Towards Performance Modeling of Hyperledger Fabric

Extended Abstract

Imre Kocsis[1], András Pataricza[1], Miklós Telek[2], Attila Klenik[1], Flórián Deé[1], Dávid Cseh[1]
[1]Dept. of Measurement and Inf. Systems and
[2]Dept. of Networked Systems and Services,
Budapest University of Technology and Economics, Budapest, Hungary

Abstract. *Blockchain technologies target domains where strict performance guarantees are required and formal Service Level Agreements are commonplace. Designing for performance targets in a trustworthy manner requires performance models; we present a performance characterization approach that addresses the complexity of Blockhain technologies. We apply the methodology to Hyperledger fabric 0.6. We also discuss the architectural shortfalls we uncovered in a systematic way.*

Blockchain technologies and performance

Initially motivated by the success of Bitcoin [1], the world is realizing the immense potential of so-called Blockchain systems. Blockchain technologies implement a shared ledger of transactions across a peer to peer system; the ledger is kept in synchrony across the peers by the virtue of system-wide consensus on the transactions, without a single point of trust. The “Blockhain” name comes from the way these technologies store their immutable and non-repudiable shared transaction log: as a chain of signed transaction blocks. The vanguard of Blockchain technologies is now “programmable”: these systems support “smart contracts” – user-defined transaction logic that can be far beyond the complexity of passing units of cryptocurrency. The applications that are being introduced reach from the financial world through enterprise asset management and business process automation to the Internet of Things and Cyber-Physical Systems. Blockchain standardization has begun [2] and open source global collaborations have been formed. Importantly, the Linux Foundation hosts the Hyperledger project [3]. The project is an umbrella for multiple distributed ledger technologies, among which the most mature is Hyperledger fabric (previously IBM Open Blockchain).

Many intended applications of Blockchain technologies require performance guarantees – irrespective of whether a Blockchain-based solution competes with legacy systems and approaches or offers truly novel functionality. Famously, Bitcoin throughput is in the order of 10 transactions per second and single transaction, single confirmation latency is at 10 minutes or more [4], which is inadequate for many applications. Other Blockchain systems, especially permissioned ones, aim at orders of magnitudes higher performance targets. However, to guarantee performance and to design deployments against performance targets, performance models are required that map workload, configuration and the characteristics of the operational environment into the “engineered performance capacity” in a trustworthy manner.

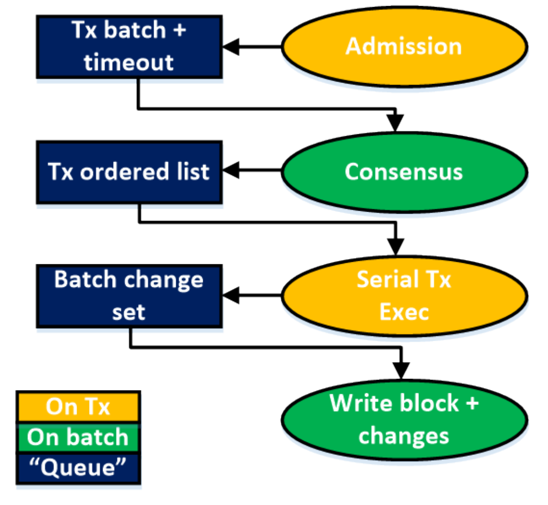

Figure 1. Hyperledger fabric high-level operational logic

A measurement-based approach

Functionally, smart contract enabled blockchains are complex systems. They accept transactions; perform distributed consensus; execute rather arbitrary transaction logic based on the current ledger state (backed by the current blockchain state); and in addition to local blockchain management, also maintain distributed ledger consistency. For instance, in Hyperledger fabric, incoming requests are synchronously accepted and batched; batches undergo consensus using the Practical Byzantine Fault Tolerance (PBFT) protocol [5] to globally establish request sequence numbers; each node executes ordered requests strictly serially, using their respective smart contract (called "chaincode" in Hyperledger fabric); and finally, changes are written to the blockchain block-wise (blocks correspond the original batches). See Figure 1.

Even in the heavily simplified view above, each of the activities can have very complex performance characteristics on its own right - and there are potentially feed-backs between the activities (we do not discuss these). These characteristics make a directly analytic approach towards Blockchain performance modelling very hard. Consequently, our approach towards characterizing Blockchain performance is the following.

1) **Measurement**: perform benchmark-like measurements with load and configuration sweeps in an operational envelope that is representative for a use case.
2) **Data analysis**:
   a) determine the qualitative performance characteristics of the system as well as the individual components, and
   b) determine bottlenecks and hot spots.
3) **Targeted sensitivity analysis**: perform experimental sensitivity analysis for the components that contribute to the bottleneck.

