Cudo decentralised compute ecosystem
Technical Paper

Cudo Ventures

October 2020

Version 1.3

Abstract

This paper describes technical components of Cudo's distributed computing platform and of the CUDOS network, a multi-chain layer 2 solution for blockchains. The Cudo platform provides cloud-like services at affordable cost while rewarding users who contribute to the network. This document explains various components of the platform including benchmarking, job life cycle, the reputation system and the economic model, as well as privacy and security features. For transparency and security purposes, the Cudo compute network implements blockchain technology for payments and staking. Furthermore, the Cudo platform can be integrated into the CUDOS network, in order to provide further compute capacity to blockchains. This paper also outlines the token model for CUDOS, an ERC20 token used initially for staking and discounts on the platform, as well as for powering the CUDOS network.

Contents

1 Introduction

Cloud services provide essential tools for enterprises, as proven by the amount of new data centres operated by hyperscale providers [1]. However, the associated economical and ecological costs that this infrastructure involves are too great [2, 3], and only set to grow considering the increasing demand for these services [4]. Furthermore, the industry is now facing physical limitations [5]. There is a need for a new, cheaper and greener solution.

Similarly, the novel blockchain technology [6] is in need of a scaling solution. Proof-of-Stake (PoS) is an essential step forward, as Proof-of-Work (PoW) networks are not scalable and have a major ecological impact [7]. While Ethereum 2.0 with its sharding solution may solve most of the issues [8], its progress is slow, which has caused other projects to emerge [9]. However, there is no definitive solution yet, nor is there a clear path for the technology's mass-adoption.

In order to allow providers of cloud services to keep up with the increasing demand while reducing the pace of construction of new data centres, a solution would be to use already existing computing power which is currently unutilised. With billions of computing-capable devices owned by the general public [10, 11, 12] which are idle most of the time [13], and with the sharing economy being part of our everyday lives [14, 15], using people's computers and idle data centre time for this purpose and rewarding them for supplying the equipment to the network stands out as a way forward for the industry. Connecting this cloud alternative, in phased stages, to emerging technologies such as blockchain may be the required breakthrough to disrupt both markets and move society towards technological decentralisation.

The idea of using users computing power for a greater goal is not new. The Large Hadron Collider in CERN for instance has been using grid computing for years to analyse particle collision data [16], and anyone is welcome to volunteer their computing power to donate idle time to the LHC [17]. In this sense, distributed computing is just promoting and moving this idea outside of academia and to the general public. SETI@home [18] is another example.

With all this in mind, Cudo Ventures is adding compute and blockchain workloads to the current product with a distributed computing platform, as well as entering the blockchain space with the CUDOS network. This paper extends the general discussion of the platform given in the whitepaper [19], by providing technical details on how the platform functions. This technical paper is organised as follows: section 2 gives an overview of the cloud platform, and introduces some of its main components. The Cudo platform, an existing mining application which can be seen as a necessary preparatory step for its compute extension, is briefly summarised in section 3.

Sections 4 through 8 cover different aspects of the Cudo compute platform. Section 4 explains benchmarking of the connected devices and the platform's job scheduler. The types of workloads in the platform are described in section 5, and section 6 details the life cycle of a given compute job. In section 7 details are given on how failed jobs are handled, connecting with the reputation system presented in section 8.

Cudo compute initially uses the Ethereum network for some aspects, as explained in section 9. The payment process is detailed in sections 10 and 11, where the CUDOS token is also introduced. After the blockchain side of the cloud platform is explained, section 12 describes how the on-chain CUDOS network works to support offloading compute jobs via smart contracts. The connection between the Cudo platform for compute jobs and the blockchain CUDOS network is discussed in section 13. Last, section 14 gives a summary of the platform, and discusses some of the future developments for it.

2 Cloud platform overview

Cudo Ventures is building a decentralised grid compute network, with the architecture presented in figure 1.

2.1 First class entities

The system comprises the following first class entities:

Developers The main contributors to the app marketplace. Developers are encouraged and rewarded to develop apps using Cudo's guidelines and underlying technology.

Consumers Representing the demand side of the network, consumers provide workloads to the network and purchase compute. These entities push jobs into the platform either through an app or directly via the Cudo compute marketplace.

Suppliers Suppliers provide compute to the network either through mining or processing computational tasks. Suppliers run the Cudo application which intelligently

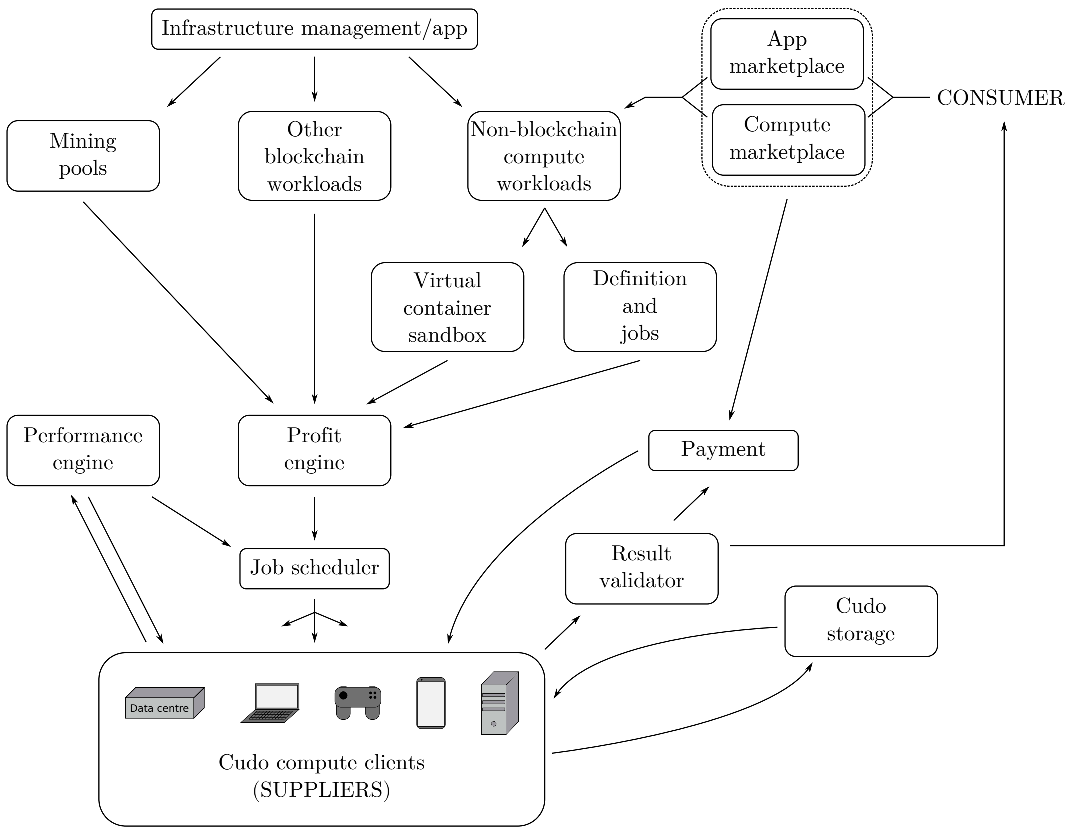

Figure 1: Diagram showing the architecture of the Cudo compute platform. It allows mining work, other blockchain workloads such as smart contracts and non-blockchain jobs, the focus of this paper. In this last case, consumers send the jobs directly or through an available app in the Cudo marketplace. The job is then processed and distributed to the different Cudo compute suppliers, which include laptops, PCs, servers, mining rigs, ASICs, FPGAs and, in the future, mobiles. The performance engine periodically benchmarks all clients to ensure that workloads are distributed correctly. Clients can also use or provide storage services on the network. Once the result is validated, payment is received by the supplier, and the output is returned to the consumer.

switches between types of workload based on various factors, detailed later in this paper. This process is performed in a transparent manner so that suppliers are aware of the nature of the current workload being performed.

