QDRVM

Polkadot Treasury Proposal
KAGOME – C++ implementation of Polkadot Host
Milestone 3

Proponent:
1544YD9AzZNXq3Bickbk4rGRQ5piRP5AP9b38Nw6boCx58q3 (Quadrivium)
Period: 01.01.24 — 01.11.24 (10 months)
Date of submission : 28.06.24
Requested allocation: 840,800 USD | 141 395 DOT

Second milestone scope:

- Grid and cluster topologies
- Security features
- Elastic scaling
- Validation protocol upgrade
- Systematic chunks
- Disabled validators
- Security audit

QDRVM

1. Context of the proposal:

KAGOME is a C++ implementation of the Polkadot Host that is protocol compatible with the original implementation in Rust and is already participating in Kusama relay chain validation. With KAGOME and other client implementations we bring client diversity to Polkadot, mitigating risks of fatal bugs, bringing innovations and broadening the development community. During previous work as part of Polkadot treasury proposal 2, the following results were achieved:

- Asynchronous backing
- BEEFY integration
- New Wasm engine (WasmEdge)
- Initial security audit by SRLabs (link)
- General maintenance

In addition to features that were planned during the previous proposal our team achieved the following:

- Implemented Grid and Cluster network topologies for statements distribution
- Partially Integrated disabled validators feature
- Partially implemented elastic scaling
- Implemented multiple security features

This proposal asks for funding for the next 4 months, in addition to the previous 6 months of work on features that have been retroactively implemented (marked with Done status in Section 4).

2. Problem statement

Client diversity is essential for the security and resilience of Polkadot as it helps to mitigate the risk of bugs and exploits. If there is only one implementation (which is currently the case Polkadot), then any bug or exploit that is found in that implementation could potentially bring down the entire network. However, with multiple implementations, the risk of a bug or exploit affecting all of the nodes in the network is greatly reduced.

Another advantage of having multiple clients is that it is opening the way for having another team that already sees the end state of the protocol and therefore has the advantage to implement protocol in the most optimal way.

Importance of having multiple client implementations was highlighted multiple times by Polkadot twitter (source), and Web3 foundation RFP (source). Moreover, W3F recognizes clients diversity importance and allocates 10 million DOT prise for the future multiple implementations of JAM protocol.

Gavin Wood mentioned KAGOME during 2023 yearly roundup article (link) and highlighted our achievement of executing KAGOME node in Kusama during his recent interview (link).

To learn more about multi-client philosophy and KAGOME Polkadot Host implementation watch:

- Building alternative clients in Polkadot | Polkadot Decoded 2023
- Polkadot Host architecture in 2024 | Sub0 Asia 2024

3. Alignment with JAM

Quadrivium is planning to develop a JAM client and is already working on some features for this in parallel with KAGOME. We also started work on SASSAFRAS many months ago, but stopped because the reference implementation in Rust had not yet been merged into the Polkadot SDK. Given that SAFROLE is the simplified version of SASSAFRAS, we are well positioned to implement a new leader election mechanism in the JAM implementation.

However, we recognize the importance of multiple Polkadot Host implementations today, especially since most features are already available and KAGOME is compatible with the Polkadot SDK. Moreover, it's not anticipated that JAM will be launched within the next one or two years. Luckily, we can repurpose much of the KAGOME codebase for the future JAM client implementation. For instance:

1. Grandpa – fully implemented and audited.
2. SASSAFRAS – partially implemented.
3. SCALE – fully implemented.

But since we didn't initially intend to reuse these components, some refactoring of KAGOME is required. This will enable these components to be reusable and easily integrated into the future JAM client implementation.

4. Proposed feature set

I. Security improvements (retroactive)

During the initial security audit of KAGOME, multiple security features were requested to be implemented.

Secure validator mode

Secure-Validator Mode offers many measures to improve key security, such as strict filesystem, networking, and process sandboxing, on top of the existing wasm sandbox.

