Proof-of-Execution: Reaching Consensus through Fault-Tolerant Speculation

Suyash Gupta, Jelle Hellings, Sajjad Rahnama, Mohammad Sadoghi

Presenters: Vybhav Achar Bhargav, Kavya Kollipara, Karan Mehta

Some Background: What is PBFT?

- Method to reach consensus on transactions
- Follows a 3 communication phase protocol under the primary-backup model
  - `PRE-PREPARE`
  - `PREPARE`
  - `COMMIT`
- Requires at least `3f+1` to deal with `f` faulty replicas

Why do we need something else?

- `PREPARE` and `COMMIT` phases are all-to-all
- The large number of replicas required to manage faulty replicas
- As such PBFT is considered costly

What is our fix? PoE!

- P.O.E or Proof of Execution
    - Byzantine fault tolerant protocol which uses speculative execution to reach consensus
    - Does not rely on non faulty replicas or trusted hardware
    - Supports batching, out-of-order processing and is scheme agnostic
- Adds four main design elements over PBFT
    - Non Divergent Speculative Execution
    - Safe Rollbacks and Robustness under Failure
    - Agnostic Signatures and Linear Communication
    - Avoid Response Aggregation

What is our fix? PoE!

- Non Divergent Speculative Execution
  - Assuming PBFT terminology, PoE executes after `PREPARE` phase
  - Non Divergent since each replica has partial guarantee that it is prepared
- Safe Rollbacks and Robustness under Failure
  - Malicious Primary can have only certain replicas get a message
  - If the client does not receive a full `proof-of-execution`, PoE permits a rollback
  - This is how PoE guarantees correctness
- Agnostic Signatures and Linear Communication
  - Parties are distrustful and require cryptographic signatures. Can be slow depending on number of replicas
  - PoE supports both Message Authentication Codes or Threshold signatures
- Avoid Response Aggregation
  - Does not require one primary to be the act as a response aggregator.
  - Responses are sent directly to the client

Analysis of Design

We take the best design principles from state of the art data distribution systems.

Impact on PBFT style consensus in terms of design due to speculative execution in PoE:

- Clients need to determine the proof of execution so that the execution of requests can be relied upon.
- The requests are executed first without having been guaranteed, which means during recovery due to failure by any reasons, the replicas need to be able to be rollback failed requests.

Analysis of Design - Execution

- Typical BFT systems order the requests and execute them in that order - reduces support for concurrent execution.
- Execution before ordering - not useful since these systems require to re-verify the execution before commit.

PoE lies between the two, by only partial ordering guarantees, hence reducing communication costs and reducing latencies.

Analysis of Design - Processing

PBFT and other primary-backup protocols use all available bandwidth of the primary for the delegation of requests to the replicas - thus eliminating high message delays

For facilitating this, replicas process the requests within an active window bounded by low and high watermarks. The window size depends on the memory available to the system.

The same is used in PoE as well to be able to handle high throughput.

Analysis of Design - Twin Path Consensus

In difference with Zyzzyva and SBFT:

- Zyzzyva: Replicas directly execute the requests upon primary proposals, and inform the client while the client waits for n responses. If n responses are not received, client sends a message to all the replicas after which an expensive slow path operation takes place to recover (prone to errors).

- SBFT: Fast path of SBFT requires a reliable aggregator and executor to aggregate messages. Even in the fast path, it requires four communication phases while PoE requires only two (or three when using threshold signatures).

Analysis of Design - Primary Rotation

- In HotStuff, the primary is switched after every consensus decision to minimize the influence of a single replica on the consensus.
- It uses an extra communication phase to reduce the cost of the primary replacement. It also uses threshold signatures to reduce the communication costs thereby making it linear.

Unfortunately, the switching of primary is done in a strictly sequential manner, so the possibility of out of order processing is eliminated.

