Predicting Latency of Blockchain-Based Systems Using Architectural Modelling and Simulation

Rajitha Yasaweerasinghelage, Mark Staples, and Ingo Weber
Data61, CSIRO, Level 5, 13 Garden St, Eveleigh NSW 2015 Australia
School of Computer Science and Engineering, University of New South Wales, NSW 2052 Australia
Email: ⟨firstname⟩.⟨lastname⟩@data61.csiro.au

***Abstract*—Blockchain is an emerging technology for sharing transactional data and computation without using a central trusted third party. It is an architectural choice to use a blockchain instead of traditional databases or protocols, and this creates trade-offs between non-functional requirements such as performance, cost, and security. However, little is known about predicting the behaviour of blockchain-based systems. This paper shows the feasibility of using architectural performance modelling and simulation tools to predict the latency of blockchain-based systems. We use established tools and techniques, but explore new blockchain-specific issues such as the configuration of the number of confirmation blocks and inter-block times. We report on a lab-based experimental study using an incident management system, showing predictions of median system level response time with a relative error mostly under 10%. We discuss how the approach can be used to support architectural decision-making, during the design of blockchain-based systems.**

***Index Terms*—Software architecture, Software performance, Distributed databases.**

I. Introduction

Blockchain is an emerging technology providing a shared distributed ledger of transactions, letting participants interact without relying on a central trusted third party. A blockchain can be used as a database, or as a software connector [1], and *smart contracts* [2] on blockchain allow programmable business logic or conditional transactions [3].

Blockchains have limitations, including latency. In a blockchain that use Nakamoto consensus (longest chain wins) [4], it may take seconds (Ethereum) or minutes (Bitcoin) for a transaction to be included in a block. These transactions are never absolutely committed, but increasing confidence of this is provided by subsequent blocks, known as *confirmation blocks*. The latency for initial inclusion of a transaction in a blockchain is higher than in traditional systems, and a large number of confirmation blocks will multiply this delay. Latency can also be impacted by network delays, the transaction fee offered, the number of transactions being processed, and strategic decisions made by miners. So, transaction inclusion times can vary widely.

Although longer than in conventional systems, this may be acceptable for some use cases, if the other potential benefits of blockchain can be achieved, such as decentralised trust. Nonetheless, it will still be important to be able to accurately predict system-level latency during design, to assess whether requirements can be met.

Previously, Xu et al. [1] explored using blockchains as connectors in blockchain-based systems. Weber et al. [3] has showed how they could be used as neutral ground for the model-driven execution of business processes (BPs). This approach allows the integration of organisations without a trusted central coordinating authority. Off-chain 'trigger' components bridge between blockchain smart contracts and enterprise systems. The triggers manage keys, manage enterprise API calls, and can interact with external services and databases. The smart contracts execute the core BP logic, and perform some framework logic, such as for monitoring. Our latency experiments are on the same incident management exemplar for this approach as used by Weber et al. [3].

Architectural models can be used to predict non-functional properties including latency, throughput, resource usage, and cost. Predictions can be made using analytical solvers or simulation engines [5]. In this paper, we use this approach to simulate blockchain-based BPs to predict latency. We use the Palladio workbench [6] as a modelling tool as it is freely available, supports architectural simulation, has a 'UML-like' interface for model construction, and has proven flexibility for extensions such as architectural optimisation [7] and new qualities [8]. The modelling concepts are well-aligned with component-based development and support the re-use of constructed models and components.

A long version of this paper is available as a technical report, which provides many additional details, background, and discussion [9]. The remainder of this paper starts with our approach for modelling and benchmarking transaction latency in blockchain-based systems. The accuracy of system-level predictions from this model is evaluated in Section III. Section IV explores the use of our approach for architectural decision making. We discuss some blockchain-specific modelling issues and future work in Section V, before concluding.

II. Performance Model Construction

We describe benchmarking transaction inclusion times for blockchain, and our approach for system-level modelling.