Finally, the results of what is essentially performance model structure discovery can be applied to perform analytical compositional modelling (what is actually outside of our research scope). The process is summarized by Figure 2.

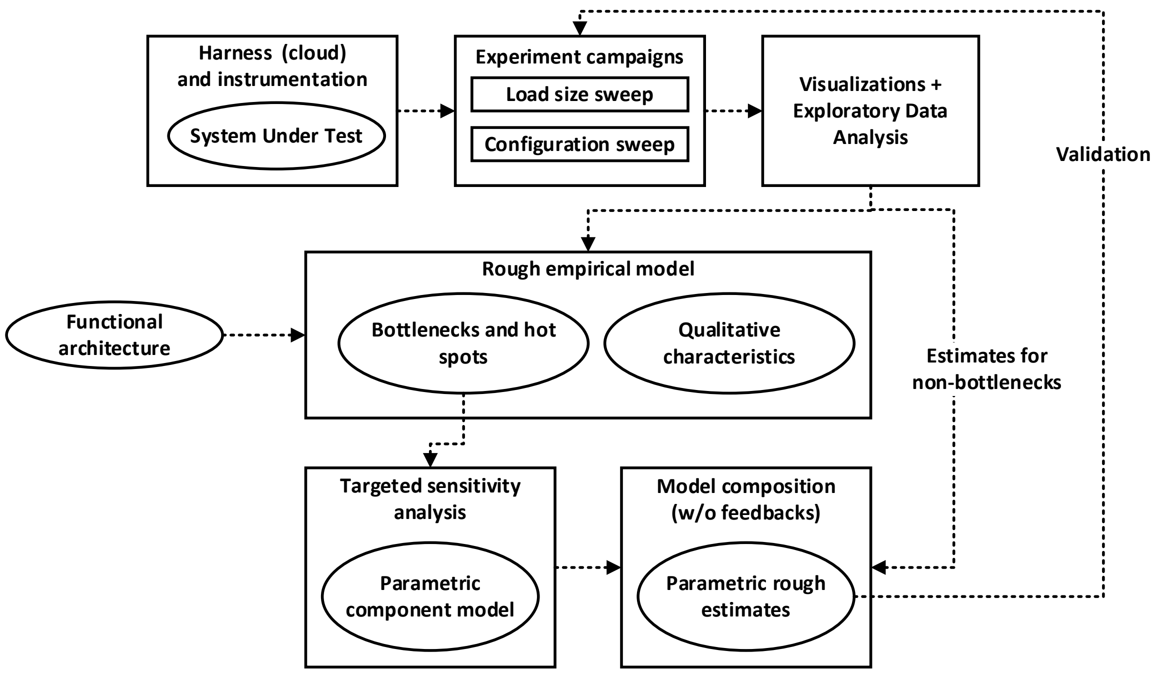
Harness (cloud) and instrumentation
System Under Test
Experiment campaigns
Load size sweep
Configuration sweep
Visualizations + Exploratory Data Analysis
Validation
Rough empirical model
Functional architecture
Bottlenecks and hot spots
Qualitative characteristics
Estimates for non-bottlenecks
Targeted sensitivity analysis
Parametric component model
Model composition (w/o feedbacks)
Parametric rough estimates

Figure 2. Performance characterization approach

Application to Hyperledger fabric

We have applied the above approach to the 0.6 branch of Hyperledger fabric. (The redesigned 1.0 goes to alpha in March, 2017.) In the current absence of open, representative performance requirement sets and workloads, we have started with requirements that in our experience are “reasonable” for performance-critical systems. In this extended abstract, we omit the details of the process (from campaign definitions to the details of data analysis) and only outline our key findings.

Hard cap on throughput

A natural and usual requirement towards performance-critical systems is that engineering their deployment and configuration are the main vehicles of influencing their performance capacity. This is not the case for fabric 0.6; there is a cap around 300 Tx/s, even for completely fault-free scenarios. We have found that the issue is the strictly sequential chaincode execution of the ordered requests. This is not an absolute necessity – in theory, e.g. consensus could become problematic before sequential execution reaches its limit; however, the Docker-based chaincode execution also uses an RPC mechanism that in its default configuration is very slow (see Table 1).