Proof Of Execution

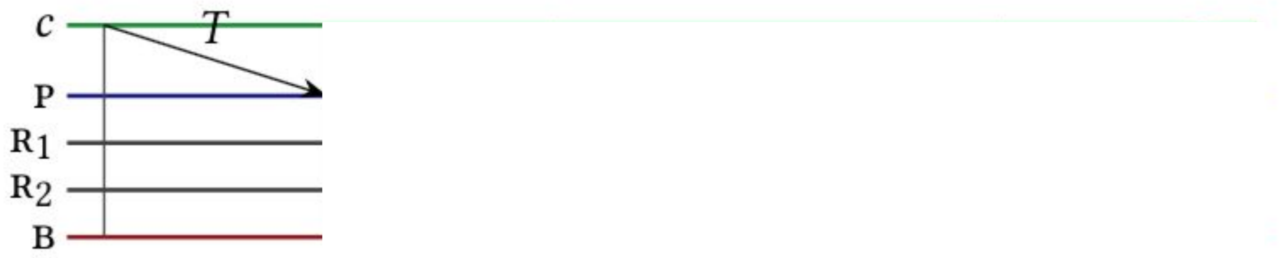
- Primary replica is responsible for proposing transactions requested by clients to all backup replicas.
- Backup replica speculatively execute these transactions with the belief that the primary is behaving correctly.
- Speculative execution expedites processing of transactions in all cases.
- Finally, when malicious behavior is detected, replicas can recover by rolling back transactions, which ensures correctness.

System model and notations

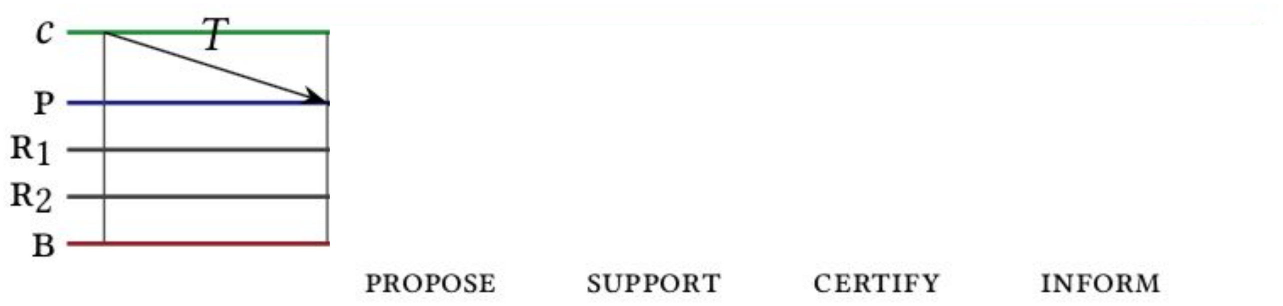
- A system is a set R of replicas that process client requests.
- Each replica $r \in R$ is assigned a unique identifier id(r) with $0 \leq id(r) < |R|$.
- $F \subseteq R$ to denote the set of Byzantine.
- Number of Replicas, $n = |R|$
- Faulty replicas, $f = |F|$
- Non-faulty replicas, $nf = |R \setminus F|$
- We assume that $n > 3f$ ($nf>2f$).

- Authenticated communication is a minimal requirement to deal with Byzantine behavior.
- Depending on the type of message, we use message authentication codes (MACs) or threshold signatures (TSs).
- MACs are based on symmetric cryptography in which every pair of communicating nodes has a secret key.
- TSs are based on asymmetric cryptography. In specific, each replica holds a distinct private key, which it can use to create a signature share.
- Next, one can produce a valid threshold signature given at least nf such signature shares (from distinct replicas). This digital signature can then be verified using a public key.

The Normal-Case Algorithm of PoE



(a) PoE using MACs

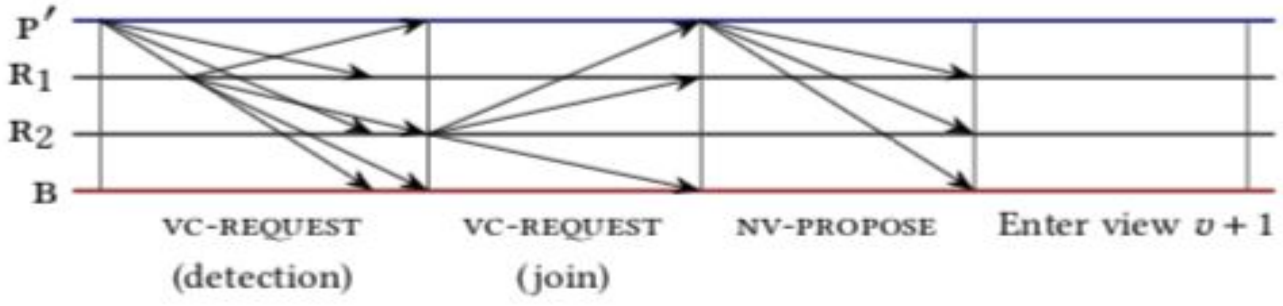


(b) PoE using TSs.

The View-Change Algorithm

The goals of the view-change are

1. To assure that each request that is considered executed by any client is preserved under all circumstances; and
2. To assure that the replicas are able to agree on a new view whenever communication is reliable.