During the initial security audit by SRLabs, they observed that KAGOME lacks a Secure-validator mode. Consequently, Kagome validators may be more vulnerable to a variety of security threats. These include direct attacks on the validator nodes, exploitation of vulnerabilities, and an elevated risk of participating in network consensus under adversarial conditions. As a result, adding support for Secure-Validator mode is considered crucial.

Status: ✅Done

Links:

- https://wiki.polkadot.network/docs/maintain-guides-secure-validator#secure-validator-mode
- https://github.com/qdrvm/kagome/pull/2042

WASM Stack depth instrumentation

The Polkadot-sdk limits stack depth to ensure consistent PVF performance, regardless of wasmtime version or architecture, unlike Kagome. This could cause

disputes between Kagome and polkadot-sdk validators. It was possible for a legitimate parachain with an unbounded recursion vulnerability to be exploited. A malicious collator could create a block exceeding polkadot-sdk's stack limit, but valid in Kagome. If backed, this block would be disputed by all polkadot-sdk validators.

Status: ✅Done

Links:

- https://github.com/paritytech/polkadot-sdk/blob/e38998801e433ecc569ff6d58d1d0aa80eaff771/substrate/client/executor/common/src/runtime_blob/runtime_blob.rs#L83-L90
- https://github.com/qdrvm/kagome/pull/1946

Safe memory storage

The security assessment by SRLabs identified an issue where, after using the private key for signing operations, the memory allocated for storing the key was not adequately zeroed or cleared. This behavior was leaving residual data in memory, which could potentially be accessed by unauthorized processes or users.

The primary risk associated with this issue was potential exposure of sensitive private key information. Although classified as low risk due to the limited scenarios in which this vulnerability can be exploited, the persistence of private key data in memory after use can be a vector for targeted attacks aimed at extracting cryptographic keys.

To solve the issue Quadrivium team implemented the storage for private keys and seeds in secure OpenSSL heap which won't be swapped on disk in a case of core dumps and is protected from reads due to out-of-bound undefined behaviour.

Status: ✅Done

- https://github.com/qdrvm/kagome/pull/1997

II. Grid and cluster topologies (retroactive)

Grid topology is a communication protocol used in Polkadot Host to distribute backing statements among validators. It is crucial for the block producing node to receive information about the backed parachain candidate as quickly as possible. The importance of this has increased, particularly after the asynchronous backing update, which led to a rise in the number of messages.

In Grid topology, each validator uses a deterministic method to create the same view of every other validator by forming a Grid. Validators can only send messages to their column and row neighbors, excluding validators from their own backing group. This method ensures that any message between two validators can be propagated within a maximum of two hops, reducing the number of messages in the network and improving efficiency.

For validators in the same group, a cluster topology is implemented, allowing validators within groups to communicate directly.

Status: Done

Links:

- https://github.com/qdrvm/kagome/pull/2095
- https://github.com/qdrvm/kagome/pull/1891
- https://github.com/qdrvm/kagome/pull/2096
- https://github.com/qdrvm/kagome/pull/2102/
- https://youtu.be/Lv2KQ2EDyM8?si=8Br6ilVIdry8xyH2&t=1084

III. Elastic scaling

With the implementation of asynchronous backing, Parachains gained the ability to produce new blocks without waiting for the inclusion of the previous ones. This reduced the block time from 12 seconds to 6 seconds.

Elastic scaling further improves Parachains' performance by enabling them to occupy multiple cores and execute multiple blocks in parallel.

The KAGOME team has already implemented a significant portion of elastic scaling, such as replacing validator assignment from parachain IDs to core IDs. This allows handling multiple candidates for the same parachain during the same slot

(https://github.com/qdrvm/kagome/pull/1996). However, there are still a few features remaining to fully align with the Polkadot-SDK implementation, such as allowing out-of-order execution of parachain blocks and switching from Fragment trees to Fragment chains for optimization.

