КІБЕРБЕЗПЕКА ТА ЗАХИСТ ІНФОРМАЦІЇ

UDC 004.056 + 004.738.5 : 004.056.53
DOI: https://doi.org/10.17721/ISTS.2025.9.5-10

Maksym KOTOV, PhD Student
ORCID ID: 0000-0003-1153-3198
e-mail: maksym_kotov@ukr.net
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

CLUSTRER STATE COORDINATION REDUCERS BASED ON RSDP

Background. *Contemporary distributed systems tend to leverage complex network topologies and intercommunication technologies for performing complex computational tasks. It is quite common for such systems to rely on centralized coordination where one or a group of designated servers serve as a management plane for the entire network. While this approach may simplify the consensus process, it introduces additional requirements for infrastructure engineering and maintenance. Nodes serving as a control plane require additional hardware resources as well as human effort and competence to manage external services capable of providing basic coordination primitives. Most cases requiring consensus could be reduced to simplistic procedures like gathering knowledge about network participants and dividing deterministically state slices between them. Therefore, implementing a complex coordination solution that requires an additional maintenance method could be inefficient. The Replica State Discovery Protocol stands as a lightweight coordination solution presenting a simple interface for achieving consensus between nodes within a cluster.*

Methods. *Within the ambit of this research, a set of cluster state reducers is described, providing basic coordination capabilities, including the formation of a network participants list and task division between them. Using mathematical modeling, we describe the procedures necessary for performing the said coordination tasks. Implementation and testing of reducers is done with the Node.js platform capable of running JavaScript code on the server side. A theoretical analysis and description of the proposed methods for distributed coordination are provided within this work to facilitate their integration into modern systems.*

Results. *As a result of this research, we propose three new cluster state reducers serving as methods of basic coordination capabilities as an exemplary application of RSDP. The first reducer is responsible for gathering the directory of participating nodes within a cluster and maintaining their statuses based on the received data. The second reducer performs a timeframe division within a cluster between the nodes to coordinate their execution in a mutually exclusive environment. Lastly, the rate limit reducer describes a logic to perform consensus regarding a single value that should be shared throughout the system as well as promptly updated if needed.*

Conclusions. *While engineering a complex distributed system requiring consensus among its subsystems or a set of homogenous components, it's important to avoid complexities related to the management of additional infrastructure while still providing a required level of consistency and availability. Having said that, the Replica State Discovery Protocol provides essential lightweight capabilities to resolve the said problem through the means of its flexible state reducers system. RSDP is built with a layered architecture in mind, capable of adjusting to the particular needs of the network as shown in this paper. By leveraging existing communication infrastructure and avoiding redundant management layers, RSDP allows for significantly reducing the computational complexity of coordination as well as costs associated with hardware needed for running a dedicated control plane. State reducers described within this article provide basic capabilities required for the most common coordination tasks, including the construction of a participants directory, task splitting and assignments, as well as consensus regarding the configuration parameters.*

Keywords: *distributed computing; device coordination; state synchronization; cluster management; service availability; security incidents; Replica State Discovery Protocol (RSDP); RSDP cluster state reducers.*

Background

The rise of the information technology age has led to the development of numerous automated systems for data gathering and processing. It is quite common nowadays to have shops or government services available directly on personal computers or phones. As a result, the demand for the availability of such services rises every year. The larger the number of clients trying to gain access to the system, the more it is evident that every large system must be capable of scaling both vertically and horizontally.

Vertical scaling is straightforward; increasing the number of CPUs or available RAM can provide a way of improving the system's productivity without modifying the software. Modern server stations can reach significant capabilities in their computational capacities, and the ceiling keeps getting higher. Although, vertical scaling is still quite limited when it comes to high-end, in-demand services such as AWS or Google Cloud. In cases where millions or billions of potential clients must be supported, vertical scaling is limited and will not be sufficient (Ali, 2019).

Horizontal scaling, on the other hand, promises nearly limitless growth potential where there are no commonly shared bottlenecks. Distributed systems primarily rely on such principles to provide both efficiency and availability through redundancy and coordination mechanisms. It comes with the cost of having to establish a consensus mechanism on a software level which is a non-trivial task tackled from the dawn of the global interconnection era (Ali, 2019; Millnert, & Eker, 2020).