A. Benchmarking Transaction Inclusion on Blockchain

A key parameter for our model is the *transaction commit time*: the time taken from submitting a transaction until we have sufficient confidence that it has been included in the blockchain. If one block is enough confirmation, we

978-1-5090-5729-0/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSA.2017.22

call this *transaction inclusion time* instead. Our benchmark measurement abstracts over blockchain-related details which affect latency, to create an overall transaction inclusion time distribution for our model. Our benchmark measurements also include latency overhead for our trigger code and the communication between the trigger and the blockchain node.

To demonstrate the approach, we ran benchmarks on a private Ethereum blockchain. This was to prevent flooding the public Ethereum blockchain, to reduce cost, and to be able to vary inter-block time. We used one virtual machine to deploy the trigger and a go-Ethereum (Geth) full node with mining disabled. The mining node was deployed on a different virtual machine. This situation would mimic practical deployment to some degree: each organisation would deploy their own full node and trigger in a virtual machine controlled by them, whereas miner node is operated on separate machines.

To benchmark latency, we submitted transactions as follows. A script invoked the trigger API, which submitted the transaction. The trigger then watched the blockchain for a sufficient number of confirmation blocks after observing a block including the transaction and returned the result to the script. The script then initiated the next transaction.

In our experiments we varied inter-block time, by controlling the complexity mechanism or leaving at its default (uncontrolled). The mean *inter-block time* of the uncontrolled blockchain was 13.6s, in two settings of controlled private blockchain settings, we measured mean inter-block times of 2.3s and 6.3s. For each of the three settings, we measured *transaction inclusion time* across 1000 transactions. The median transaction inclusion times were 6.91s, 14.65s, and 25.8s respectively for mean inter-block times of 2.3s, 6.3s, and 13.6s.

B. Blockchain-Based System Performance Modelling

We model the blockchain from the perspective of the client application, as a component. So, we do not model the blockchain mining network, node communication, or consensus algorithm. These are rolled up in our model and measurements. Client applications interact with the blockchain through a local node. We model the resource and performance characteristics of this node as a component. In the architecture of a scalable client application, one may need to operate multiple blockchain nodes, each independently participating in the blockchain system; in such cases, we would model those as multiple deployed instances of the blockchain client.

1) Component Repository Model: In Weber et al.'s [3] method, off-chain business systems interact with the blockchain through trigger components. We modelled triggers and Ethereum nodes as two components each exposing a relevant interface. In a Palladio Component Model (PCM), component operations are specified in an interface. The trigger interface provides operations for each BP action. The trigger interface also provides a *createInstance* operation, which creates an instance of a BP monitor by invoking a factory smart contract, pre-configured on the blockchain. The trigger translates API calls into corresponding blockchain transactions, whose execution is initiated by the local Ethereum node.

2) Resource Demanding Service Effect Specifications: After modelling the components, interfaces, and their relationships, we then model the non-functional behaviour of component operations in PCM as effect specifications. Each operation translates an API call to a blockchain transaction and uses an external action to forward the transaction to the blockchain node. The resource utilisation of each component is configured as a probability distribution function (PDF) constructed using benchmarks as described in Section II-A. For manual process steps, operator resolution time must be separately benchmarked, but this is not dealt with in this paper.

3) Usage Model: To simulate execution of the system, we model its representative use and points of variation. This usage model reflects process flow in our example BP, and the points of variation are optional branches of the process. For the purpose of our laboratory experiments, we assumed that at each stage of incident response (except for the final developer stage), 75% of issues received were resolved in that stage. The final developer support stage resolves every request.

III. Evaluating System-Level Latency Predictions

We evaluate the prediction accuracy of our performance model using an exemplar BP system, by comparing simulation predictions with macro-level measurements. We used the same private Ethereum environment as for the micro-level benchmarking (Section II-A). However, rather than measuring latency for individual transaction inclusion, we compare latency over entire BP execution instances to our predictions.

