

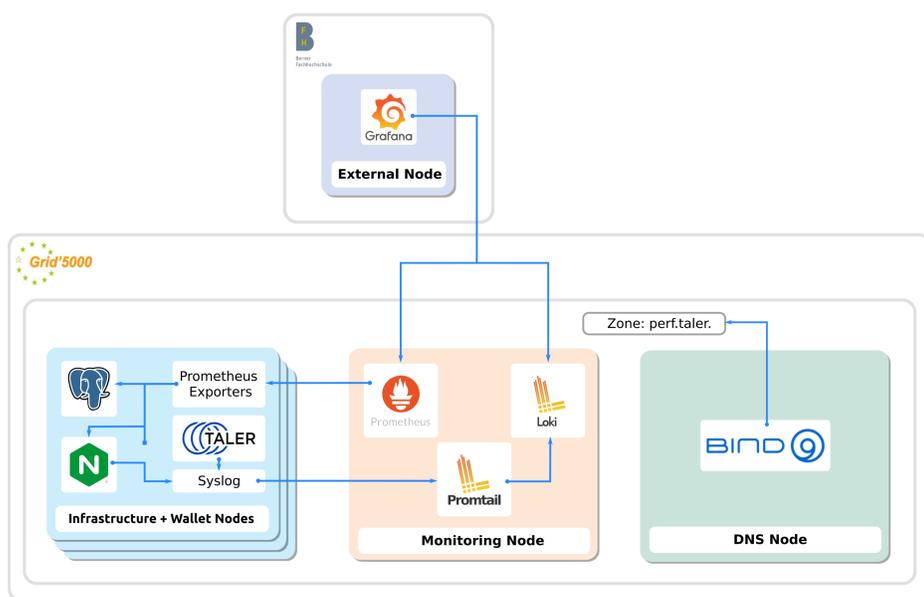
GOALS

- Assess and improve GNU Taler scalability
- Develop setup that collects necessary information
- Provide automated, reproducible experiments
- Realistic distributed deployment

CHALLENGES

- Learn to use Grid'5000 and associated tools
- Collect and manage performance data
- Identify and resolve scalability bottlenecks in GNU Taler and underlying technologies

DEMO SETUP



RESULTS

- Successfully deployed GNU Taler for distributed experiments in Grid'5000 using jFed/Kameleon for deployment and Prometheus, Loki and Grafana for data collection
- Identified and resolved performance problems in Taler client (load generation), Taler core logic and Postgres database use
- Suggestions for improvements and/or bug fixes in jFed and Grid'5000

MORE RESULTS

Result of first improvements in the load generating clients (Better handling of the large file based Database - IndexedDB)

Before:

Periodic client restarts to prevent performance loss of the growing DB file

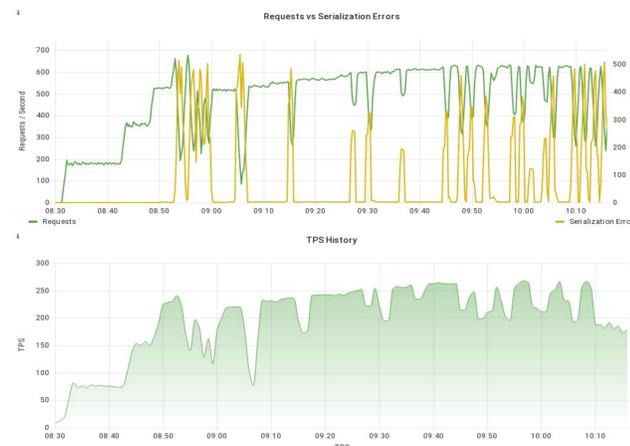


After:

New client implementation purges DB after each iteration



Current state showing the impact of serialization errors on the number of handled requests and thus the total Transaction (Withdraws and Payments) per second (TPS). Some of those are fixed others remain a target for the future.



CONCLUSIONS

- Encountered surprising difficulties in our load generation code simulating Taler clients
- Grid'5000 is suitable for large-scale performance evaluation of network software
- jFed can be a bit confusing as it generalizes over various testbeds. The general-purpose UI has many features that do not apply to all testbeds (are not masked, but do not work on Grid'5000)

POST MORTEM

- Database seems to currently be the primary bottleneck, major revision of the schema to enable better sharding was implemented but could not yet be evaluated
- Without major improvements to the load generation logic, it seems likely that the next limit will be the number of clients/users we can simulate with Grid'5000