With that, the Replica State Discovery Protocol serves as a solution, providing basic simplified interfaces for establishing a consistent and coordinated cluster environment. The purpose of this protocol is to abstract out the common steps required to aggregate a shared state among multiple nodes. Firstly, such steps include the discovery phase called “DEBATE”. Its purpose is to introduce a node to the network by broadcasting its initial configuration along with its identifier and network address. As a result, every node after receiving such notification will respond with its own configuration to complete the discovery process. The following step includes state derivation and consensus based on the statuses received from the peers. After initial aggregation, the nodes share their perspective with other participants to achieve consensus. This essentially is a security mechanism to avoid both accidental and intentional discrepancies within the cluster state (Toliupa et al., 2024; Kotov et al., 2024).

ISSN 2707-1758

Comparing RSDP with other coordination solutions like Apache ZooKeeper and ETCD, the difference becomes apparent (Junqueira, & Reed, 2013; Toasa et al., 2019; Nalawala et al., 2022; Larsson et al., 2020). The said services act as a central management layer distinct from the operational plane. As such, they require additional maintenance and resources tied to the deployment of the control-plane servers. In contrast, RSDP is a lightweight protocol that does not require deployment and management of any additional infrastructure. In that regard, it is more comparable to other consensus protocols such as Paxos and Raft (Kirsch, & Amir, 2008; Hu, & Liu, 2020; Ni et al., 2024). Though, a significant difference is that these protocols serve as a solution to elect a single value within the network, while RSDP, besides consensus as a final stage, provides capabilities to establish complex state derivation logic based on the available cluster perception.

RSDP itself is based on a layered model described within its own dedicated articles. In essence, there are application-level state reducers, a consensus layer, and media access layers. This article primarily concentrates on the first, but for completeness, we will briefly cover other interaction layers. We've already described the consensus layer and its operation principles above, when discussing the steps the protocol engine takes to exchange the information among nodes and establish a consistent environment.

The communication media layer is primarily responsible for providing two simple methods to the upper layers: send to and broadcast. These could be implemented in multifarious ways by leveraging existing interconnection protocols for security or reliability. The initial version of RSDP utilized the AMQP protocol and RabbitMQ implementation to establish such operations. AMQP is an asynchronous communication protocol based on queues and provides complex message routing capabilities for its clients. Though RSDP does not require AMQP nor RabbitMQ, the same logic could be implemented with simple HTTP requests or even based on bare TCP connections, which in effect removes dependency on any additional external services and reduces complexities tied to their management.

Having said that, within this article, the aim is to develop a set of coordinating state reducers in cases where interval services rely upon the availability of external servers. That is a quite common pattern nowadays, when most applications are some sort of combination of external tools. Such tools often implement their own security policies to guarantee availability in the face of potential DOS attacks. Hence, they implement various rate-limiting strategies to stifle any potential malicious activities. On top of that, it is quite common for such external services to provide their internal state data as snapshots. For example, some aggregates of financial or crypto-related information on centralized exchanges have limited historical availability or none.

In such cases it is important to establish data redundancy techniques where multiple internal processes within the controlled environment poll the external data provider. In case one polling process fails, the other would gather information successfully. Implementation of such logic becomes quite complex since the external service might impose rate-limiting restrictions, which necessitate the creation of internal coordination logic to avoid violation of such policies by a pool of uncoordinated parallel execution entities.

The purpose of the article. The purpose of this article is to provide examples of potential RSDP application within coordination-related problems. To do so, we've chosen a commonly met problem while designing a system requiring data redundancy and at the same time relying on third-party servers. An architectural description of such a case is provided with diagrams to establish the necessary context. This paper presents three state reducers providing functionality required to establish a managed and compliant gathering of data from an external limited source.

Starting with a directory-gathering reducer, responsible for maintaining a real-time list of current cluster members. As will be shown later, such a primitive operation is the basis of most coordination-requiring tasks since, in order to partition the task, a holistic view of the network is required in most cases. Given that capability, we develop a second state reducer tasked with timeframe division among the nodes, allowing us to mitigate risks associated with potential violation of security policies imposed by external services. Lastly, we develop a rate-limit state reducer responsible for maintaining a synchronized configuration of remote security policy. Since that configuration could be dynamic, we describe the communication channels and methods that could be used to keep the cluster state in a consistent manner.

Analysis of literary sources. Consensus and coordination have been actively studied in numerous works throughout the world and are extremely relevant topics especially in context of emerging blockchain technologies (Yaga et al., 2019). Nevertheless, the category of state management methods that do not involve external coordination plane remains an open area requiring additional effort and discussion.

The description and evaluation of centralized coordination methods based on ZooKeeper is done within works of Junqueira, F., & Reed, B. (2013). Similarly, contribution towards utilization of ETCD service and its capabilities are provided within works of Nalawala, H. S. et al. (2022).