A. Incident Management Business Process Implementation

We reuse the incident management system from our previous work. [3] *Trigger* component connect the BP executing on blockchain to enterprise systems. The trigger manages keys, keeps track of the data payload in API calls, and interact with external databases or web services. When a customer submits an issue, the trigger creates a smart contract instance. Other actors can interact with this smart contract instance via the trigger. The trigger updates the status of the process by sending transactions via a blockchain node to that instance, and keeps track of the process. When the transaction is included in a block that is confirmed by a pre-decided number of blocks, the system considers the action to have successfully completed.

B. Experiment Setup

We generated a synthetic workload for the complete scenario. An separately-deployed python script invoked trigger operations using HTTP requests and measured the time delay. The experiment was run 1000 times (created 1000 process monitor instances) and executed for approximately 20 hours. The SimuCom simulation engine was used to execute the PCM model, also for 1000 scenario executions.

C. Comparing Measurement and Simulated Results

The measured and predicted latency distributions are shown in a cumulative density graph in Fig. 1. The cumulative distribution is informative as it shows the percent of process

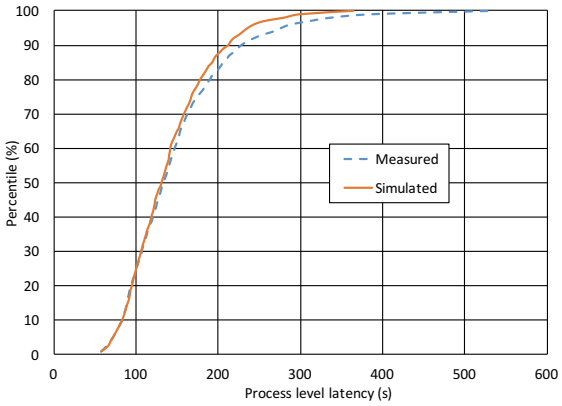


Fig. 1. Scenario latency – Cumulative distribution

executions under specific times. As seen in Fig. 1 the cumulative distribution highly coincides with the results from the benchmark. The simulation predicted the mean latency of the process scenario with a relative error of 1.6%. The measured mean latency was 136.29s and predicted was 134.08s, with a standard error of mean (SEM) of 1.27 and 1.07 respectively.

Further statistical measures are illustrated as boxplot diagram in Fig. 2a. For many applications, 95% and 99% percentiles are useful measures when considering the latency and the skewness of the distribution. The PCM model predicted the 95% and 99% percentiles with a relative error of 9.4% and 11.5% accuracy. Error in predicted maximum and minimum are respectively 7.62% and 16.89%.

IV. Architectural Decision Making

Design alternatives can be evaluated by predicting latency in example scenarios. This lets us explore *what-if* questions in architectural decision making. Xu et al. [1] have discussed blockchain system design alternatives, and the impact of design decisions on quality attributes. Here we focus on latency.

A. Choice of Inter-Block Time

In a public blockchain, the target inter-block time is fixed. However, in private blockchains, it can be varied as a design choice. This reduces transaction inclusion time, which can reduce system-level latency. When evaluating inter-block time alternatives, we use the same models, but modify the transaction inclusion time parameter.

We conducted an experimental evaluation of the accuracy of our simulation for various transaction inclusion times, on a private blockchain. The results are shown as boxplots in Fig. 2.

For the default inter-block time of 13.6s, the measured and simulated median process latency was 132s and 130.93s respectively. For inter-block times of 6.3s and 2.3s, the median measured latencies were 64.7s and 28.1s, and the median simulated latencies were 71.1s and 30.7s respectively. The relative errors of median were 1.4%, 9.6%, and 9.4%, while the relative error of 95^{th} percentiles were 14.6%, 8.5%, and 0.7% respectively.

B. Choice of Number of Confirmation Blocks

The vulnerability of blockchain-based systems to double-spending attacks can be reduced by increasing the number of confirmation blocks [10]. This introduces additional latency to the system. We measured the transaction commit time with 6 and 12 blocks separately and populated the model as mentioned above. We ran a separate experiment for benchmarking the latency of the BP with 1, 6, and 12 confirmation blocks. The results are illustrated as boxplot diagrams in Fig. 3.