Status: 🤷Partially implemented

Links:

- https://github.com/paritytech/polkadot-sdk/issues/1829
- https://github.com/paritytech/polkadot-sdk/pull/4035
- https://github.com/qdrvm/kagome/pull/2102
- https://forum.polkadot.network/t/polkadot-summit-24-pov-reclaim-elastic-scaling/7232
- https://polkadot.network/blog/elastic-scaling-streamling-growth-on-polkadot

IV. Validation protocol upgrade (validation v3 + assignments v2)

Assignments v2 protocol introduces a new type of assignment certificate for tranche0 assignments. Now, instead of issuing or importing one tranche0 assignment per candidate, there will be a single certificate per relay chain block for each validator. However, these new assignment certificates won't be distributed immediately, so operations should continue as before. To enable the new protocol, the majority of validators need to run this version. Hence, it's essential for KAGOME to support the new protocol version.

Validation v3 protocol is the optimization allowing more validators participating in parachains consensus. While this implementation is backwards compatible with validation v2 already supported by KAGOME, it is required for KAGOME to implement the most recent version of protocol to allow other nodes take advantage of the optimization.

New version of protocol batches multiple approval messages into a single message allowing batch verification by validator

Status: Not started

Links:

- https://github.com/qdrvm/kagome/issues/1923
- https://github.com/paritytech/polkadot-sdk/pull/1178

V. Systematic chunks (RFC-47)

Polkadot employs erasure coding to encode and distribute data availability chunks among validators. Despite this, the systematic nature of the erasure coding algorithm used by Polkadot allows room for optimizing the decoding process, which is crucial for approval checking and dispute resolution.

An erasure coding algorithm is deemed systematic if it includes the original unencoded data within the resulting code. The erasure coding algorithm used for Polkadot's data availability is systematic. Essentially, the first N_VALIDATORS/3 chunks of data can be effortlessly concatenated to retrieve the original data, bypassing the need for a resource-intensive and time-consuming reconstruction algorithm.

The new *req_chunk/2* protocol enhances data retrieval by using systematic chunks for data recovery. For compatibility with Polkadot-SDK, it's essential for KAGOME to integrate this protocol.

Status: Not started

Links:

- https://polkadot-fellows.github.io/RFCs/approved/0047-assignment-of-availability-chunks.html

VI. Disabled validators mechanism

Disabling validators means temporarily ignoring votes from misbehaving validators.

Disabled validators mechanism is integrated into different parts of Polkadot Host such as Grandpa, BABE, Backing, Disputes resolution. Integration is done via new runtime entries that return disabled validators.

Also the validator should initiate extrinsics if they notice any misbehavior that should be escalated to other validators.

Status: 🤷Partially implemented

Links:

- https://github.com/paritytech/polkadot-sdk/issues/784
- https://github.com/qdrvm/kagome/issues/2068
- https://github.com/qdrvm/kagome/issues/2060
- https://github.com/qdrvm/kagome/issues/2005

VII. Minor features and improvements

RFC-0043: Introduce `storage_proof_size` Host Function for Improved Parachain Block Utilization

This RFC suggests the introduction of a new host function for parachains, `storage_proof_size`. This function will indicate the size of the current storage proof to the runtime. This will allow runtime authors to enhance block utilization by retroactively recovering unused storage weight.

Status: Not started

Links:

- https://polkadot-fellows.github.io/RFCs/approved/0043-storage-proof-size-hostfunction.html

RFC-0091: DHT Authority discovery record creation time

Extend the DHT authority discovery records with a signed creation time, so that nodes can determine which record is newer and always decide to prefer the newer records to the old ones.

Status: Not started

Links:

- https://polkadot-fellows.github.io/RFCs/proposed/0091-dht-record-creation-time.html#rfc-0091-dht-authority-discovery-record-creation-time

RFC-0013: Prepare `Core` runtime API for MBMs

Introduces breaking changes to the `Core` runtime API by letting `Core::initialize_block` return an enum. The versions of `Core` is bumped from 4 to 5.