Studies of Paxos consensus protocol were conducted within the works of Van Renesse, R., & Altinbuken, D. (2015). Respectively, research on Raft consensus protocol and its properties is described within papers of Hu, J., & Liu, K. (2020); Ni, L. et al. (2024).

Research on DDOS and its variations has been going on since the early 2000s. Significant contributions within the scope of taxonomy, detection and prevention methods are provided by the works of Mirkovic, J., & Reiher, P. (2004); Douligeris, C., & Mitrokotsa, A. (2004). Farther improvement of protection models and methods are discussed in Srivastava, A., et al. (2011). More recent papers involving descriptions of using artificial intelligence capabilities to detect DDOS attacks are done within works of Zhang, B., Zhang, T., & Yu, Z. (2017).

The Replica State Discovery Protocol, its implementation details, terminology and phases are described within the works of Toliupa, S. et al. (2024). The media access layer based on AMQP, its architecture, properties, and characteristics are discussed in Kotov, M. S. et al. (2024).

Methods

Within the ambit of this research, a set of cluster state reducers is described, providing basic coordination capabilities, including the formation of a network participants list and task division between them. Using mathematical modeling, we describe the procedures necessary for performing the said coordination tasks. Implementation and testing of reducers are done with the Node.js platform capable of running JavaScript code on the server side. A theoretical analysis and description of the proposed methods for distributed coordination are

ISSN 2707-1758

provided within this work to facilitate their integration into modern systems.

Additionally, this paper provides a description of coordination-requiring tasks related to the rate-limited external services. The architectural model of the potential clustered data-gathering solution is laid out and evaluated using diagrams and mathematical notations. As we propose new coordination methods based on RSDP's reducer interface, we've described their potential integration within the scope of such systems.

Results

The following section presents a description of the proposed state reducers aimed at facilitating cluster coordination. We will first start with the description of the cluster requirements and the general architecture that could be encountered in cases where there is a combination of external service, temporary availability of data, and rate-limiting security policies in place. We will define the general approach towards request limitation commonly done by external service providers. In addition, a diagram of architecture will be used to further clarify the setup.

Architecture of a system with external limited dependencies. Let us start the discussion of results with the definition of the target distributed system. Before proceeding with the description of cluster state reducers, we will first define the network topology and its constraints. We will shortly discuss external security policies and their implications while building data-gathering or synchronization services for temporarily available aggregates.

The following Fig. 1 depicts the architecture of such a system:

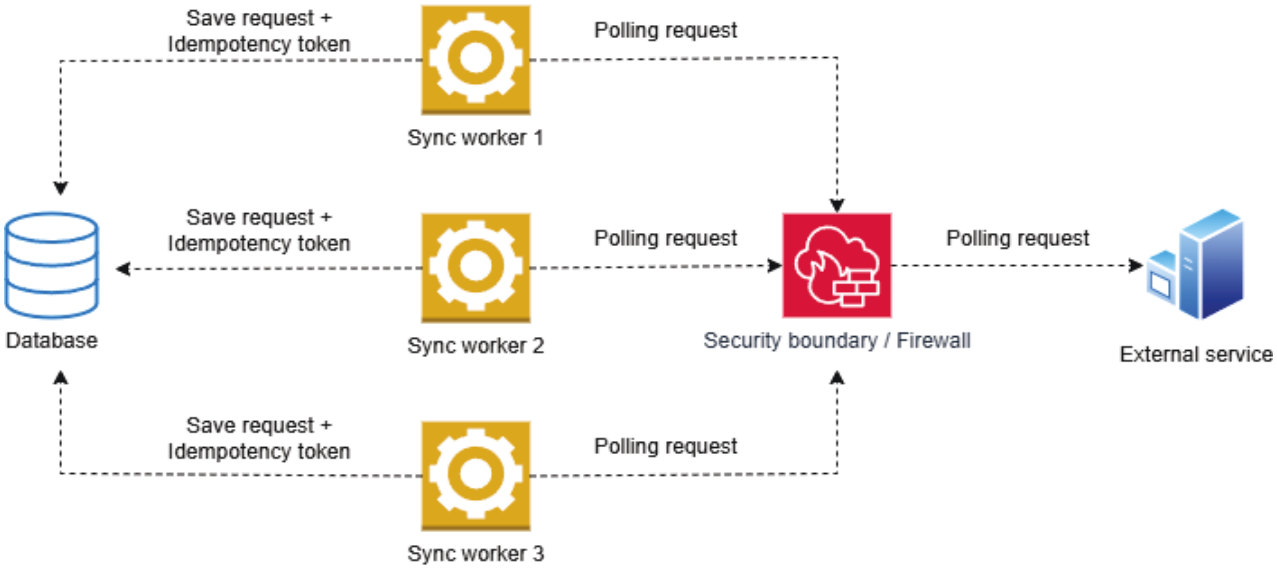

Fig. 1. Architecture of the target distributed system

This system is comprised of multiple synchronization workers that form an RSDP cluster. Their main task is to continuously poll data from an external service provider. The conditions that make this task complex are as follows:

- The external service provides data aggregations temporarily, meaning that historical reconciliation is impossible since older records will not be available in the future.
- At the same time, the service has a security policy that restricts incoming requests with a shared identifier within a time window.
- Controlled service is supposed to minimize data loss through any available means.

Hence, the architecture of a controlled network is comprised of multiple duplicate polling processes allocated on different machines or even availability zones. At every collection period, all three must try to retrieve the data from the target service. If one of them failed, the other would try to save the records received.

Overall, the idea is to leverage redundancy to mitigate potential risks associated with data loss. This is a typical strategy met within many distributed systems.

In order to avoid data duplication inside the database, the idempotency token could be used to check the availability of the record. In the case of the described system, polling interval start and interval end, along with specific request type parameters, could be used as such tokens. Before inserting a new record in the database, every worker would first look to see if there is already a record stored with the mentioned field. To further strengthen consistency guarantees, a unique constraint policy could be implemented, which is commonly supported in RDBM systems and other databases.

The problem arises due to the potential security policies on the provider's side. Since rate limits block excessive requests to the service, our task is to ensure that all the parallel polling processes work in a coordinated way to avoid violations and possible further restrictions. This problem could be generalized to any type of access or work with a shared resource. When multiple parallel actors try to work on the same problem, coordination is required to avoid inconsistent behavior.

Rate-limiting policies come in multiple variations but could be generalized as an access sliding window. In that case, the external service would define a number of requests allowed within a specific timeframe. If that amount is exceeded, the sender might be subjected to additional restrictions or even a permanent ban from accessing the API (Serbout et al., 2023; Firmani, Leotta, & Mecella, 2019).

To describe how such policies work, let us declare the following:

- $R \in \mathbb{R}^+$: maximum request rate (requests per second);
- $T \in \mathbb{R}^+$: time window (in seconds);

ISSN 2707-1758

- $N(t)$: number of requests made in interval $[t - T, t]$;
- $u(t) \in \{0,1\}$: unit impulse indicating a request at time t.
- Now the constraint, or the rate limit condition could be expressed as:

$$\int_{t-T}^{t} u(\tau)\, d\tau \leq R \cdot T, \quad \forall t \in \mathbb{R}^{+}. \tag{1}$$

Equivalently, in discrete time (e.g., if requests occur at discrete timestamps t_i):

Let $\{t_i\}_{i=1}^{n} \subset \mathbb{R}^{+},\ t_i < t_{i+1}$. Now define the windowed count:

$$N(t) = \sum_{i=1}^{n} \mathbb{1}_{[t-T,t]}(t_i). \tag{2}$$

The rate limit condition could be expressed as:

$$N(t) \leq R \cdot T. \tag{3}$$

The acceptance function $a(t_i) \in \{0,1\}$ could be defined as:

$$a(t_i) = \begin{cases} 1 \ \ if\ N(t_i^-) < R \cdot T, \\ 0 \ \ if\ otherwise. \end{cases} \tag{4}$$

That being said, we can proceed with the definition of coordination mechanisms capable of working with such external constraints to avoid potential security incidents.

The state reducer interface notation. Before proceeding with descriptions of the proposed state reducers, we will first have to establish a mathematical notation basis that will be used throughout these definitions.

Firstly, the cluster itself could be defined as a graph $G_{\text{cluster}} = (V, E)$, with a set of replicas $V = \{v_1, v_2, \dots, v_n\}$ and a set of directed edges $E \subseteq V \times V$. In this network each node has its own initial state s_i used to later derive an aggregated state s_i^* along with incoming initial states from in-neighbors $\mathcal{N}^-(v_i)$. The process could be defined as follows (Toliupa et al., 2024):

$$s_i^* = f_{agg\ status}\left(s_i, \{M_{status}(v_j) \mid v_j \in \mathcal{N}^-(v_i)\}\right) \tag{5}$$

Here, $M_{status}(v_j)$ represents an incoming message from in-neighbor in response to initial $M_{hello}(v_i)$. This message could be then defined as $M_{status}(v_j) = \left(f_{\text{id}}(v_j), f_{meta}(v_i), s_j\right)$. We've additionally defined utility functions $f_{id}(v_j)$ that returns a routable identifier for v_j and $f_{\text{meta}}(v_i)$ that returns meta information about participating node (Toliupa et al., 2024).