We used a controlled blockchain for this experiment with mean inter-block time of approx. 2.3s. The measured median process latencies with 1, 6, and 12 blocks confirmation were 28.1s, 81.5s, and 152s respectively while the simulation predicted 30.7s, 81.7s, and 164s. The relative errors of median predictions were 9.4%, 0.2%, and 0.2% and the relative errors of 95^{th} percentiles were 8.5%, 6.7%, and 12.3% respectively.

C. Process Level Changes

It is straightforward to use architectural performance models to evaluate process-level changes. In our approach, the process is defined by the Palladio usage model. Performance models can also be useful for estimating the impact on latency of process redesign [11] such as task elimination, process integration, or task composition. These are modelled by changing the workflow. Most of the BP control flow patterns [12] can be directly translated to Palladio component model patterns.

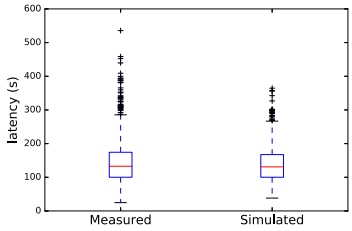
We experimented with a changed process model, where the account manager assigns issues directly to second-level support (skipping the first level) in 5% of cases. We used a private Ethereum blockchain with a mean inter-block time of 2.3s. The results are shown in Fig. 4. The median process latency was measured as 26.8s, vs. simulation as 27.6s. The relative error of median was 2.9% and the relative error of 95^{th} percentile was 0.3%.

V. Discussion and Future Work

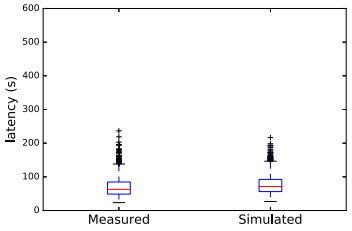
The transaction inclusion-time benchmarks reported in Section II are not intended to be generalisable. Instead, they illustrate our approach to benchmarking to configure a performance model. In particular, our laboratory experiments were performed on a private deployment of Ethereum with only one mining node. This means that there are no significant network delays for transaction or block propagation among peers, and there is no occurrence of *uncles* (short-lived competing alternate histories). Uncles can affect transaction inclusion time. We recommend benchmarking end-to-end latency in the target blockchain platform in order to account for all sources of variation and delay.

The off-chain portion of a system can be modelled conventionally. Our approach reuses standard modelling functionality, so all components can appear in the same model. This can aid visualisation and improve understanding [7].

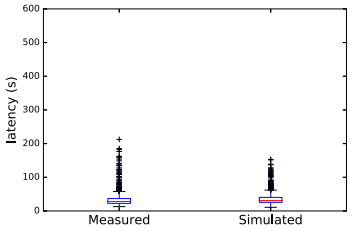
Component and hardware cost can be modelled for blockchain-based systems in a conventional way [7]. However, blockchain-based systems have other costs including *gas* cost for executing smart contracts and transaction fees. It is



(a) For 13.6s average inter-block time: median time (measured 132s, simulated 130s), relative error (median 1.4%, 95th percentile 14.6%)

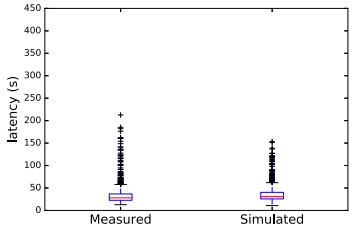


(b) For 6.3s average inter-block time: median time (measured 64.7s, simulated 71.1s), relative error (median 9.6%, 95th percentile 0.7%)

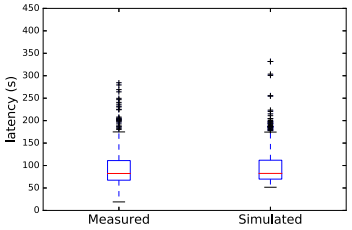


(c) For 2.3s average inter-block time: median time (measured 28.1s, simulated 30.7s), relative error (median 9.4%, 95th percentile 8.5%)

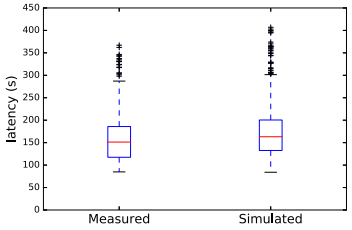
Fig. 2. Boxplot diagrams of measurement and simulated results for transaction inclusion time under various inter-block times.



