Hyperledger Burrow (formerly eris-db)

2017-03-28

HIP identifier

Burrow, the permissionable smart contract machine.

Sponsors

Casey Kuhlman
Monax
casey@monax.io

Benjamin Bollen
Monax
ben@monax.io

Silas Davis
Monax
silas@monax.io

Dan Middleton
Intel
dan.middleton@intel.com

Abstract

Burrow (formerly known as eris-db) is a permissionable smart contract machine. The first of its kind when released in December, 2014, Burrow provides a modular blockchain client with a permissioned smart contract interpreter built in part to the specification of the Ethereum Virtual Machine (EVM).

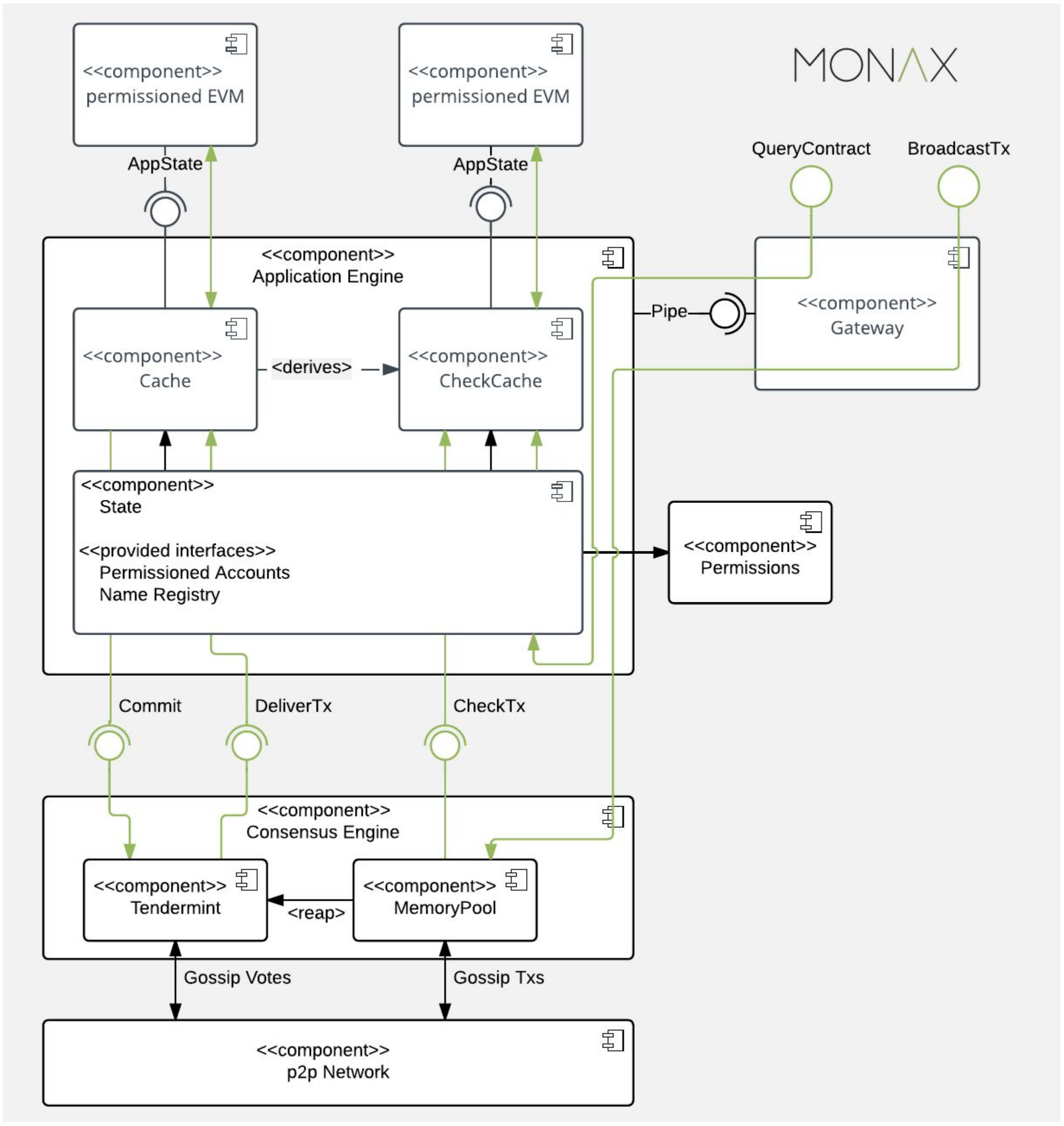
Initially licensed under GPLv3 and as of version 0.16 relicensed to Apache 2.0, Monax's permissioned EVM is functionally separate from the Ethereum protocol or any of the codebases implementing it. Burrow's users can operate any smart contract that has been compiled by any EVM language compiler in their own permissioned blockchain environments.

Burrow extends previous work within the Hyperledger Project by providing a strongly deterministic smart contract focused blockchain design to the Project's overall effort. Burrow's architecture opens a range of interesting cross-project collaborations within the Hyperledger umbrella.