Later, we will use $s_i^* = f_{agg\ status}$ notation to describe every subsequent state reducer aggregation function. Having said that, let us proceed with the state reducers definition.

Cluster members state reducer. This reducer is one of the most commonly used to resolve any coordination task. That becomes apparent since, in order to distribute tasks between participants, their complete list is required along with the metadata, such as their capacity and capabilities.

The reducer could be described as follows:

$$s_i^* = \left\{ f_{id}(v_j), f_{meta}(v_i), s_j \mid v_j \in \mathcal{N}^-(v_i) \cup v_i \right\}. \tag{6}$$

Its implementation is quite simple but equips the network with an important ability to discover peers.

Timeframe division state reducer. Now, when we can get a list of network participants, we can create an additional state reducer that is capable of dividing timeframes among nodes. Its implementation could be farther augmented by introducing sharding categories, in case there are multiple API keys, but the general approach could be defined as:

$$s_i^* = \left\{ \left(f_{id}(v_j), \frac{\Delta T}{|\mathcal{A}|} \cdot rank\left(f_{id}(v_j), \mathcal{A}\right) \right) \,\middle|\, v_j \in \mathcal{N}^-(v_i) \cup v_i \right\}, \tag{7}$$

where, $\mathcal{A} = \{f_{id}(v_j) \mid v_j \in \mathcal{N}^-(v_i) \cup v_i\}$, $|\mathcal{A}|$ is the amount of node addresses registered within $\mathcal{A}$. Then $rank\left(f_{id}(v_j), \mathcal{A}\right)$ is a function that returns the position of $f_{id}(v_j)$ within a sorted set $\mathcal{A}$ for a given node v_j.

Also, ΔT is a synchronization period that could be obtained either as a static configuration shared in the cluster, or dynamically calculated based on rate limit as follows:

$$\Delta T = |\mathcal{A}| \cdot \frac{1}{R}, \tag{8}$$

where inverse of R basically represents how many seconds a process should wait before it can issue another request in compliance with the restriction policies.

This reducer could be farther improved to support multiple parallel limitation scopes. Let's say that within the observed architecture there are multiple access keys, each having its own dedicated rate limit policy and counter. In that case, it would be more efficient to separate coordination tasks in the following manner:

$$s_i^* = \left\{ \left(f_{id}(v_j), \frac{\Delta T}{|\mathcal{A}(f_{group}(v_j))|} \cdot rank\left(f_{id}(v_j), \mathcal{A}(f_{group}(v_j))\right) \right) \,\middle|\, v_j \in \mathcal{N}^-(v_i) \cup v_i \right\}, \tag{9}$$

where $\mathcal{A}(L_k)$ is now a function that returns a subset of nodes belonging to a particular subset of $\mathcal{A}$ where each member is associated with a label L_k. Here, $f_{group}(v_j)$ would return L_k associated with v_j.

With that setup it is now possible to distribute tasks among nodes in several groups. Each group would have its own isolated access context and would still be capable of coordinating separately.

Rate limit state reducer. It is quite common for the R value to be dynamic itself. These policies tend to get updated over time due to the demand and capability factors on the external service's side. Hence, one of the key state reducers is responsible for maintaining this value in a consistent manner:

$$s_i^* = \left\{ f_{rl}(v_j) \mid v_j \in \mathcal{N}^-(v_i) \cup v_i \right\}, \tag{10}$$

where $f_{\text{rl}}(v_j)$ returns rate limit value observed at node v_j.

The RSDP has undergone multiple iterations, some of which perform popular vote decisions on aggregated states. That implies that propagation of the rate limit value should be done through side channels as shown in the following Fig. 2.

Since a new value in such RSDP version would only be accepted if most of nodes share it, without side channel such propagation would be slow. Another approach is to pass timestamp t along with $f_{rl}(v_j)$ and make the state derivation function find such $f_{rl}(v_j)$ that is associated with the largest value t.