(a) 1 confirmation block: median time (measured 28.1s, simulated 30.7s), relative error (median 9.4%, 95th percentile 8.5%)



(b) 6 confirmation blocks: median time (measured 81.5s, simulated 81.7s), relative error (median 0.2%, 95th percentile 6.7%)



(c) 12 confirmation blocks: median time (measured 152s, simulated 164s), relative error (median 0.2%, 95th percentile 12.3%)

Fig. 3. Boxplot diagrams of measured and simulated confirmation time for varying numbers of confirmation blocks. Average inter-block time was 2.3s.

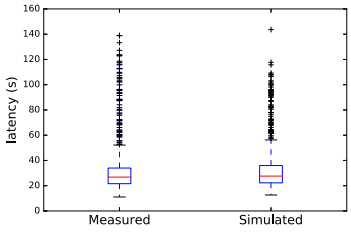


Fig. 4. Measured and simulated latency of modified BP. Median time (meas. 26.8s, sim. 27.6s), relative error (median 2.9%, 95th percentile 0.3%).

possible that these might be modelled as passive resources, We expect simulation-based approaches for cost modelling to help understand the differences in the cost model for blockchains compared to conventional systems.

VI. Conclusion

In this paper, we have proposed and evaluated an approach for predicting the latency of blockchain-based systems using architectural performance modelling and simulation. For an illustrative experimental system in a laboratory environment, our predictions had a relative error of mostly under 10%. We further demonstrated the capability of using these performance models to support evaluation of design alternatives that would be encountered in architectural design. Some of these decisions are about blockchain-specific issues, such as inter-block time or the number of confirmation blocks. Some decisions about a blockchain-based system may be about system-level design options but are impacted by latency arising from the blockchain-related factors. The proposed architectural models also provide a basis for future research into optimal system configuration, cost, and other non-functional properties. Many additional details can be found in our technical report [9].

References

[1] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *Proc. 13th Working IEEE/IFIP Conf. on Software Architecture (WICSA)*, 2016.

[2] S. Omohundro, "Cryptocurrencies, smart contracts, and artificial intelligence," *AI matters*, vol. 1, no. 2, pp. 19–21, 2014.

[3] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, "Untrusted business process monitoring and execution using blockchain," in *Intl. Conf. Business Process Mgmt. (BPM)*, 2016.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[5] A. Brunnert, A. van Hoorn, F. Willnecker, A. Danciu, W. Hasselbring, C. Heger, N. Herbst, P. Jamshidi, R. Jung, J. von Kistowski *et al.*, "Performance-oriented devops: A research agenda," *arXiv preprint arXiv:1508.04752*, 2015.

[6] S. Becker, H. Koziolek, and R. Reussner, "The Palladio component model for model-driven performance prediction," *Journal of Systems and Software*, vol. 82, no. 1, pp. 3–22, 2009.

[7] T. De Gooijer, A. Jansen, H. Koziolek, and A. Koziolek, "An industrial case study of performance and cost design space exploration," in *Proc. SPEC Int'l Conf. on Performance Eng.* ACM, 2012, pp. 205–216.

[8] F. Willnecker, A. Brunnert, and H. Krcmar, "Predicting energy consumption by extending the Palladio component model," in *Symposium on Software Performance*, 2014, p. 177.

[9] R. Yasaweerasinghelage, M. Staples, and I. Weber, "Using architectural modelling and simulation to predict latency of blockchain-based systems," School of Computer Science and Engineering, UNSW Australia, Tech. Rep. 201704, 2017.

[10] M. Rosenfeld, "Analysis of hashrate-based double spending," *arXiv preprint*, 2014. [Online]. Available: http://arxiv.org/abs/1402.2009

[11] H. A. Reijers and S. L. Mansar, "Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics," *Omega*, vol. 33, no. 4, pp. 283–306, 2005.

[12] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.