Workloads Generic packages of work created by the Cudo platform and processed by suppliers. These may represent mining tasks, compute tasks or any other type of work the platform supports.

Cudo platform The Cudo platform is a decentralised compute ecosystem, currently supporting cryptocurrency mining workloads. The individual components of the compute platform are outlined in the next section.

2.2 Platform components

The primary components of the Cudo compute platform comprise those depicted in figure 1. Each component is outlined below, with Kubernetes [20] used to manage and deploy the full set.

Mining pools Mining pools allow workers to share their processing power over a network to split the block reward in proportion to the amount of work they contribute. Mining pool workloads are always available. For more details see section 3.

Other blockchain workloads Smart contract platforms, such as Ethereum, are mostly suited to run smart contracts only. Running complex workloads is highly inefficient due to the need for the network to reach consensus. Other blockchain workloads in the figure represent complex workloads offloaded from smart contracts, similar to the outsourcing of knowledge to oracles. The connection of these workloads for the Cudo platform with the CUDOS network is described in section 13.

Non-blockchain compute workloads Jobs from this workload type are the result of consumers ordering compute jobs either via a third party application or directly via their own process. Jobs are comprised of a virtual container sandbox and the job's associated definition as explained further in section 5.

Performance engine The performance engine periodically schedules workloads to benchmark the performance characteristics of the supplier's hardware, including computation, network and storage speed, latency and availability. The benchmark is used to determine how much the worker charges for workload execution per second. For further details see section 4.

Profit engine The profit engine calculates the profitability of workloads based on network and market conditions. For cryptocurrency mining, profitability is a function of network characteristics, such as the block reward, block time and network difficulty. Section 4 discusses further how job profitability is derived.

Job scheduler The job scheduler assigns pending workloads to workers based on the worker's availability and workload constraints. These constraints include hardware support, geographic location, minimum reputation and security certification. The job scheduler attempts to minimise the total cost of workloads being executed by comparing all the eligible workers for each job.

Payment The payment element is responsible for unlocking the stake and finalising a supplier's transaction in the case of a successful job, as determined by the result validator. The role of the transaction engine, which is part of the payment component, is detailed in the job life cycle description in section 6 and in figure 2.

Result validator The result validator runs the validation module for a given job type after its result has been uploaded to the Cudo platform by a supplier. The different types of possible validations are expanded upon in section 5.6.

Cudo Storage Cudo storage is a planned product enabling consumers to utilise the spare storage capacity of suppliers.

3 Current Cudo product

The Cudo platform is currently live and supports a single workload – cryptocurrency mining. This section describes the current version of the product. The following sections explain Cudo compute, the flagship application which extends the current one to a wide range of workloads, as well as its parallel blockchain part, the CUDOS network.

Supplier-side software A lightweight executable running on the supplier's hardware. Initially the software does not contain any mining or compute task knowledge, as this is delivered by the Cudo platform.

Benchmarking After installation, the software determines the specifications of the machine. These include details pertaining to GPU, CPU, memory and storage media, as well as network capabilities. In cases where the hardware is not already known to Cudo, such as when a new graphics card is released, the hardware is benchmarked by the software and the results returned to Cudo.

Since the mining process benchmarks while it runs, the process for unknown hardware is simply to iterate through and complete available work until further information is known.

Profitability and scheduling The Cudo platform determines the best mining task for an individual supplier by iterating through the available mining tasks, checking the market price of those assets and calculating the value that a supplier's work would generate during a given time window. This check is performed approximately every six hours and may eventually be completed at more regular intervals. In most cases the outcome is that the current tasks remain the most profitable.

In the case that the most profitable task is not the one the supplier is already running, the Cudo platform provides the necessary software components required to switch to it.

Payouts Work from each supplier is contributed to the Cudo pool, after which the resulting payout is distributed accordingly. The payouts correspond to the amount of work contributed by each supplier, which is calculated as a percentage of the total amount of work contributed by all suppliers. Payouts can currently be made in Bitcoin, Ethereum, Monero or Ravencoin, with the conversion performed close to the point of work (as opposed to the point of payout).

Cudo's cryptocurrency mining application also supports ASIC mining, for which the above benchmarks and configurations are not necessary, but which benefits from auto-tuning and other tailored approaches.

4 Benchmarking and scheduling

The benchmarking process is based on that of the current Cudo product, as detailed above. The client software downloads one job per piece of hardware which needs to be benchmarked, usually GPU, CPU, memory, storage media and networking capabilities. A score is derived for each piece of hardware based on how long the machine takes to run the benchmarking job.

The result of this process is a vector of name-value pairs which represents the processing power of the supplier's machine and is used in the Cudo Performance Engine.

Automated switching between job types After each job is finished, the platform determines the next-most profitable job for the supplier. Mining workloads are always available and form the default base workload. The most profitable mining task is calculated using the benchmarking process explained above.

If compute jobs are available, the Cudo platform assesses the suitability of the job for the supplier by checking a series of constraints. These include:

- Reputation (*e.g.* reliability)
- Security level (*e.g.* ISO certification)
- Location
- Performance (*e.g.* CPU IPS, GPU FLOPS, storage IOPS – see section 10)
- Capabilities (*e.g.* memory availability, storage availability, instruction set support)
- Availability (*e.g.* uptime, periods online).

If the supplier satisfies the constraints, the Cudo platform allocates the job to the supplier, usually priced at least three to four times the mining revenue for that supplier. See section 10 for a more in depth explanation of pricing. If multiple jobs are available, the Cudo platform delivers the most profitable job to the supplier.

The most common job type consists of batch work with a bounded duration, producing an output on completion. This often takes an input that can be divided to run asynchronously across multiple suppliers, with the outputs later recombined into the final product.

Workloads such as deep learning, big data and signal processing are suited to parallel batch operation across most types of suppliers. Vertically-scaling work involving for example relational databases is also properly handled and paired to appropriate suppliers, usually data centres.

5 Compute workload types

There are three workload types which generate generic workloads to be processed in a uniform manner by the Cudo platform.