Given that Monax's permissioned EVM is built to the specification of Ethereum's active blockchain development community, and that Burrow has active enterprise users, some of whom are existing members of the Hyperledger Project, we feel that Burrow has significant potential to play an important role within the quickly-growing Hyperledger community.

Context

Burrow's high-level architecture is described (in diagrammatic form) below.



More details on the specific architecture and features of Burrow are included below in the "*Solution"* section of this HIP.

Motivation

Burrow's primary users are businesses aiming at value chain level optimization amongst other blockchain and smart contract benefits. These users *require* permissions on their blockchain deployments in order to fulfill numerous legal and/or commercial requirements for their applications.

Burrow was designed to be a general-purpose smart contract machine and is not optimized for the requirements of any single industry; rather Burrow has been optimized for general-purpose, cross-industry smart contract use cases.

Startups to enterprises are using Burrow's permissioned EVM so that they may leverage smart contract innovations from the open source world in a more secure, legally compliant, cryptocurrency-free, enterprise-grade setting.

For a bit of overall historical context on the background, goals, and learnings of where Burrow has grown out of please see this Github file. At that time we said the following:

> *At Eris* [now Monax] *our approach to blockchain technologies is that blockchains, but really more interestingly and importantly, smart contract networks (which just so happen to currently reside on blockchains), are an immensely helpful tool.*

This is "*the road we have travelled*". So far. As to why we are submitting Burrow for incubation by the Hyperledger Project, we have covered our motivations in this blog post.

Status

Burrow's current release is version 0.16.1. It has existed under various names since October, 2014.

Burrow is under active development. The latest source code may be found on Github.

Burrow is licensed under the Apache License, version 2.

Burrow is deployable over any cloud platform, currently available on AWS and Azure with further ready-to-deploy cloud solutions already under development.

Several tasks for the project are either in progress or on the design table. These can be broken down into "usability improvements" and "code improvements".

N.B., the following tasks do not include the harmonization efforts described below in the "*Solution"* section of this HIP.

Usability Improvements

- Improved documentation
- Testing improvements (particularly with respect to long running tests, performance or load testing, fault tolerance testing at node level)
- QA and acceptance improvements

Code Improvements (these are largely described in this file)

1. Harmonization of the RPC against Ethereum's web3 standard
2. Enhancements to the internal events system
3. Privacy enhancements
4. Scalability enhancements
5. Formalizing Secure Natives framework and packages (multisig, various cryptographic primitives, etc.)

Solution

Users of Burrow are able to benefit from having an access control layer through the use of smart contracts and our "secure natives" based permission layer. Burrow's generalized architecture should be largely approachable to the members of the Hyperledger Technical Steering Committee.

The major components of Burrow are as follows:

- **Consensus engine** which is responsible for maintaining the networking stack between nodes and ordering transactions to be utilized by the application engine.
- **Application Blockchain Interface** ("ABCI") provides the interface specification for the consensus engine and application engine to connect.
- **Smart contract application engine** provides application builders with a strongly deterministic smart contract engine for operating complex industrial processes.
- **Gateway** provides programmatic interfaces for systems integrations and user interfaces.

Each of these components is addressed in greater detail below.

Consensus Engine

Burrow currently uses the Tendermint consensus engine which implements documented consensus and p2p protocols. Transactions are ordered and finalised by its Byzantine fault-tolerant, deposit-based proof of stake engine.

The Tendermint consensus engine provides high transaction throughput over a set of known validators and prevents a blockchain from forking. This instant-confirmation finality is vital for our users who require non-reconciling integrations with their other systems.

The Tendermint consensus engine is a separate project which is utilized as a dependency of Burrow. It is not included in this submission. Tendermint is also currently licensed under the Apache License, version 2.

Burrow is a *user* of consensus engines rather than focused on *making* a consensus engine. Due to Burrow's use of an application engine-consensus engine interface (ABCI) it has an ability to leverage other consensus engines (the concept known as "pluggable consensus" within the blockchain community). We are particularly interested in collaborating with the PoET team to work towards integration of that consensus engine.

Application Blockchain Interface (ABCI)

The consensus engine interfaces with the smart contract application engine over the ABCI. The interface abstraction allows for the consensus engine to remain (largely) agnostic from the smart contract application.

We envision working closely with other current and future Hyperledger projects to enable increased utility and separation of concerns (modularity) for these very different feature suites provided by current blockchain technology, which is largely monolithic.

Smart Contract Application Engine

At a fundamental level, the application engine validates transactions and applies them to the application state in the order that the consensus engine provides them to the application engine over the ABCI. The majority of the value Burrow directly provides to users is included in the Smart Contract Application Engine. Some of its key sub-components are outlined below:

Application Global State

The application state consists of all accounts, the validator set, and Burrow's in-built name registry. A transaction that calls on the smart contract code in a given account will activate the execution of that account's code in a permissioned virtual machine.