Status: Not started

Links:

- https://polkadot-fellows.github.io/RFCs/approved/0013-prepare-blockbuilder-and-core-runtime-apis-for-mbms.html

Qtils

To resolve technical debt, the KAGOME team decided to consolidate common tools used in both libp2p and KAGOME into a separate library called Qtils. This makes it easier to maintain all the necessary tooling for both projects in one location.

Status: Done

Links:

- https://github.com/qdrvm/qtils

Optimize SCALE (retroactive)

KAGOME improved C++ SCALE implementation in KAGOME project by getting rid of redundant memory allocations that led to slower encoding and decoding

Status: Done

Links:

- https://github.com/qdrvm/kagome/pull/1782

VIII. DevOps and QA maintenance

In order to constantly ensure the quality of KAGOME as well as keep it compatible with Substrate (and potentially other Host implementations such as Gossamer) Quadrivium maintains, monitors and improves multiple environments for KAGOME.

It is important to notice any incompatibilities if any as soon as possible. Therefore we maintain multiple syncing and validating nodes:

- Kusama syncing node
- Westend validating node
- Westend Polkadot-SDK node (for debugging Polkadot-SDK behaviour against KAGOME nodes)
- Paseo syncing node

Moreover, we are maintaining the list of zombienet tests and constantly improving our CI to conveniently execute them. During the past 6 months we introduced:

- Migrated our entire infrastructure to Google Cloud Platform (GCP), leveraging its robust and scalable environment.
- Actively utilized Kubernetes for managing infrastructural services, deployments, instance maintenance, and for syncing and validating nodes.
- Adapted Helm charts for internal use to fully exploit the capabilities of GCP and meet our specific requirements.
- Implemented automation tools to swiftly diagnose the causes of system crashes and analyze core dumps.
- Centralized our log collection and analysis system, significantly improving our ability to monitor and troubleshoot issues.
- Integrated with GitHub Actions, allowing us to optimize builds by utilizing the diverse range of GCP instances available.
- Enhanced our monitoring systems and improved the visualization of metrics, providing clearer insights into system performance and health.
- Enabled CI Zombienet tests execution for any branch within the project, streamlining our development and testing processes.

In addition, our QA team is constantly ensuring quality of the most recent KAGOME features by running them against different versions of Polkadot-SDK to quickly

notice incompatibilities and create bug reports. In addition we conduct regression test scenarios before every release.

IX. Security assurance by SRLabs

KAGOME's mission to provide a robust and secure alternative client implementation for node operators in the Polkadot ecosystem remains a top priority. Following the initial security audit conducted by SRLabs on KAGOME v0.9.3, it is imperative to establish a structured and ongoing assurance process to continually safeguard the integrity and resilience of KAGOME's implementation. This section outlines the scope and activities for the second phase of the security assurance, emphasizing proactive measures and collaborative efforts to enhance security over time.

Objectives of the Security Assurance Scope:

- **Assure changes to KAGOME before they go live:**
  Ensure that all modifications to KAGOME are thoroughly reviewed and secured prior to deployment, minimizing the risk of vulnerabilities being introduced in the production environment

- **Ensure code quality and security assurance**
  Implement code review and security assurance processes to maintain high code quality standards, detect potential vulnerabilities early, and ensure robustness of KAGOME's implementation

Support scope for this milestone:

The scope of this security assurance milestone resolves around the following key activities, but is not limited to:

1. **Conduct scheduled security audits and follow milestone PRs of KAGOME:**
   a) Perform security audits on the new features developed since the initial audit by SRLabs, including:
   - Security improvements (retroactive)
   - Asynchronous backing
   - Elastic scaling
   - Validation v3
   - Disabling validators
   - Grid and cluster topologies

- WasmEdge integration
- Systematic chunks
- Reviewing minor enhancements

The full diff for review is available by the link:
https://github.com/qdrvm/kagome/compare/v0.9.3...master

b) In collaboration with the KAGOME development team, monitor pull requests (PRs) submitted to the KAGOME repository. The KAGOME development team will guide which PRs require detailed security audits, ensuring comprehensive review and validation before integration into the main codebase