ISSN 2707-1758

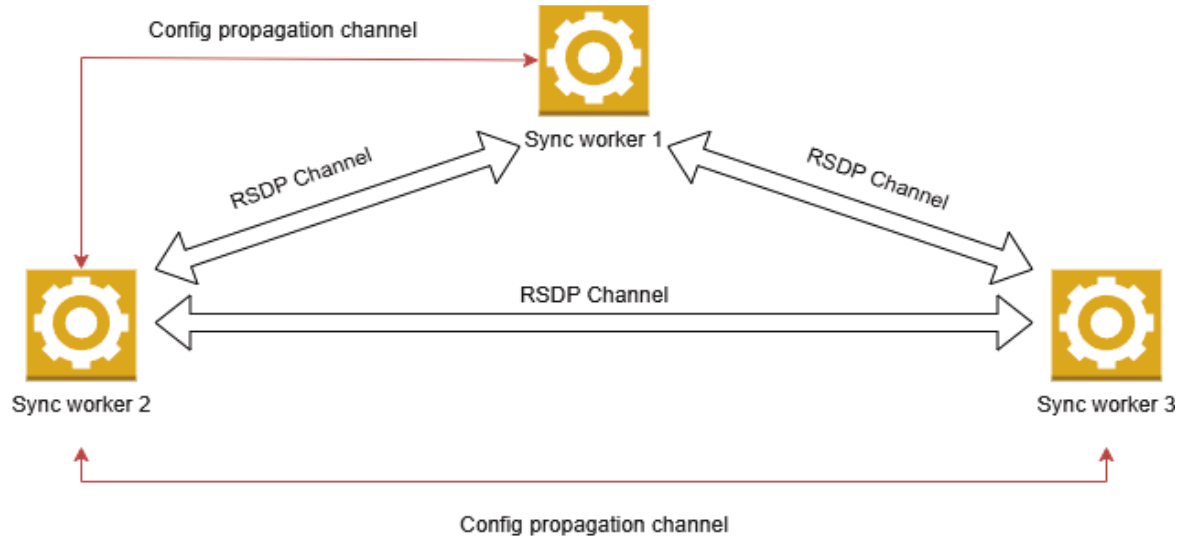

Fig. 2. RSDP initial state updates propagation

Discussion and conclusions

While engineering a complex distributed system requiring consensus among its subsystems or a set of homogenous components, it's important to avoid complexities related to the management of additional infrastructure while still providing a required level of consistency and availability. Having said that, the Replica State Discovery Protocol provides essential lightweight capabilities to resolve the said problem through the means of its flexible state reducers system. RSDP is built with a layered architecture in mind, capable of adjusting to the particular needs of the network as shown in this paper. By leveraging existing communication infrastructure and avoiding redundant management layers, RSDP allows for significantly reducing the computational complexity of coordination as well as costs associated with hardware needed for running a dedicated control plane. State reducers described within this article provide basic capabilities required for the most common coordination tasks, including the construction of a participants directory, task splitting and assignments, as well as consensus regarding the configuration parameters.

Within this paper, we've covered a common architectural problem we met while integrating with myriads of external services. Since most SaaS platforms encounter spamming and DoS problems, most utilize rate-limiting policies for each connecting client. While mitigating availability risks on the server's side, it significantly complicates data redundancy techniques and distributed polling for clients, which require coordination capabilities.

As such, the proposed state reducers provide all the necessary information for running polling processes to both retrieve the data and comply with external security restrictions. The demonstrated reducers allow you to gain a complete network view, starting with the participants list. The list serves as basic information for more complex state division among nodes. By knowing how many nodes there are in the network and being able to dynamically adjust the list according to their health or availability, it is possible to construct more complex coordination methods. Hence, the second state reducer, based on the list, divides the available timeframes among the participants, allowing them to avoid collisions and accidental bans from the target API. At the same time, the third reducer provides a real-time configuration for the available rate limit policy.

Overall, the purpose of this article is to show examples of RSDP's potential application in solving coordination-related tasks. We've laid out the direction of the state management capabilities available through RSDP interfaces and their versatility. It is the intent of this paper to spark further research into the application of RSDP's capabilities in solving non-trivial tasks in contemporary parallelized systems requiring access to a shared resource.

Author's contribution: Maksym Kotov – conceptualization, methodology, formal analysis, development of software, analysis of sources, preparation of a literature review and theoretical foundations of research, editing and reviewing.

Sources of funding. This study did not receive any grant from a funding institution in the public, commercial, or non-commercial sectors.