Secure Native Functions

Secure native functions provide the ground rules that all accounts and all smart contract code must follow. They do not reside as EVM code, but are exposed into the permissioned EVM via interface contracts. Permissioning is enforced through secure native functions and underlies all smart contract code execution (the permission layer is covered below).

Burrow envisions a secure native functions framework which supports the use of native language code for higher performance and security. Secure native functions can be exposed into the permissioned EVM within Burrow.

Secure native functions can also be structured to provide the basis for increased smart contract performance. Secure native functions can provide a range of “sudo” level functions to ecosystem applications which can and should be built natively and exposed into the permissioned EVM. Work remains as to how to systematize this and also to add additional advanced capabilities necessary to support a wide variety of users.

Permission Layer

Burrow comes with a capabilities-based, evolvable permissioning layer. This permissioning layer was, when released in December 2014, the first of its kind available on the market.

The network is booted with an initial set of accounts which have permissions as well as a global default set of permissions. Network participants with the correct permission can modify the permissions of other accounts by sending an appropriate transaction type to the network which is then vetted by the network validators before the new permissions are updated on the target account. Via the EVM, additional sophisticated roles-based permissioning can be leveraged via the use of Burrow’s roles feature which exists on each account. Roles can be updated via discrete transactions or by smart contracts.

In addition, Burrow exposes the ability for smart contracts within the permissioned EVM to modify the permission layer and account roles. Once a contract leveraging this functionality has been deployed to a chain, a network participant with the appropriate permissions can grant the contract the capability.

In the future, we feel there is a possibility that this suite of Burrow’s features would harmonize to a great degree with the work of the Fabric and Sawtooth Lake teams.

Permissioned EVM

This virtual machine is built to observe the Ethereum operation code specification and additionally asserts the correct permissions have been granted. An arbitrary but finite amount of gas is handed out for every execution to ensure a finite execution duration (halting problem) - the idea is that *“you don’t need money to play, when you have permission to play”*.

Transactions need to be formulated in a binary format that can be processed by the blockchain node using an Application Binary Interface (ABI). Currently a range of open source tooling made both by Monax and the general Ethereum community provides users with the functionality to compile, deploy and link smart contracts compiled for the permissioned EVM and to formulate transactions that call smart contracts. Future work on the light client will be aware of the ABI to natively translate calls on the API into signed transactions that can be broadcast on the network.

The permissioned EVM itself is designed and implemented as a stateless function to deterministically and verifiably transition an application state given a transaction. This code package has the potential to be integrated in different Hyperledger projects. For example, the extensible transaction families in Sawtooth allow us to think of the permissioned EVM as a *Transaction Processor* in Sawtooth's permissioned ledger framework.

Gateway

Burrow exposes RESTful and JSON-RPC endpoints for clients to interact with the blockchain network and the application state through broadcasting transactions, or by querying the current state of the application.

Websockets allow interfacing components to subscribe to events, which is particularly valuable as the consensus engine and smart contract application engine can give unambiguously finalised results to transactions after each block.

Signing

Burrow accepts client-side formulated and signed transactions for which we have an interface for remote signing available. Further work is in progress to integrate identity management to allow existing RSA X.509 certification solutions to assert valid elliptic curve public keys on the blockchain. We look forward to input from and alignment with the Hyperledger Identity workgroup on this matter.

External signing solutions are crucial for Burrow's users as they allow the blockchain nodes to be ran on commodity hardware. We are very interested in harmonizing our ongoing work to implement remote signing with other Hyperledger Projects that also are seeking to leverage external signing solutions rather than requiring users to run their blockchains on highly secure hardware.

Interfaces

Burrow also uses boot and runtime interfaces, largely via files read by the blockchain node at boot; of course it also includes an RPC which allows for interfacing with the node during runtime. We would be very interested in working across the other Hyperledger projects to come to agreement on the following interfaces:

- genesis.json (or equivalent network state at boot file) specification
- config.toml (or equivalent node runtime file) specification

- high-level RPC | API | Gateway interface specification (while various blockchain clients will differ in their RPC, much information can and should be standardized)

It is our view that as much standardization as possible for those interfaces across the various Hyperledger blockchain engines would greatly benefit developer tooling, developer operations systems, and blockchain explorers.

Effort and resources

Currently two persons (Silas Davis and Benjamin Bollen, Monax) are committed full-time to developing and maintaining the Burrow project. To realize its full potential, we would welcome and encourage other developers and test engineers to join Silas and Ben in this effort.

How to

The project is hosted at Github. The project and documentation contain all relevant build, test, and operation specifics.

The CI is tracked by circle-ci. We will follow the Hyperledger community's guide for the CI once the proposal gets accepted.

Documentation is included in the repository.

Closure

The project will succeed if organisations use Burrow's application engine as the basis of their ecosystem applications.

Acknowledgement

We are building a community of co-maintainers (as defined by the relevant Hyperledger documents) around Burrow and will remain as lead contributor with our expertise on the platform. Thus far we have interest in co-maintenance from:

No.	Name	Organization	Email
1	Dr. Andreas Freund	Tata Consultancy Services	andreas.freund@tcs.com