Correctness of PoE

Consider a system in view v, in which the first $k - 1$ transactions have been executed by all non-faulty replicas, in which the primary is non-faulty, and communication is reliable. If the primary received $\langle T \rangle c$, then the primary can use the normal case algorithm of PoE to ensure that

1. there is non-divergent execution of T ;
2. c considers T executed as the k -th transaction; and
3. c learns the result of executing T (if any),

this independent of any malicious behavior by faulty replicas.

Fine-Tuning and Optimizations

1. To reach nf signature shares,the primary can generate one itself. Hence, it only needs nf − 1 shares of other replicas.
2. The propose, support, inform, and nv-propose messages are not forwarded and only need MACs to provide message authentication. The certify messages need not be signed, as tampering them would invalidate the threshold signature. The vc-request messages need to be signed, as they need to be forwarded without tampering.

Designing PoE using MACs

The usage of only `MACs` makes it impossible to obtain threshold signatures or reliably forward messages.

Using MACs requires changes to how client requests are included in proposals, to the normal- case algorithm of PoE, and to the view-change algorithm of PoE.

The changes to the proposal of client requests and to the view-change algorithm can be derived from the strategies used by `PBFT` to support `MACs`. Hence, next we only review the changes to the normal-case algorithm of PoE.

ResilientDB Fabric - Batching

ResilientDB is used for comparison of performance between different consensus protocols and to reach high throughput.

Instead of PROPOSE message sending one request from the primary, we can aggregate multiple client requests into one message and reach a single consensus for all these requests.

- At the primary replica, batch-threads are spawned which work on aggregate the client requests into a batch.Batch-threads also hash the requests in a batch into a unique digest.
- Batch-threads also hash the requests in a batch into a unique digest.
- The input threads at primary receives these requests, assigns them with a sequence number and enqueues them into a batch-queue (common lock-free).

ResilientDB Fabric - Ledger Management

A blockchain ledger is maintained across different replicas. This ledger $B_i := \{k, d, v, H(B_{i-1})\}$ comprises of the sequence number, digest of the request, view number and the hash of the previous block.

- In ResilientDB, prior to any consensus, the primary first creates a genesis-block which contains some basic data and acts the starting block of the blockchain. It uses the hash identity of the initial primary as this information is available to all the replicas.
- Post the genesis block is created, each replica can independently add next block to this chain. A block is only created once it completes execution of a batch of transactions.
- ResilientDB stores the proof of execution of the k-th request as a threshold signature in the k-th block to prove the validity of execution.

Evaluation of PoE

- Tested on `c2` compute clusters on GCP
    - 320k clients across 16 machines
- Benchmarks were performed with the help of the Blockbench's macro benchmarks using the Yahoo Cloud Serving Benchmark (YCSB) dataset
- Benchmarked against PBFT, ZYZZYVA, SBFT and HOTSTUFF

Evaluation of PoE

- Two main two-dimensional conditions to look at
  - Zero Payload vs Standard Payload
    - Zero Payload: all replicas execute 100 dummy instructions. No Batching
    - Standard Payload: Sending messages in a batch size of 100. `PROPOSE` message size is `5400B`, `RESPONSE` message is `1748B`. All other messages are `250B`.
  - Failures vs Non-Failures

Evaluation of PoE - Scaling Replicas

- Standard Payload
  - Single Backup Failure
    - PoE has up to 43%, 72%, 24x, 62x more throughput than PBFT, SBFT, HOTSTUFF, ZYZZYVA respectively.
  - No Replica Failure
    - PoE has 20% to 13% less throughput than ZYZZYVA and 35%, 27%, 21x more throughput than PBFT, SBFT, HOTSTUFF respectively.
- Zero Payload
  - PoE attains up to 85%, 62% and 27× more throughputs than Pbft, SBFT and HotStuff, respectively.
  - In failure-free conditions, the throughput attained by PoE is comparable to ZYZZYVA

Related Work

- Consensus for Blockchains
  - Bitcoin employs Proof of Work protocol which is computationally intensive
  - Low throughput
  - Can cause forks
  - Use Proof Of Stake
    - Replicas own n% of the new blocks
    - Resource Driven
    - Can handle attacks where replicas together work against several forks
- Meta protocols that may run PBFT in parallel, aim to remove single primary dependency

Conclusion - Proof Of Execution

- Byzantine fault-tolerant consensus protocol
- Guarantees safety and liveness, in three linear phases
- Out of order execution through speculative execution
- May have to revert based on replicas
- Achieves up to 80% higher throughputs than existing BFT protocols