	Typ. serv. time	Throughput
Tx admission	**1.5 ms/Tx** *“<few> ms/batch”*	Cap not known, **Highly parallel**
Batch consensus	Wait: **0..1000ms/Tx** *“<= 1s/batch”*	**Highly adaptable** Depends on: number of peers, network, …
	Cons.: *“0.15ms/Tx”* **45ms/batch**	
Tx execution	**1.8..2.5 ms/Tx** **540..750ms/batch**	**Sequential, overheads, nontrivial payloads?**
Batch block creation	*“0.01..0.14ms/Tx”* **3..42 ms/batch**	**Highly depends on workload**

Table 1. Typical request service times by peer activity

Critical service failures under overload

A usual requirement towards performance-critical systems is that they manage overload situations in a predictable and well defined way and with maintaining service at least partially. The simplest of the applicable patterns is actively rejecting new requests that are over the capacity of the system. This is not the case for Hyperledger fabric 0.6. The following can be observed:

- For requests that are served, the write-to-ledger delay runs off.
- A heavily increasing number of requests get silently dropped (after they have been actively accepted!)
- One peer gets “stuck” in a state resynchronization loop (decreasing remaining fault tolerance to zero!), or two peers get “stuck” (disabling consensus and thus, the system).

Chaincode: lack of guaranteed timeliness due to „backend”

Chaincode execution time should be predictable; not only because certain scenarios may be latency-sensitive, but also because chaincode executions are subject to timeoff on each peer. Here, as the „database beckend”, the RocksDB-backed ledger (and blockchain) is the critical component; slow queries may result in failure to execute a transaction on one or more peers. With targeted sensitivity analysis, we have found that a) RocksDB latencies are heavily influenced by configuration, b) latencies have a long tail distribution what – due to the strictly sequential execution – can have a serious global performance impact.

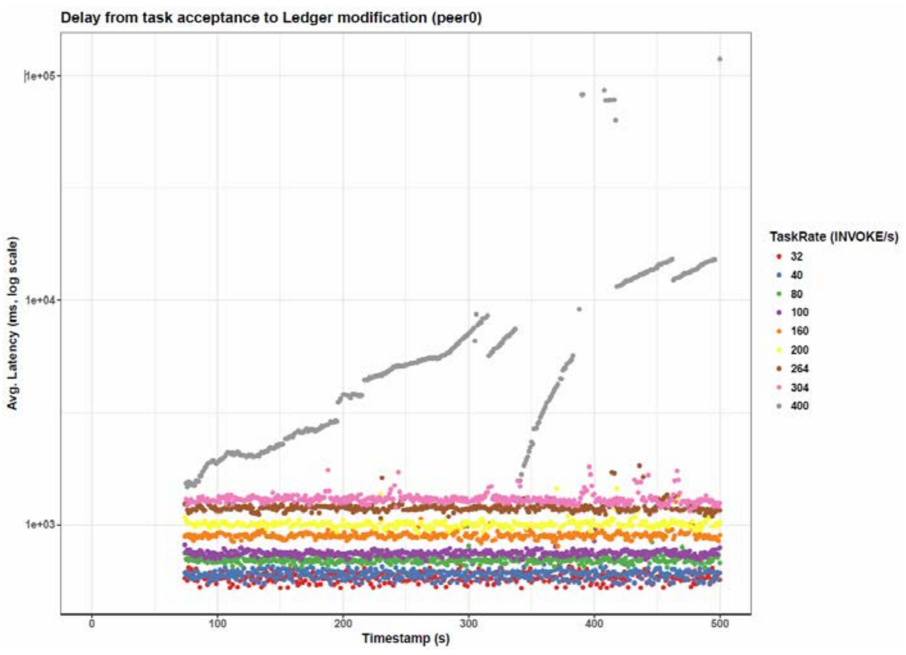

Figure 3. Delay run-off on overload (only not dropped requests)

Summary

We have designed an experimental performance evaluation methodology to support the performance modelling of Blockchain technologies and demonstrated it on Hyperledger fabric 0.6. Our findings show that performing structured experimental performance evaluation is not only key to Blockchain performance-composable deployment design and performance validation, but also provides essential feedback to software design to support design for performance behavior requirements. The methodology can be applied already at early stages of design, when reworking the specification has the least cost and time impact.

Acknowledgements

This work was partially supported by a2016 IBM Faculty Award granted to Dr. András Pataricza.

References

[1] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system, 2008." URL: http://www. bitcoin. org/bitcoin.pdf (2012).

[2] ISO/TC 307: Blockchain and electronic distributed ledger technologies. https://www.iso.org/committee/6266604.html

[3] Home page of the Hyperledger project. https://www.hyperledger.org/

[4] Croman, Kyle, et al. "On scaling decentralized blockchains." International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2016.

[5] Castro, Miguel, and Barbara Liskov. "Practical Byzantine fault tolerance and proactive recovery." ACM Transactions on Computer Systems (TOCS) 20.4 (2002): 398-461.