5.1 Apps

Apps are created either by Cudo or (more frequently) by third party developers. Each app must define:

- A process for taking input (such as a video in the case of a transcoding app), which may be defined via a web UI or via an API.
- A workload image, usually created from the base image guidelines provided by Cudo.
- A configuration customisation that extends the base image to make it workload specific, typically in one of the following forms:

  - Scripting to download, install and set up software inside an operating system image *e.g.* Bash or PowerShell.
  - Container build instructions to create a template image *e.g.* Dockerfile.
  - Upload of a complete preconfigured image *e.g.* an OVF appliance or a Node.js package.

- An estimation component (optional) responsible for estimating the execution time of a job based on the input.
- A validation component (optional) responsible for any app-specific automated validation that can be used to verify a job has been completed properly.
- An ingest function (detailed in section 5.4).
- An egress function (detailed in section 5.5).

5.2 Custom compute workloads

Separate to using an app, consumers have the option to generate workloads directly. Similar to the apps, any custom compute workload should comprise of:

1. A workload image
2. An estimation component (optional)
3. A validation component (optional)
4. An egress function.

5.3 Test workloads

Used by Cudo periodically to validate suppliers, tests are workloads for which the expected output is known. Tests have the following components:

1. A workload image
2. An expected output
3. A validation component
4. An egress function.

After a job result is uploaded by the egress function, the validation component checks that the delivered work matches the expected output. Test workloads are used to measure the performance, integrity and availability of a device.

5.4 Ingest function

Each job type has a UI or API responsible for taking input. The ingest function produces a workload image from a given input. This logic may include support for splitting the job into sub-work packages for parallel processing.

Consider the following video transcoding example. At input, the source video is scanned at ingest and split at certain key frames. Each of the split sections is then packaged as a discrete job. Each video section is sent to an individual supplier, and the video transcoding operations run in parallel.

5.5 Egress function

The egress function resides on the workload image and runs on the supplier's hardware. It executes upon successful completion of a job and is responsible for taking the completed work and uploading it to the Cudo platform.

In the case of a parallel job, where a job is split into sub-work packages, the Cudo platform asynchronously collects uploaded work and then re-combines it to produce the final result, which is then verified.

5.6 Validation module

A job type's validation module runs inside the Cudo platform upon a completed workload being uploaded to it. Validation comprises the following methods:

Consensus check Consensus in the Cudo platform describes the process of running a job N times on different suppliers and ensuring that the results are equivalent. This guarantees that none of the suppliers are faking work. The cost to a consumer increases by a factor of N with respect to a job that does not use consensus.

Job type-specific check These are defined per job type. Possible examples include:

Time based If the job has an estimation function and the actual time to complete is significantly lower, the job can be considered failed. If the job was split into equal sized chunks and the job type creator specifies that they expect each chunk to be equally computationally intensive, a sub-job can be considered failed if it finished significantly faster than the others while running on similar hardware.

Hash validation A workload can be designed so that it includes a hash representing the expected output. This hash can then be compared with the provided results, to ensure that the full workload has been completed. This type of validation is relevant for test workloads for example.

Custom validator A custom validator can be written by the developer which can be used to validate their own workloads. These can be set to either validate all or some of the workload results. These validators normally run before the workload is committed as a pre-test to ensure the validations are correct.

Security level Workloads which are not tolerant to incorrect data or tampering can be raised to a higher security level. These workloads can be prevented from running on the lowest level of devices which are anonymous. The primary workload can be run on a high security level such as ISO 27001. For further validation, if the data is non-private data, consensus validation can also be run at a low cost on the lowest level of devices.

6 Job life cycle

The life cycle of a job is the process between the consumer pushing a job to the platform and the completed work being delivered to the consumer. This process is depicted in figure 2. The flow can be described as follows:

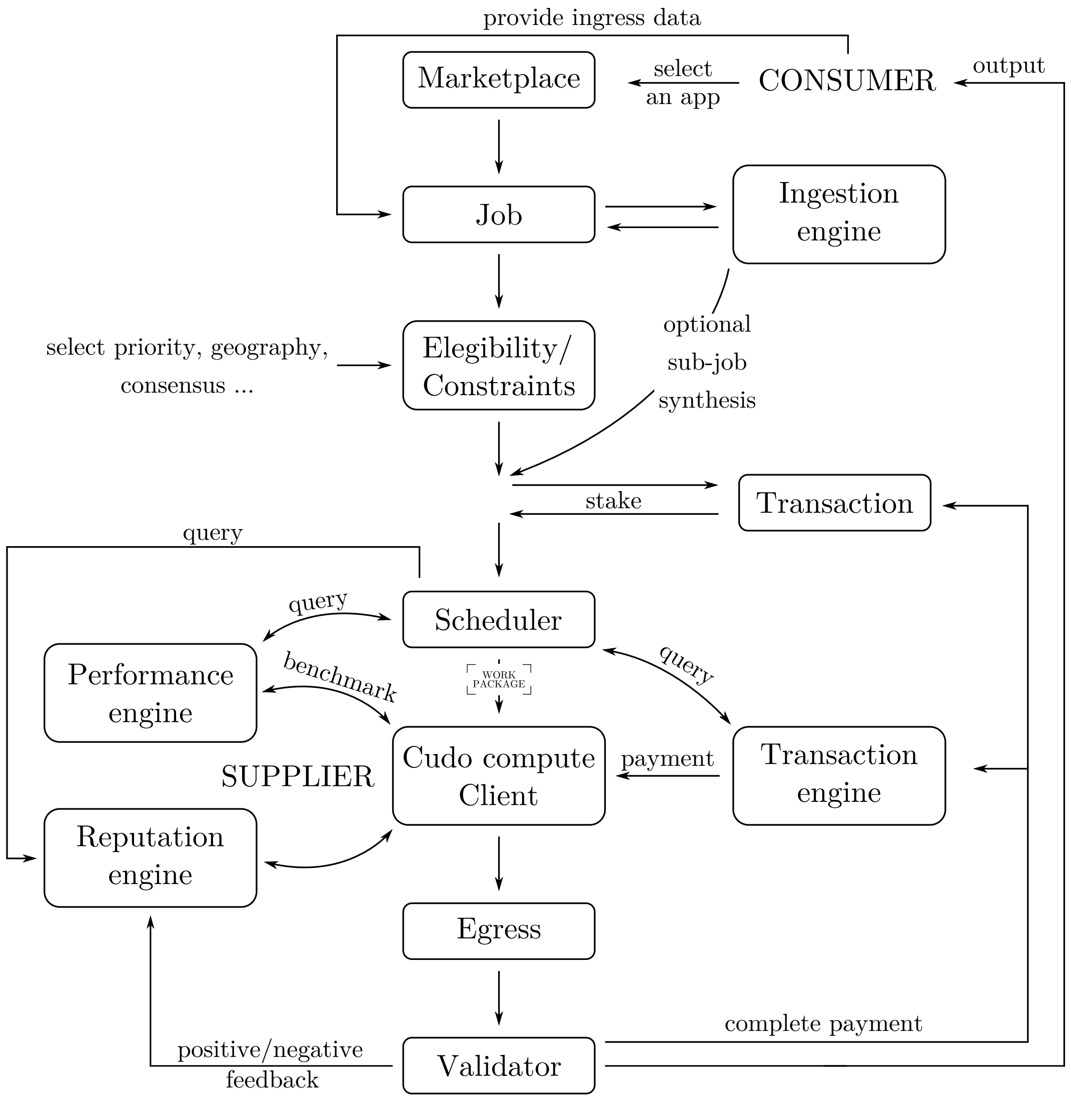

Figure 2: Diagram showing the complete life cycle of a job with a single work package in the Cudo platform. The process comprises the following steps: consumers with a job that needs to be computed go to the marketplace and choose an app to submit their job. The expected price range for this job is estimated by the ingestion engine. This job is matched with the desired constraints of the consumer, and is divided into multiple work packages where possible (the diagram illustrates a single work package for simplicity). Once the packages are created and characterised, the scheduler queries the different engines (reputation, transaction, eligibility constraints and staking) for availability on the network, decides where each package should be sent and submits them to the relevant Cudo compute clients. Once the jobs are returned, they are validated and the reputation of each supplier is updated. If the output is trusted, the consumer receives it and the supplier is paid for the work done.

1. A consumer pushes a job to the platform either via an app or directly via their own process.
2. Ingress data is used by the ingestion engine to create a workload image.
3. As each supplier becomes available, the Cudo platform checks eligibility constraints, reputation, stake and profitability to determine if the supplier can perform the workload.
4. Once suitable suppliers are found, the consumer is committed to pay for the work and the scheduler assigns jobs to the suppliers.
5. Once the work is complete, the completed work is uploaded to Cudo and handled by the egress function.
6. The validator is run and if the work is deemed acceptable, the transaction is finalised and the completed work returned to the consumer.

7 Error handling

Jobs can fail for several reasons, which the Cudo platform catches and handles. The main failure types are:

Timeout If a supplier starts a job and the job is not completed within a certain length of time, then is treated as a timeout. In this case the job is rescheduled with a second supplier and the reputation of the first supplier is adjusted accordingly. The first supplier's stake is not slashed.

Malicious When a supplier submits fake work in order to game the system and Cudo recognises this, the work is treated as malicious. In this case the job is rescheduled with a second supplier and the reputation of the first supplier is adjusted accordingly. Additionally Cudo may slash the stake of the supplier if it is certain that the failure was malicious and not accidental. For more details of the staking/slashing mechanism see section 9.

Consumer complaint In a situation where work has been delivered to the consumer and the consumer complains to Cudo that there is a problem with the output, the job goes to an internal dispute resolution process. Depending on the circumstances, Cudo may handle this by various means including running a consensus check, manually running the job or a technical investigation into the work image. Depending on the result of this process, Cudo may opt to undertake any of the following actions: reschedule the work, penalise the supplier or refund the consumer. The system is designed to minimise the number of such incidents.

8 Security and reputation

Consumers, particularly at enterprise level, cite security as an important factor for not using decentralised computing for their processing needs [21]. The Cudo platform is designed to address security concerns from a number of angles:

- Trusted execution environments (Intel SGX [22] / AMD SEV [23])
- Transport encryption

- Encryption at rest
- Security certified data centers
- Data center verification.

These are offered to consumers as constraints which they can select while ordering a job. Additionally, each supplier has a reputation level which exists as an internal metric in the Cudo platform.

8.1 Life cycle of an SGX/SEV-enabled job

Upon request, the Cudo API starts an SGX or SEV enabled job. While any job can benefit from encryption in transit and at rest when sending data to and from the storage layer, these secure technologies provided by Intel and AMD increase the security of a workload by running them in a secure enclave within the supplier's hardware.

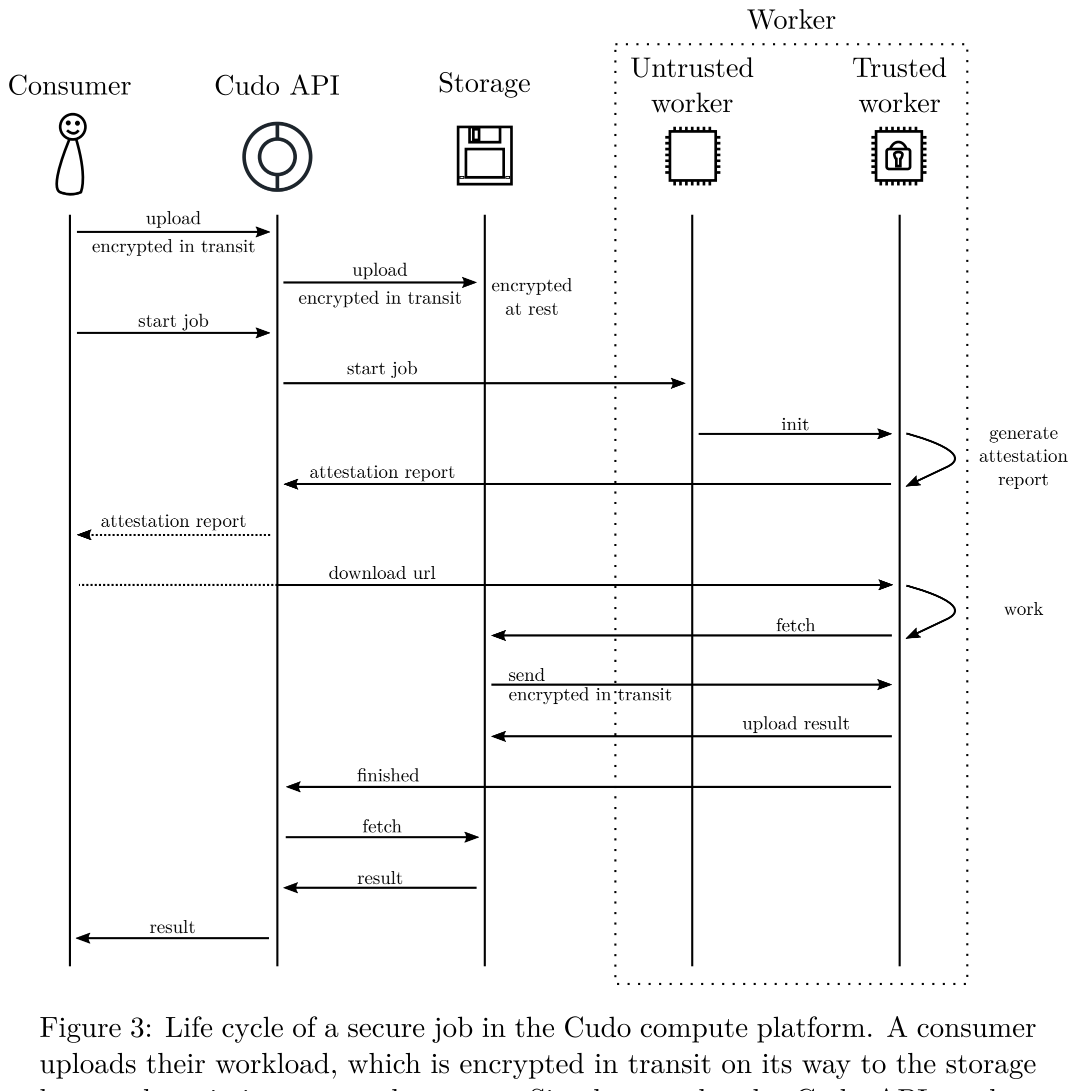

Figure 3: Life cycle of a secure job in the Cudo compute platform. A consumer uploads their workload, which is encrypted in transit on its way to the storage layer, where it is encrypted at rest. Simultaneously, the Cudo API sends a request to the selected worker to start a secure enclave, and does not send the download URL until after the enclave has been verified. Following verification, the data is securely sent from the storage layer to the enclave, which performs the work. The result is then sent back to the consumer.