References

Ali, A. H. (2019). A Survey on Vertical and Horizontal Scaling Platforms for Big Data Analytics. *International Journal of Integrated Engineering*, 11(6), 138–150. https://publisher.uthm.edu.my/ojs/index.php/ijie/article/view/2892

Douligeris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: Classification and state-of-the-art. *Computer Networks*, 44(5), 643–666. https://doi.org/10.1016/j.comnet.2003.10.003

El Malki, A., Zdun, U., & Pautasso, C. (2022). Impact of API rate limit on reliability of microservices-based architectures. In B. Smith, & J. Doe (Eds.), *Proceedings of the 2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)* (pp. 19–28). IEEE Press. https://doi.org/10.1109/SOSE55356.2022.00009

Firmani, D., Leotta, F., & Mecella, M. (2019). On computing throttling rate limits in web APIs through statistical inference. In J. Zhang & H. Zhao (Eds.), *Proceedings of the 2019 IEEE International Conference on Web Services (ICWS)* (pp. 418–425). IEEE Press. https://doi.org/10.1109/ICWS.2019.00075

Hu, J., & Liu, K. (2020). Raft consensus mechanism and the applications. *Journal of Physics: Conference Series*, 1544(1), 012079. https://doi.org/10.1088/1742-6596/1544/1/012079

Junqueira, F., & Reed, B. (2013). *ZooKeeper: Distributed process coordination.* O'Reilly Media, Inc. https://www.oreilly.com/library/view/zookeeper/9781449361297/

Kotov, M. S., Toliupa, S. V., & Nakonechnyi, V. S. (2024). Method of building local area network simulation based on AMQP and its support protocols suite. *Telecommunication and Information Technologies*, 3, 102–119 [in Ukrainian]. [Котов, М. С., Толюпа С. В., Наконечний В. С. (2024). Метод побудови симуляції локальної мережі на основі amqp та набір протоколів підтримки. *Телекомунікації та інформаційні технології,* 3, 102–119]. https://doi.org/10.31673/2412-4338.2024.039989

Millnert, V., & Eker, J. (2020). HoloScale: Horizontal and vertical scaling of cloud resources. In A. Editor, & B. Editor (Eds.), *Proceedings of the 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)* (pp. 196–205). IEEE Press. https://doi.org/10.1109/UCC48980.2020.00038

ISSN 2707-1758

Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Computer Communication Review*, 34(2), 39–53. https://doi.org/10.1145/997150.997156

Nalawala, H. S., Shah, J., Agrawal, S., & Oza, P. (2022). A comprehensive study of “etcd”– An open-source distributed key-value store with relevant distributed databases. In Emerging Technologies for Computing, *Communication and Smart Cities. Lecture Notes in Electrical Engineering*, 875. Springer, Singapore. https://doi.org/10.1007/978-981-19-0284-0_35

Ni, L., Ye, Q., Yang, J., Zhang, S., & Xian, M. (2024). Research of Raft algorithm improvement in blockchain. In X. Editor, & Y. Editor (Eds.), *Proceedings of the 2024 IEEE 16th International Conference on Advanced Infocomm Technology (ICAIT)* (pp. 290–294). IEEE Press. https://doi.org/10.1109/ICAIT62580.2024.10808140

Serbout, S., El Malki, A., Pautasso, C., & Zdun, U. (2023). API rate limit adoption: A pattern collection. In J. Noble, & K. Beck (Eds.), *Proceedings of the 28th European Conference on Pattern Languages of Programs (EuroPLoP '23)* (Article 5, pp. 1–20). Association for Computing Machinery. https://doi.org/10.1145/3628034.3628039

Srivastava, A., Gupta, B. B., Tyagi, A., Sharma, A., & Mishra, A. (2011). A recent survey on DDoS attacks and defense mechanisms. In Advances in parallel distributed computing. PDCTA 2011. *Communications in computer and information science*, 203. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24037-9_57

Toliupa, S., Kotov, M., Buchyk, S., Boiko, J., & Shtanenko, S. (2024). Stateful cluster leader failover models and methods based on Replica State Discovery Protocol. *Proceedings of the IT&I Workshops 2024*, 120–140. https://ceur-ws.org/Vol-3933/Paper_10.pdf

Van Renesse, R., & Altinbuken, D. (2015). Paxos made moderately complex. ACM Computing Surveys, 47(3), Article 42, 1–36. https://doi.org/10.1145/2673577

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2019). *Blockchain technology overview.* National Institute of Standards and Technology Internal Report. https://doi.org/10.48550/arXiv.1906.11078