2. **Additional security assurance measures upon request**:
   Provide expert consultation and brainstorming sessions to explore and recommend additional security measures. These sessions will be conducted upon request and will focus on approaches to enhance the overall security framework of KAGOME
   - Conduct brainstorming sessions to identify potential security vulnerabilities in upcoming features or changes
   - Engage in discussions on best practices and security measures that can be integrated into KAGOME
   - Provide recommendations for
     - Security testing methodologies, such as fuzzing
     - Protocol and architecture enhancements
     - Risk mitigation strategies

Joint alignment sessions will be conducted between Quadrivium and SRLabs to prioritize the support scope during this milestone. This session will ensure that the most critical security needs are addressed promptly and effectively.

The security assurance for KAGOME by SRLabs is designed to be adaptive and responsive, addressing the evolving needs of the project. By ensuring thorough audits of code changes, conducting in-depth reviews and exploring additional security measures, we aim to maintain and enhance the security of KAGOME. This approach will help KAGOME its goal of providing a secure and reliable client implementation for the Polkadot ecosystem.

5. Projected task allocation and payment details

Quadrivium development team

1. Engineering manager
2. Senior C++ developer x 4.5
3. DevOps engineer x 0.5
4. QA engineer x 0.5

Epic	Feature	Description	ETA (hours)
Security improvements	**Secure validator mode**	**Enhances key security with measures like strict filesystem, networking, and process sandboxing.**	**150**
	WASM Stack depth instrumentation	**Limits stack depth to ensure consistent PVF performance. Prevents disputes between different validators.**	**150**
	Safe memory for keystore	**Secure storage for private keys in the OpenSSL heap, protecting against data exposure during core dumps and out-of-bound read operations.**	**50**
Grid and cluster topology		**A mechanism for efficient communication between validators using a grid formation and cluster topology**	**300**
Elastic scaling	**Fragment trees updates**	**Allows for Parachains performance improvement through parallel execution of multiple blocks.**	**300**

	Allow backing of multiple candidates	Use core index instead of parachain index with corresponding updates in statement messages	300
	Unit tests	Ensure conformance with Polkadot-SDK by implementing the same unit tests	100
	Zombienet test	Implement and integrate into CI zombienet test executing elastic scaling in network with Polkadot-SDK and KAGOME validators	50
Validation protocol upgrade	Validation V3	An upgrade to the validation protocol that introduces a batch verification process for approval messages, enhancing the overall performance of the protocol.	100
	Assignments V2	A new version of assignment certificates for tranche0 assignments, reducing the number of assignments needed per candidate and improving efficiency.	100
Disabled validators	Disabling validators in GRANDPA	Temporarily ignores votes from misbehaving validators in the GRANDPA consensus protocol.	100
	Disabling validators in BABE	Temporarily ignores votes from misbehaving validators in the BABE consensus protocol.	100
	Disabling validators in approval mechanism	Temporarily ignores votes from misbehaving validators during the approval voting process.	100
	Disabling validators in dispute	Temporarily ignores votes from misbehaving validators during the dispute resolution process.	100

	mechanism (done)		
	Disabling validators in backing	Temporarily ignores votes from misbehaving validators during the candidates backing process	100
	Zombienet test	Development of zombienet scenarios ensuring correct work of disabled validators feature. Fixing and debugging discovered issues	100
Systematic chunks		Enhancement for data retrieval using systematic chunks for data recovery	200
	Zombienet test	Fixing and debugging of discovered issues after zombienet development.	100
Minor features and improvements	Storage proof size host api	Allows runtime authors to enhance block utilization by retroactively recovering unused storage weight.	50
	DHT Authority discovery record creation time	Extends the DHT authority discovery records for nodes to determine which record is newer.	150
	Update initialize block runtime api	Introduces breaking changes to the Core runtime API by letting Core::initialize_block return an enum.	50
	Optimize SCALE	Scale codec improvement	50
	Qtils	Library for common utils among KAGOME and cpp-libp2p projects	150
DevOps and QA maintenance (10 months)			1500