To ensure that the enclave is created correctly and that no data tempering is detected, these technologies generate attestation reports on the supplier's hardware, as shown in figure 3. We refer to the official documentation for a detailed explanation of SGX [24] and SEV [25], and to the relevant academic literature for ongoing research around these technologies (see, for instance, [26]).

8.2 Reputation system

In order to be part of the network, a supplier requires reputation. The more reputation, the more jobs and priority the machines of that supplier will have in the network. Increased reputation may also imply increased trust in these machines subject to a sufficient period of validated job submissions to the network.

The reputation of the suppliers may be used as a metric by consumers in order to decide which machines will receive their jobs. Thus, building reputation is a method through which suppliers can ensure they receive the most profitable jobs.

Cudo reputation is a tier system ranking different suppliers by popularity and work history. Entry-level consumers are tier 0, with some base reputation. This reputation is increased gradually by successfully accepting and completing jobs, and for successfully passing any tests received.

Once a supplier has accumulated enough reputation, that supplier rises to a higher tier, granting access to higher priority and better paid jobs, *i.e.* jobs that are required to be finished fast.

A given supplier's reputation is calculated as a weighted sum over functions of various different factors. These factors may change for different tiers. For example, for tier 1 and above, the number of computers that the supplier is sharing with the network may be included.

Suppliers who have recently joined the network may not all start at tier 0, as there are other relevant factors considered by the reputation system such as security certificates. Namely, the SAS 70 standard for the US and the UK, or ISO 27001 for the EU, grant suppliers access to a higher tier.

Secure hardware like Intel SGX or AMD SEV also affects reputation scoring. Suppliers with secure hardware are rewarded with increased reputation, subject to successful job completion, and this increased security may also be a constraint which consumers sending jobs can demand for the hardware on which their job will run.

Various secondary factors are also taken into account in the reputation system. While some of these may be used as a separate metric to allow consumers a more precise selection of hardware (or to keep it to a certain location), such as security certificates for storage, there are others applied at the top supplier level that are not specific to a single piece of hardware or location.

For instance, platform incorporates (and takes into account in the reputation system) verification processes via third parties, such as Know Your Customer (KYC) and Anti-Money Laundering (AML).

The total reputation is bounded within each tier and also applied overall, with certain conditions and values required to change tiers. The amount of successful results received by the result validator for a given supplier to advance a level may also change for each tier. Factors involved in reputation calculation include:

- Success rate of test jobs
- Success rate of completed jobs

Tier	Description
-2	Banned from compute. Suppliers can perform mining only with the option to go back to tier -1 after an approval process.
-1	Can be reached by either faking jobs or failing jobs repeatedly. Suppliers receive only mining tasks and compute test jobs.
0	Default entry level tier for suppliers with no security credentials. Suppliers receive compute tasks containing only public data.
1	Suppliers who have been at tier 0 for a certain period of time while successfully completing compute work. Eligible to receive jobs containing private data which is not sensitive.
2	Suppliers which have completed Know Your Customer (KYC).
2.5	Data centers without any certificate.
3	Data centers with an acceptable security certificate.

Table 4: Table showing the different tiers in the Cudo reputation system. Users with no initial staking enter at tier 0 with some base reputation, and can either increase tiers by completing jobs and providing KYC/security certificates or fall to negative tiers if they behave maliciously on the network.

- Amount of time positively contributing to the network
- Number of computers provided
- Number of CUDOS tokens staked in the past.

Negative tiers are included, for instance tier -1, for suppliers who are consistently malicious. Suppliers in these negative tiers are not able to receive compute jobs, but they are still able to mine and return to tier 0 by, for instance, successfully returning test jobs.

Where relevant, a tier -2 class may also be incorporated. In this class, suppliers are permanently restricted to mining cryptocurrency and do not receive compute jobs at all until further notice. See table 4 for a summary of all internal tiers.

9 Cudo's compute platform blockchain side

Creating a fully decentralised blockchain-based service that can compete with existing cloud computing platforms is challenging, as the current generation of blockchain solutions are not viable for the majority of modern workloads. Furthermore, Cudo is initially separating its cloud solution, which has been described up until now, from a purely blockchain part, the CUDOS network, which is explained in detail in section 12. While the way both connect is outlined in section 13, the cloud side of the Cudo platform initially allows for some blockchain integration, to add transparency and security. As a result, Cudo initially employs a hybrid approach in which much of the business logic runs centrally on Cudo servers, with smart contracts used for specific functions related to the CUDOS token. The platform and service have been designed in such a way as to minimise the friction of moving to a full blockchain solution in the future.

Initially there are two relevant smart contracts for the cloud compute platform:

Token contract (ERC20) The CUDOS token is a standard ERC20 token contract on the public Ethereum blockchain.

Staking contract As detailed further in section 11, suppliers may stake tokens for two reasons:

1. In order to qualify for running jobs. The stake acts as a deposit to help prevent abuse on the platform.
2. To obtain a discount in fees from the discount pool (see section 11).

Tokens can be staked by suppliers in two ways:

1. A portion of a suppliers' earnings can be converted to CUDOS and staked automatically by the Cudo platform to qualify them for higher value jobs over time. This is configurable by the supplier.
2. Tokens manually staked by sending CUDOS to the contract by a supplier.

10 Pricing engine

Pricing for Cudo compute jobs is based on mining revenue. Grouping hardware in reasonable ranges, Cudo proposes pricing to suppliers, which is three to four times their mining revenue. Note however that Cudo only proposes a price; suppliers are free to set their own pricing.

Consumers are able to hire the supplier's hardware to run their jobs as Virtual Machines (VMs). Consumers are able to choose between high and low priority VMs, similar to spot instances or preemptible VMs in other cloud supplier platforms. Both are billed at a variable rate according to the prevailing market conditions, unless consumers buy a pricing commitment.

Pricing commitments guarantee fixed-rate pricing for a fixed term. By choosing a commitment term, a predefined machine type and the payment plan consumers obtain a discount on job pricing. The pricing engine creates the hardware ranges described above and recommends a price for each piece of hardware. Note that a single supplier may own multiple pieces of hardware, thus the contract is for the hardware piece rather than the supplier's entire array of devices.

