Poster Abstract: A Glimpse of the Matrix

Scalability issues of a new message-oriented data synchronization middleware

Florian Jacob
Karlsruhe Institute of Technology
florian.jacob@kit.edu

Jan Grashöfer
Karlsruhe Institute of Technology
jan.grashoefer@kit.edu

Hannes Hartenstein
Karlsruhe Institute of Technology
hannes.hartenstein@kit.edu

Abstract

Matrix is a new message-oriented data synchronization middleware, used as a federated platform for near real-time decentralized applications. It features a novel approach for inter-server communication based on synchronizing message history by using a replicated data structure. We measured the structure of public parts in the Matrix federation as a basis to analyze the middleware's scalability. We confirm that users are currently cumulated on a single large server, but find more small servers than expected. We then analyze network load distribution in the measured structure and identify scalability issues of Matrix' group communication mechanism in structurally diverse federations.

ACM Reference Format:
Florian Jacob, Jan Grashöfer, and Hannes Hartenstein. 2019. Poster Abstract: A Glimpse of the Matrix: Scalability issues of a new message-oriented data synchronization middleware. In *20th International Middleware Conference Demos and Posters (Middleware Demos and Posters '19), December 9–13, 2019, Davis, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3366627.3368106

1 Introduction

Matrix[1] is a federated middleware for decentralized applications, e.g. federated instant messengers. It is based on a client-server architecture where independent servers with limited mutual trust cooperate. It provides topic-based publish-subscribe access on an eventually-consistent database of messages and state changes. Topics are called *rooms* in Matrix. For their users, servers broadcast messages to other servers in a room, and receive messages and history from those servers. Messages and state changes form a partial order in the replicated per-room data structure from which the current state is derived. At present, the public federation is centered around one large server with about 50 000 daily active users. However, Matrix is growing fast and is intended to be more decentralized. The French government is deploying a large private federation between its ministries for about 6 million civil servants[2]. We crawled parts of the public federation and observed an imbalanced network. Based on our measurements, we show scalability issues of the current message distribution algorithm. We present ideas for a scalable replacement that allows leveraging Matrix' combination of messaging and storage e.g. in the Internet of Things.

2 Related Work

Distributed variants of message-oriented middlewares [1] like RabbitMQ [3], and distributed storage systems like Cassandra [2], are decentralized, i.e. every node operates indepentently. Requests are equally distributed on all nodes, scaling linearly with the number of nodes. This distribution requires that all nodes are controlled by a single entity, i.e. it is not byzantine fault tolerant and can not be operated in a federation with limited trust. XMPP[3] as message-oriented middleware supports public federation, but its group communication relies on centralized, trusted parties. Matrix, however, is a promising new type of middleware for decentralized applications as it combines distributed messaging and storage, while supporting federations of limited trust.

3 Measuring the Federation Structure

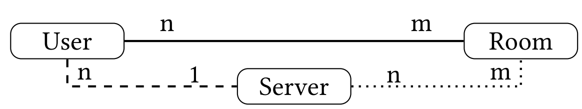


Figure 1. Relations in the network structure

We measured the relations between users, rooms and servers in public rooms (c.f. Fig. 1) of the public Matrix federation using a crawler bot[4]. The bot observed a part of the public network: It joined 798 rooms, seeing 131 463 users on 2003 different servers. From Fig. 2 we see that users are cumulated on few servers, but there are more small servers than expected. Fig. 3a shows that most rooms have very few participating servers, but there are some rooms with many. The few small rooms in Fig. 3b are explained by the fact of looking at public rooms, which typically have more users. The regularities in Fig. 2 & Fig. 3c enable the algorithmic

Middleware Demos and Posters '19, December 9–13, 2019, Davis, CA, USA

ACM ISBN 978-1-4503-7042-4/19/12.
https://doi.org/10.1145/3366627.3368106

[2] https://fosdem.org/2019/schedule/event/matrix_french_state/
[2] https://matrix.org, https://matrix.org/docs/spec/
[3] https://xmpp.org/, https://xmpp.org/extensions/xep-0045.html
[4] https://github.com/kit-dsn/dsn-traveller

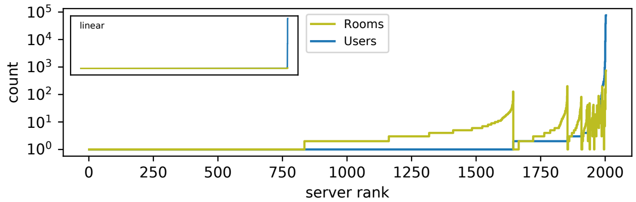
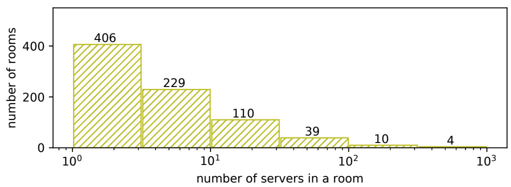
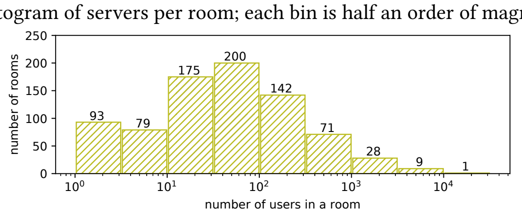


