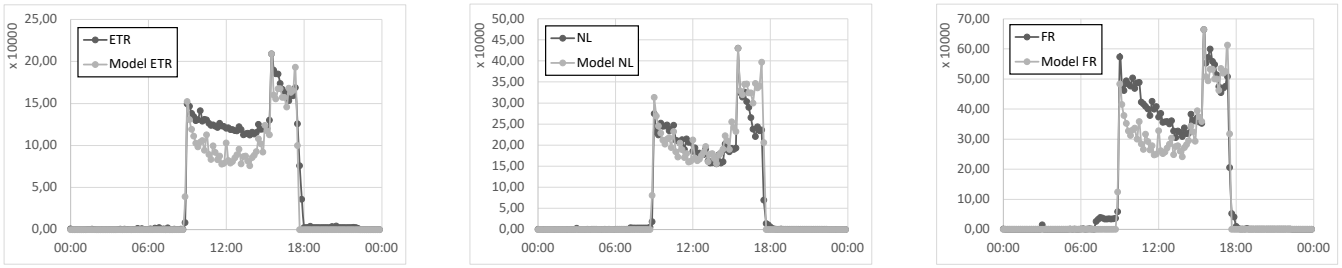


Figure 4: Distribution of symbols: Pareto vs. [GC22]

First, we compute the distribution of symbols in [GC22]. Fig. 4 shows the distribution when splitting symbols into 40 bins based



(a) Frankfurt/ETR exchange **(b) Amsterdam/NL exchange** **(c) Paris/FR exchange**

Figure 5: Daily pattern: model vs. [GC22]

on the number of event notifications they generate. As Fig. 4 shows, the first bin alone ($1/40^{th}$ of all symbols) already produces 35% of all event notifications, and the second bin produces about 15% of all event notifications. After that, half of the bins produce roughly the same number of events (about 3%) and the remaining bins contribute to the number of event notifications only marginally. Fig. 4 also compares the distribution of symbols with a Pareto distribution (shape set to 16), which is the one we build our model on. As Fig. 4 shows, the distribution approximates the trend in [GC22], but in [GC22] intermediate bins (between 4 and 20) produce a larger and more uniform number of event notifications. This confirms the validity of splitting symbols into few classes and consider a uniform distribution within each class, as our model suggests.

Second, we study the temporal distribution of event notifications produced by the three exchanges in [GC22]. For each exchange, [GC22] contains one week of data and in our analysis we consider the average number of events produced in a given time window over the 7 days of the week. Fig. 5 compares the daily pattern produced by our workload generator with respect to real data from Frankfurt/ETR – Fig. 5a, Amsterdam/NL – Fig. 5b, Paris/FR – Fig. 5c. As different exchanges produce different volumes of events, we scaled our model using parameter α_e based on the maximum rate observed in each exchange: we obtained $\alpha_{ETR} = 128$ for Frankfurt/ETR, $\alpha_{NL} = 264$ for Amsterdam/NL, $\alpha_{FR} = 407$ for Paris/FR. A qualitative analysis shows a clear similarity between the real and the modeled patterns. Despite the expected differences across exchanges, our model mimic the expected daily patterns with a simple approach that can be easily implemented at scale.

In summary, the initial assessment using the [GC22] data set confirms our model's capability to reproduce key characteristics of market data feeds, such as the highly skewed distribution of symbols and the variable generation rates over a day.

5 CONCLUSIONS

Performance benchmarks using configurable, reproducible, and scalable workloads are essential to properly assess and fine-tune modern ESPs. This paper presented two novel contributions from an ongoing project on the development of a benchmark specific to financial market data processing: (i) a detailed statistical analysis of real-world market data provided by a leading provider of financial data, and (ii) a simple yet accurate workload model derived from this analysis. Our model simulates high-volume workloads that reflect the specifics of market data, and has been evaluated using the data set *Trading Data* recently published as part of the DEBS 2022 Grand Challenge. We are currently working on a reference implementation of the model in our open-source benchmarking framework *wrench* [3]. Future work will focus on fine-tuning the parameters presented here and adding support for semantic dependencies between symbols.

ACKNOWLEDGMENTS

The authors would like to thank the three anonymous reviewers for their valuable feedback on this work and Fredrik Koch of the Infront group for his sponsorship.

REFERENCES

[1] M. I. Ali, F. Gao, and A. Mileo. 2015. CityBench: A Configurable Benchmark to Evaluate RSP Engines Using Smart City Datasets. In *ISWC'15*. Springer.
[2] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. 2004. Linear Road: A Stream Data Management Benchmark *(VLDB '04)*. VLDB Endowment.
[3] M. Coenen, C. Wagner, A. Echler, and S. Frischbier. 2019. Benchmarking Financial Data Feed Systems. In *DEBS'19*. ACM.
[4] S. Frischbier, M. Paic, A. Echler, and C. Roth. 2019. *Daily load patterns of six global exchanges.* https://doi.org/10.5281/zenodo.6381970
[5] S. Frischbier, M. Paic, A. Echler, and C. Roth. 2019. Managing the Complexity of Processing Financial Data at Scale - An Experience Report. In *CSDM'19*. Springer.
[6] S. Frischbier, P. Pietzuch, and A. Buchmann. 2014. Managing expectations: Runtime negotiation of information quality requirements in event-based systems. In *ICSOC '14*. Springer. https://doi.org/10.1007/978-3-662-45391-9_14
[7] S. Frischbier, J. Tahir, C. Doblander, A. Hormann, R. Mayer, and H.-A. Jacobsen. 2022. *DEBS 2022 Grand Challenge Data Set: Trading Data.* https://doi.org/10.5281/zenodo.6382482
[8] J. Karimov, T. Rabl, A. Katsifodimos, R. Samarev, H. Heiskanen, and V. Markl. 2018. Benchmarking Distributed Stream Data Processing Systems *(ICDE '18)*.
[9] M. Li, J. Tan, Y. Wang, L. Zhang, and V. Salapura. 2017. Sparkbench: a spark benchmarking suite characterizing large-scale in-memory data analytics. *Cluster Computing* 20, 3 (2017).
[10] M. R.N. Mendes, P. Bizarro, and P. Marques. 2013. Towards a Standard Event Processing Benchmark. In *ICPE '13*. ACM.
[11] R. Nambiar, M. Poess, A. Masland, H. R. Taheri, M. Emmerton, F. Carman, and M. Majdalany. 2013. TPC Benchmark Roadmap 2012. In *Selected Topics in Performance Evaluation and Benchmarking*. Springer.
[12] K. Sachs, S. Kounev, J. Bacon, and A. Buchmann. 2007. Workload Characterization of the SPECjms2007 Benchmark. In *Formal Methods and Stochastic Models for Performance Evaluation*. Springer.
[13] A. Shukla, S. Chaturvedi, and Y. Simmhan. 2017. Riotbench: An IoT benchmark for distributed stream processing systems. *Concurrency and Computation: Practice and Experience* 29, 21 (2017).
[14] T Treat. 2017. bench - A generic latency benchmarking library. https://github.com/tylertreat/bench. [Online; accessed 2022-03-22].
[15] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu. 2014. BigDataBench: A big data benchmark suite from internet services *(HPCA '14)*.