11 Cudo token: CUDOS

The CUDOS token has utility in the Cudo distributed compute platform in the following ways:

1. Staking to receive a discount in fees
2. Staking to qualify for jobs.

In the future, more uses for the token will be explored for the cloud offering, including as a medium of exchange (using a scalable low-gas solution such as payment channels [27]) and further benefits for consumers such as inexpensive cross-border remittance.

Notice that the CUDOS token has further utility as the backbone that powers the CUDOS network, which is discussed in section 12. The details about the added utility and functionality of the token for the CUDOS network, including delegated staking, are deferred until that section, and this section is focused on the use of the token within the Cudo cloud computing platform.

11.1 Staking to receive a discount in fees

The CUDOS token utilises a discount token model based on a staking mechanism, where a fixed percentage of the total fee revenue generated on the Cudo network is distributed to suppliers. This discount is based on the fees paid by each supplier on the network and the amount of CUDOS each supplier has staked.

In order to receive higher discounts, suppliers can:

1. Stake the relevant number of tokens required, until the discount equals the maximum discount available (a percentage of the fees), and
2. Generate as much revenue as possible on the platform so as to increase the maximum discount available.

This model is based on a staking mechanism in the Sweetbridge protocol [28].

The discount received by suppliers is obtained by dividing the amount of capital available to be distributed in discounts in a given staking round (discount pool) by the number of tokens that participate in the round (token supply) and multiplied by the number of tokens staked. To be more precise, consider the following definitions,

DP: Discount Pool. A portion of Cudo Ventures' fee revenue contributed to discounts in a given period of time (every week, month, etc).

R: Discount Rate. A percentage of revenues that Cudo Ventures is contributing to DP, and the maximum percentage discount suppliers can get. This can be a fixed value, say 50%, or a variable determined by a formula.

Cr: Cudo Ventures' fee revenues in a given period.

DpT: Discount per Token. A value of discount that suppliers can enjoy for each token they hold.

N: Multiplier. This can be set by Cudo to increase the discount per token in case the return on staking is deemed too low. It will always be greater than or equal to one.

TS: Token Supply. The number of tokens that participate in the distribution of discounts. This can be a sum total of all the tokens staked for discounts by suppliers or the total circulating supply of tokens.

M_iF: Fees paid by an individual supplier i.

TM_i: Token supply of an individual supplier i staked for discounts in the period.

M_iD: Discount granted to an individual supplier i.

In order to obtain the discount a supplier receives, the discount pool needs to be calculated first,

$$DP = R \cdot Cr. \tag{1}$$

The discount per token can then be obtained as

$$DpT = N\frac{DP}{TS}, \tag{2}$$

and thus the discount supplier i receives is

$$M_iD = \min\left(DpT \cdot TM_i, M_iF \cdot R\right). \tag{3}$$

Example This scenario assumes \$100M in Cudo revenue (30% of the revenues generated on the platform), with a 50M locked token supply, a discount rate of 50% ($R = 0.5$) and a multiplier $N = 1$. A supplier (Alice) contributes to the network in a single month with \$100 in revenue by generating compute power as she leaves her computer on every night.

With the assumptions above, the discount pool (1) is

$$DP = \$100\text{M} \cdot 50\% = \$50\text{M} \tag{4}$$

and the discount per token (2) is

$$DpT = \frac{\$50\text{M}}{50\text{M tokens}} = \$1 \text{ per token.} \tag{5}$$

If Alice sells \$100 worth of compute, her payable fees are \$30 (\$100 · 0.3) in a given month. If she holds between 0 and 15 tokens, the discount in the fees is between 0 and 50% respectively, given that the discount per token is \$1 (the discount is capped at \$15 as the discount rate is 50% and the fees paid are \$30, see formula (3)). By staking, Alice saves up to \$15 in fees that month.

Suppose now that Alice holds more than 15 tokens. Since her discount is capped at \$15, or 50% of her fees (15% of the total money earned), she still receives the same discount as if she had 15 tokens. In this case, assuming she has for instance 10 extra tokens than the ones necessary to obtain the full discount (25 tokens in total), she has several options to choose from:

1. Keep the tokens in her wallet.
2. Sell 10 tokens on the market at the current price.
3. Allocate more computing power to the network. Putting more computing power online effectively increases the maximum discount Alice can get. This is because as the contribution to the network increases, so does the discount a supplier can get.
4. Exchange the extra tokens for additional services. For example, gamers could choose to buy Steam vouchers and other supported assets with CUDOS natively on the platform.

In any case, the discount that Alice receives is given by equation (3), and so in this case it is

$$M_iD = \min\left(\text{number of tokens staked} \cdot \$1, \$15\right). \tag{6}$$

11.2 Staking to qualify for jobs

The CUDOS token utilises a staking/slashing mechanism, where in order to qualify for a given job the supplier may need to have staked CUDOS tokens. This model serves to discourage malicious behaviour such as submitting fake work in order to game the system.

The slashing mechanism is executed when the supplier acts maliciously, but not when a job fails due to non-malicious reasons, such as a timeout (which may be due to a power cut or disconnected network, for example). The slashing mechanism is enforced automatically under the following conditions:

1. The workload is completed and uploaded to the Cudo platform.
2. The Cudo platform executes the job type's validation module and either
   (a) A consensus check fails, or
   (b) A job-type specific validation check fails which proves malicious behaviour, for example:
      i. The workload was completed with significantly less work than the estimation.
      ii. The workload was completed with significantly less work than the other suppliers in the consensus check.
      iii. An algorithmic check to verify the output data fails.

Additionally, Cudo may opt to slash stake in extreme, manually verified situations such as when a supplier is proved to be using hacked devices. If the job fails and the slashing mechanism is executed, the staking smart contract retains an amount of staked tokens equivalent to the total required for that job.

11.3 Staking mechanism

CUDOS tokens are staked in the staking contract. Optionally, a supplier may opt to automatically stake a portion of their earnings as CUDOS tokens, such that they qualify for a larger discount. Over time, they may stake sufficient tokens to qualify for higher-value jobs that their increasing reputation would allow for. This process is transparent to the supplier and only a small proportion of their earnings need to be staked in order to significantly increase the value of their jobs over time.

No initial staking is required to participate in Cudo's compute platform – the initial amount of CUDOS needed to accept a user's first compute jobs can be earned through mining jobs upon installing the Cudo platform.

12 CUDOS network

The previous section has introduced the CUDOS token, and has focused on its utility for the compute workloads on the Cudo platform. This section introduces and details the CUDOS network, a separate application aimed at providing a layer 2 network for blockchains, allowing the off-loading of compute and data to overcome scalability issues. See section 13 for a description of how the Cudo platform and the CUDOS network are related.