Zhang, B., Zhang, T., & Yu, Z. (2017). DDoS detection and prevention based on artificial intelligence techniques. In M. Chen, & L. Wang (Eds.), *Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC)* (pp. 1276–1280). IEEE Press. https://doi.org/10.1109/CompComm.2017.8322748

Отримано редакцією журналу / Received: 17.05.25
Прорецензовано / Revised: 27.05.25
Схвалено до друку / Accepted: 13.06.25

Максим КОТОВ, асп.
ORCID ID: 0000-0003-1153-3198
e-mail: maksym_kotov@ukr.net
Київський національний університет імені Тараса Шевченка, Київ, Україна

РЕДУКТОРИ СТАНУ ДЛЯ КООРДИНАЦІЇ КЛАСТЕРА НА ОСНОВІ RSDP

В с т у п . *Сучасні розподілені системи, зазвичай, використовують складні мережні топології та технології внутрішнього зв'язку для виконання складних обчислювальних завдань. Досить часто такі системи покладаються на централізовану координацію, коли один або група призначених серверів є площиною управління для всієї мережі. Хоча цей підхід може спростити процес досягнення консенсусу, він вводить додаткові вимоги до проєктування та обслуговування інфраструктури. Вузли, які є площиною управління, вимагають додаткових апаратних ресурсів, а також людських зусиль і компетенції для управління зовнішніми службами, здатними надавати базові примітиви координації. Більшість випадків, що вимагають консенсусу, можна звести до спрощених процедур, таких як збір інформації про учасників мережі та розподіл детермінованих зрізів стану між ними. Отже, впровадження складного координаційного рішення, яке вимагає додаткового методу обслуговування, може бути неефективним. Протокол виявлення стану репліки (RSDP) виступає як "lightweight" рішення для координації, що представляє простий інтерфейс для досягнення консенсусу між вузлами у кластері.*

М е т о д и . *У межах цього дослідження описано набір редукторів стану кластера, що забезпечує базові можливості координації, включаючи формування списку учасників мережі та розподіл завдань між ними. За допомогою математичного моделювання описано процедури, необхідні для виконання зазначених координаційних завдань. Впровадження та тестування редукторів виконується за допомогою платформи Node.js, здатної запускати код JavaScript на боці сервера. Теоретичний аналіз і опис пропонованих методів розподіленої координації представлено в цій роботі для полегшення їхньої інтеграції в сучасні системи.*

Р е з у л ь т а т и . *У результаті цього дослідження ми пропонуємо три нових редуктори стану кластера, які являють собою методи базових можливостей координації як зразкове застосування RSDP. Перший редуктор відповідає за збір каталогу вузлів-учасників у кластері та підтримку їхніх статусів на основі отриманих даних. Другий редуктор виконує розподіл часових термінів у межах кластера між вузлами для координації їхнього виконання у взаємовиключному середовищі. Нарешті, редуктор обмеження швидкості описує логіку для досягнення консенсусу щодо єдиного значення, яке має бути спільним для всієї системи, а також негайно оновлюватися за потреби.*

В и с н о в к и . *Під час проєктування складної розподіленої системи, що потребує консенсусу між її підсистемами або набором однорідних компонентів, важливо уникати складнощів, пов'язаних з управлінням додатковою інфраструктурою, забезпечуючи водночас необхідний рівень узгодженості та доступності. Зважаючи на це, протокол виявлення стану репліки надає важливі полегшені можливості для розв'язання зазначеної проблеми за допомогою своєї гнучкої системи редукторів стану. RSDP створено з урахуванням багаторівневої архітектури, здатної адаптуватися до конкретних потреб мережі, як показано в цій статті. Використовуючи наявну комунікаційну інфраструктуру й уникаючи надлишкових рівнів управління, RSDP дозволяє значно зменшити обчислювальну складність координації, а також витрати, пов'язані з апаратним забезпеченням, необхідним для роботи спеціальної площини управління. Редуктори стану, описані в цій статті, надають базові можливості, необхідні для найпоширеніших завдань координації, включаючи створення каталогу учасників, розподіл завдань і призначення, а також консенсус щодо параметрів конфігурації.*

К л ю ч о в і с л о в а : *розподілені обчислення; узгодження пристрою; синхронізація стану; управління кластерами; доступність сервісів; інциденти безпеки; протокол виявлення стану репліки (RSDP); редуктори стану кластера RSDP.*

Автор заявляє про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; в рішенні про публікацію результатів.

The author declares no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; in the decision to publish the results.

ISSN 2707-1758