Project management (10 months)			**1000**
Total			**5450**

Hourly rate: 100$/h
Cost (in USD): **$545000**
Cost (in DOT): **86507**

SRLabs security assurance team

1. Code assurance lead
2. Senior code assurance auditors x 3
3. Expert code assurance auditor x 1

Task	Description	Hours	Costs
IX.1 Conduct scheduled security audits and follow milestone PRs of KAGOME	a) Perform security audits on the new features developed since the initial audit, including: • Asynchronous backing • Elastic scaling • Validation v3 • Disabling validators • Grid and cluster topologies • WasmEdge integration • Systematic chunks • Reviewing minor enhancements • Security improvements (retroactive) b) Security review of pull requests (PRs) submitted to the KAGOME repository, guided and prioritized by KAGOME development team	1600	$291200

IX.2 Additional security assurance measures upon request	Expert consultation on additional security measures (e.g. dynamic testing improvements, security best practice workshops)	300	$54600

Hourly rate: 182$/h

Cost (in USD): **$345800**

Cost (in DOT): **54888**

Total cost (Quadrivium + SRLabs)

Cost (in USD): **$890800**

DOT/USD 30 day EMA (as of 28.06.24): $6.3

Cost (in DOT): **141 395**

Mentoring and technical support

The Web3 Foundation will partner on this proposal as technical advisor and deliverables auditor to ensure the completed milestones are technically sound. It is expected for the Quadrivium team to publish deliverable reports after the Web3 Foundation technical evaluation and before continuing with subsequent milestones or proposals.
In turn KAGOME team will help W3F improve Polkadot Host specification by reviewing spec changes. Kamil (KAGOME project lead) has already joined the Spec committee to decentralize the process of spec development.

Please note funds will come from the community treasury and Web3 Foundation technical team has no control over these funds and will not be rewarded as a reviewer of the milestones. The team will serve the sole purpose of evaluating deliverables in alignment with the community approval of this proposal and this role is based on their past participation in this project.

Appendix & additional information:

KAGOME presentations:

✅ KAGOME: C++ implementation of PRE presentation at DOTCon, August 18th, 2019
✅ Web3 Builders: Soramitsu | C++ Implementation of Polkadot Host, April 21th, 2020
✅ Building alternative clients | Polkadot Decoded 2023
✅ Polkadot Host architecture in 2024 | Sub0 Asia 2024

About Quadrivium

Quadrivium (https://www.qdrvm.io) is a blockchain infrastructure development company founded in 2023. The company specializes in the development of blockchain clients, peer-to-peer networking tools, and zk-cryptography. Quadrivium develops KAGOME Polkadot Host implementation, in partnership with the Web3 Foundation. The company also maintains the C++ libp2p library.

Quadrivium's mission is to build the infrastructure for a decentralized future. The company believes that blockchain technology has the potential to revolutionize many industries, and it is committed to developing the open-source tools and services that will make this possible.

Team experience

- https://github.com/libp2p/cpp-libp2p/ – official implementation of libp2p – a modular, upgradable network stack providing convenient interface for networking layer in p2p networks
- https://github.com/filecoin-project/cpp-filecoin – C++ implementation of Filecoin network protocol
- https://github.com/hyperledger/iroha – permissioned blockchain from Hyperledger umbrella, that is currently being used in Cambodian CBDC system

About SRLabs

SRLabs (https://www.srlabs.de/) is home to knowledge leaders securing critical infrastructures in finance, blockchain, energy, and telecommunications.
We focus on hands-on hacking resilience – not compliance –, which we shape by combining our hacking research with impactful consulting work for innovation leaders that have a natural thrive for cutting-edge technologies.
SRLabs is one of the leading blockchain audit companies with experience in many Substrate-based blockchains, including the Polkadot layer-0 relay chain and parachains built on top.