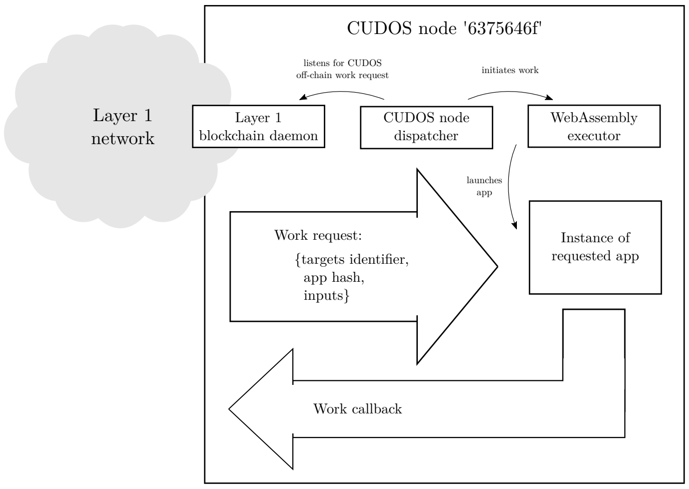
CUDOS node '6375646f'
listens for CUDOS off-chain work request
initiates work
Layer 1 network
Layer 1 blockchain daemon
CUDOS node dispatcher
WebAssembly executor
launches app
Work request: {targets identifier, app hash, inputs}
Instance of requested app
Work callback

Figure 5: Sketch of a request triggering a CUDOS workload. A layer 1 smart contracts requests some work through the CUDOS smart contract, triggering an event. The CUDOS nodes listen to the contract's events, and execute the requested work when they are chosen. After fetching the data and running the workload, the result is returned to the CUDOS smart contract.

12.1 Overview

Smart contracts written in a layer 1 network can invoke the CUDOS smart contract, deployed in that same layer 1 network, in order to request work to be computed off-chain or to access external data. The off-chain computation is done in the CUDOS nodes, which need to stake 2,000,000 CUDOS in order to be eligible. These nodes are constantly listening to events in the CUDOS smart contract, to see when a new request for a compute job is created. This request includes three main components:

- A targets identifier
- An app hash identifier
- Any inputs needed for that workload.

The targets identifier refers to some piece of data that is used by the CUDOS nodes to decide when they need to run a job. This can either be a set of hashes identifying each individual node separately, or some unique identifier which the nodes use to decide whether they need to run the job. More details on this will be given in subsection 12.2.1, and a full overview of the process is shown in figure 5.

The app hash is used in order to decide which code the CUDOS nodes need to run. That hash can either refer to an existing app from the CUDOS dapp marketplace, or it can point to an external storage address where some code written by the requester has been previously uploaded. Last, the request to the CUDOS contract may also include a list of inputs to be used by the code or app that will run in the nodes. These inputs can be passed directly in the request, if they are just short numbers or characters, but will typically be addresses to an external storage solution where the input data has been uploaded beforehand. In the next subsection more details are given about storage.

Once a node has heard an event and has decided that it should run the work, that triggers its WebAssembly executor part. Note that listening to the blockchain is already an off-chain process, so this process is not constrained by the blockchain's limitations anymore. Once execution starts, the node will fetch the passed inputs, and will send the relevant API requests to the marketplace and the app in order to run the compute workload.

After the result is obtained in each CUDOS node, a consensus check might be needed in order to return a unique result to the original requesting smart contract on the layer 1 blockchain. See subsection 12.2.2 for an explanation of how consensus can be reached in the CUDOS network. Once that unique result (or address storing the result) has been decided and sent to the CUDOS smart contract, the original smart contract can fetch it.

12.2 CUDOS nodes

As has just been described, CUDOS nodes are responsible for listening to the CUDOS smart contracts and running any workloads when relevant. Initially, all CUDOS nodes run SEV-ready hardware. AMD's SEV technology allows the memory contents of a VM to be transparently encrypted with a key unique to the guest VM. This adds an extra layer of security and protection to the network, for both the node and the job requester.

In addition to the workloads requested from a blockchain's smart contract, CUDOS nodes have constant utilisation. All the hardware not working on the CUDOS network to provide compute or data to a blockchain runs the Cudo software, in order to ensure full utilisation and monetisation. As such, part of the technology behind the CUDOS nodes is shared with that of Cudo's workers, in order to enhance the integration between both layers.

12.2.1 Identifying nodes

As mentioned above, initially the CUDOS smart contracts will require the requester to select which validator nodes need to run the job. However, other selection methods are currently being investigated in order to increase automation and reduce gas costs, in the case of layer 1 networks like Ethereum. For example, following [29], one option would be to deterministically elect workers based on a randomly generated job ID.

12.2.2 Consensus

CUDOS nodes create an off-chain peer-to-peer network, in order to cross-check results, share data and run validations. This peer-to-peer network may be used for several things, including the node identification mentioned above. This network may also be responsible for reaching consensus on specific workloads. Namely, consensus in the CUDOS network may run in two ways:

- On-chain through a smart contract
- Off-chain in the peer-to-peer validator network.

While the result of some workloads may be simple enough to cheaply run consensus on-chain, more complex workloads will require custom code for the validation. This code will either be chosen from the dapp marketplace, or will need to be written by the requesting blockchain developer.

13 Cudo and CUDOS connection

Sections 2 to 10 in this paper have focused on the Cudo platform, while sections 11 and 12 introduced the CUDOS token and the CUDOS network. As discussed, the CUDOS

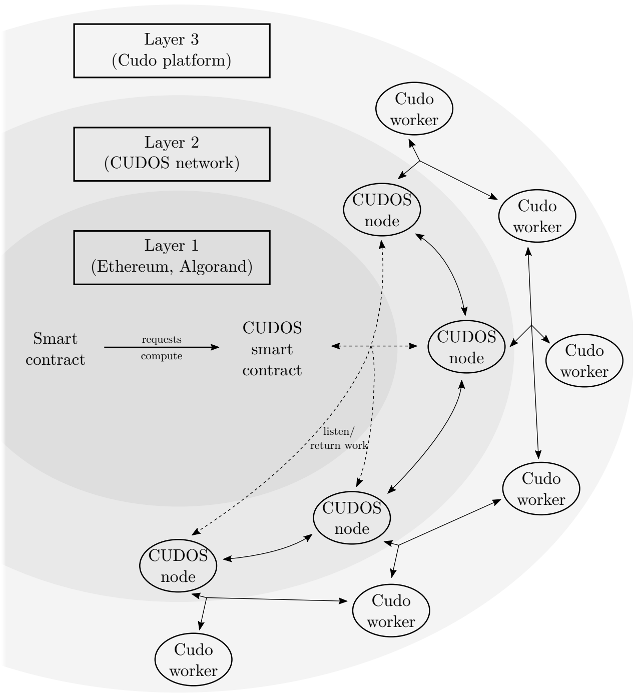

Figure 6: High-level overview of the integration of the CUDOS network and the Cudo platform with layer 1 blockchains. Smart contracts from a layer 1 in need of compute or external data may request the layer 2, CUDOS network services through the CUDOS smart contracts. In turn, if the job has been required so, the CUDOS nodes may connect to Cudo workers, in order to use more computing power or specialised hardware, such as high-end GPUs. Thus, the Cudo platform is effectively a layer 3, providing extra compute to the layer 2.

network is a layer 2 solution for blockchains which require extra compute or external data. Blockchains like Ethereum have very high gas costs, which make impractical running complex workloads on them. Other blockchains like Algorand need a layer 2 solution in order to add extra functionality to the platform, through a Turing-complete network that also provides external data.