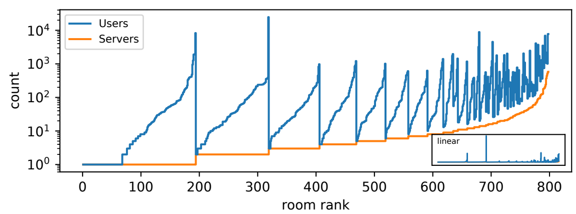
Figure 2. User and room count per server[6] on 2018-07-25.



(a) Histogram of servers per room; each bin is half an order of magnitude



(b) Histogram of users per room; each bin is half an order of magnitude



(c) Server and user count per room[6]

Figure 3. Observed room composition on 2018-07-25

generation of larger networks with similar characteristics to make predictions on Matrix’ scalability. The anonymized raw data is freely available for further research[5].

4 Efficiency & Scalability Issues

Matrix’ group communication mechanism is inherently asymmetrical between transmissions and receptions: In a room with n servers, the sending server does $n-1$ individual transmissions to each receiving server. With the frozen snapshot of the Matrix federation gathered by the crawler bot and a fixed average message rate per user, we calculated Fig. 4 which shows that the network load from sending is strongly concentrated in large servers, while receiving load is more distributed. With a network structure as found in the public federation, the single largest server sends nearly 90 % of all messages, and is part of nearly 45 % of all message transfers. The reason for this load centralization lies in unequally distributed users: Most events are generated on large servers, which then have to be distributed to many small

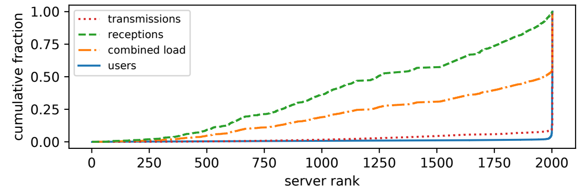


Figure 4. Cumulative fraction of network load per server[6]

servers, while small servers reach many users with a single transmission to a large server. Decreasing the degree of centralization will worsen the load distribution during the transition phase when more users set up small servers but the majority is still cumulated. When reaching full decentralization, the efficiency benefit of reaching multiple users by a single transmission vanishes. These aspects affect Matrix’ scalability and hinder the future growth of the public federation. Due to the limited trust between servers, load balancing can not be achieved by moving users away from busy servers as with non-federated middlewares. Matrix instead needs a group communication mechanism that scales regardless of topology and usage pattern. We envision an adaptive message routing algorithm, which utilizes the available room structure information to dynamically select a suitable per-room group communication mechanism, to deliver Matrix’ promising combination of federation, messaging and storage to decentralized applications at any scale.

5 Conclusion & Future Work

We presented a partial measurement of the network structure of the public Matrix federation. The crawler bot and raw data is provided to the community. We identified scalability issues in the group communication mechanism of the Matrix middleware in form of load centralization in structurally diverse federations, which cannot be mitigated by rebalancing users due to the limited trust between Matrix servers. As future work, we plan to solve the underlying problem using room structure-adaptive message routing algorithms. To test their scalability, we will model and simulate larger federations with similar characteristics using the found regularities in the measurements. We will continue to observe the evolution of the public Matrix federation towards more decentralization. An extended tech report of this work is available at https://arxiv.org/abs/1910.06295.

References

[1] Guruduth Banavar, Tushar Chandra, Robert Strom, and Daniel Sturman. 1999. A case for message oriented middleware. In *Distributed Computing*. Prasad Jayanti, (Ed.) Springer Berlin Heidelberg, Berlin, Heidelberg, 1–17. ISBN: 978-3-540-48169-0.

[2] Avinash Lakshman and Prashant Malik. 2010. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44, 2, 35–40.

[3] Maciej Rostanski, Krzysztof Grochla, and Aleksander Seman. 2014. Evaluation of highly available and fault-tolerant middleware clustered architectures using rabbitmq. In *2014 Federated Conference on Computer Science and Information Systems*. IEEE, 879–884.

[5] https://github.com/glimpseofthematrix/networkgraph

[6] Server (resp. room) rank: Each particular server (resp. room) is assigned with a different, ascending rank based on its number of users (resp. servers). More users (resp. servers) result in a higher rank.