While the CUDOS network provides all this required functionality, Cudo Ventures is going one step further, by seamlessly integrating this technology with the Cudo platform. As seen throughout this document, the Cudo platform provides cloud-like services, focusing on low costs, low latency and a high degree of personalisation. Hence, just like the CUDOS network is a layer added on top of blockchains to provide extra compute capacity, the Cudo platform can be seen as an extra layer on top of the CUDOS network, to provide even more on-demand compute capacity. This added capacity might be needed in order to select different types of hardware, or to request jobs that demand more resources than the CUDOS network can directly absorb. Figure 6 pictures how the layers are organised.

As such, the CUDOS smart contracts will provide access to the CUDOS layer 2 and the Cudo layer 3, in order to run any kind of workload on any kind of supported hardware, for as long as needed. This integration takes blockchains a step further into mass-adoption, as there will be no limit in the kind of workloads that can be requested through an on-chain smart contract.

14 Summary

Cudo is building two new products: a distributed computing platform that expands its current cryptocurrency mining platform and the CUDOS network, a layer 2 solution that provides extra compute and external data to blockchains. Currently Cudo is a cryptocurrency mining application that mines the most profitable coin for each hardware, and pays out in the user's choice of the most popular cryptocurrencies. This application is used to ensure that all Cudo workers and CUDOS nodes have constant utilisation and remuneration, even when market demand for compute jobs goes down.

In the CUDOS network, nodes get rewarded by their contribution to the network, which is twofold: running workloads requested from the CUDOS smart contracts and maintaining the ecosystem by being an active node. Similarly, all users can support the nodes, by delegating their stake to their favourite CUDOS node. This allows for an active and extensive network that powers and pushes forward the boundaries of the current blockchain technology and DeFi, by enabling extra functionalities which are only possible with compute-intensive work and external data.

On the Cudo platform side, the software benchmarks all the hardware connected to the platform, and records the real world capacity in order to distribute workloads efficiently. This benchmarking includes GPU, CPU, memory, storage and networking capabilities, as the platform will support workloads incorporating all of these resources in the future. Based on these capabilities, the Cudo platform determines the most profitable task for each hardware, and switches automatically to cryptocurrency mining when no distributed computing workloads are available or profitable. Jobs can be sent to Cudo compute through job-specific applications, discovered through a marketplace where developers are rewarded for creating apps that facilitate this task. Consumers are also free to send their own custom compute workloads. In addition, Cudo compute sends test workloads which are part of the benchmarking process.

Internally, the Cudo platform has different engines which accept, prepare, schedule, distribute, validate and return the different workloads, as well as manage all payments. The distribution of jobs is partly based on a reputation model, which determines the levels o security and of confidence in individual hardware suppliers. This reputation model is a tiered system which restricts malicious users to cryptocurrency mining only.

Suppliers can be entitled to discounts on the fees they pay on the platform, provided they stake enough CUDOS, an ERC20 token. Staking not only leads to discounts, but acts as a deposit to prevent abuse. The minimum stake needed to participate in the compute network may be obtained by completing mining jobs.

While Cudo compute provides recommended pricing to hire computing power, suppliers are free to change and tune the pricing according to different metrics. For instance, they can set the price at a certain percentage above mining revenue, or as a percentage above the expected electricity costs. All these options come with a user-friendly interface, in line with all of Cudo's existing applications. Hardware suppliers are also free to turn off mining altogether, and only receive distributed computing workloads.

References

[1] Synergy Research Group. Hyperscale data center count jumps to 430; another 132 in the pipeline. `srgresearch.com`.

[2] Becky Peterson. Companies waste $62 billion on the cloud by paying for capacity they don't need, according to a report. `businessinsider.com`.

[3] Roddy Scheer and Doug Moss. How clean is the energy used by tech companies for cloud computing? `scientificamerican.com`.

[4] Louis Columbus. Public cloud soaring to $331b by 2022 according to gartner. `forbes.com`.

[5] Susan Platt. Metamorphosis of an industry, part two: Moore's law and dennard scaling. `micron.com`.

[6] Richard Bradley. Blockchain explained... in under 100 words. `deloitte.com/blockchain-explained`.

[7] Niall McCarthy. Bitcoin devours more electricity than switzerland. `forbes.com/bitcoin-devours-more-electricity-than-switzerland-infographic`.

[8] Ethereum 2.0. `ethereum.org/eth2`.

[9] Kirsten Richard. Polkadot and cosmos. `wiki.polkadot.network/learn-comparisons-cosmos`.

[10] Statista. Shipment forecast of laptops, desktop pcs and tablets worldwide from 2010 to 2023 (in million units). `statista.com`.

[11] Statista. Number of smartphone users worldwide from 2014 to 2020 (in billions). `statista.com`.

[12] Statista. Number of tablet users worldwide from 2013 to 2021 (in billions). `statista.com`.

[13] Louis-Benoit Desroches *et. al.* Computer usage and national energy consumption:results from a field-metering study. `eta.lbl.gov`.

[14] Airbnb. `airbnb.co.uk`.

[15] Blablacar. `blablacar.co.uk`.

[16] Worldwide LHC computing grid. `wlcg.web.cern.ch`.

[17] LHC@home. `lhcathome.web.cern.ch`.

[18] SETI@home. `setiathome.berkeley.edu`.

[19] Cudo Ventures. CUDOS. Next Generation Cloud. Whitepaper. `cudos.org/whitepaper`.

[20] Kubernetes home page. `kubernetes.io`.

[21] André Müller, André Ludwig, and Bogdan Franczyk. Data security in decentralized cloud systems – system comparison, requirements analysis and organizational levels. *Journal of Cloud Computing*, 6, 12 2017.

[22] Intel software guard extensions. `software.intel.com`.

[23] Amd secure encrypted virtualization. `developer.amd.com`.

[24] Intel® 64 and ia-32 architectures software developer's manual: 3d. `IntelSoftwareDeveloper'sManual`.

[25] Amd64 architecture programmer's manual volume 2: System programming. `AMDtechdoc`.

[26] Dayeol Lee, Dongha Jung, Ian T. Fang, Chia-Che Tsai, and Raluca Ada Popa. An off-chip attack on hardware enclaves via the memory bus, 2019.

[27] Jim McDonald. Introduction to ethereum payment channels. `medium.com`.

[28] J. Scott Nelson, David Henderson, Glenn Jones, Micha Roon, Michael Zargham, Aleksandr Bulkin, Jake Brukhman, and Kenny Rowe. Sweetbridge: a blockchain-based protocol stack for global commerce and supply chains. *Sweetbridge*, 2018.

[29] Daniel E. Eisenbud, Cheng Yi, Carlo Contavalli, Cody Smith, Roman Kononov, Eric Mann-Hielscher, Ardas Cilingiroglu, Bin Cheyney, Wentao Shang, and Jinnah Dylan Hosein. Maglev: A Fast and Reliable Software Network Load Balancer. *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*, 2016.