The Content Generation Network

Technical White Paper 2.0.0.3

A cutting-edge Infrastructure-as-a-Service that reduces current costs and lets you earn money with unused CPU/GPU power. This is the missing DServices platform for DApps. Pay as you go with CPUcoin.

TABLE OF CONTENTS

ABSTRACT

CPUcoin is developing a first-of-its-kind Content Generation Network (CGN). A unique, flexible, and scalable Infrastructure-as-a-Service offering targeted at the needs of B-to-B. A distributed system for delivering services powering DApps (Decentralized Applications) – both consumer AND enterprise-class. The CGN is a general-purpose ecosystem, uniquely architected to excel in providing services that meet demand in real-time, processing requests to perform the specialized work needed to generate and deliver content for DApps on a global scale. CGN ecosystem work is transacted exclusively using our new utility token, the CPUcoin.

What makes the CGN unique is a design capitalizing on vast amount of unused CPU power available worldwide, enabling a new sharing economy for CPU resources. The CGN amasses a pool of computational power of inconsistent capability and availability and reshapes it into a massive, powerful and coherent infrastructure, one which provides overall consistent availability with excellent performance. Due to the relatively low value associated with idle CPU power, we can offer a lower cost means of delivering services. We have coined the term DService (Decentralized Server Application) to mean any service built or adapted to run using the CGN.

The sharing economy for CPU power can be likened to a sharing economy for rental space. If you plan to travel away from home for a while, you may choose to use a service such as Air B&B to rent out rooms in your home to short-term occupants while you are away. The occupant, pleased to have access to a wide range of accommodations, will pay for the privilege. Whether you rent or own your home, this extra income can be used to offset your living costs. Likewise, if you need a place to stay while you travel, you can use the same service as a consumer to quickly find short-term lodging while you travel, paying only for what you use. Sharing economies improve efficiencies for society by making better use of resources that already exist.

Our proposed service will work in much the same way, with housing as a metaphor for computer power. Server software also needs a place to reside, and like housing space being both provided and consumed from the available unused housing pool, server space can also be provided and consumed from the available pool of unused computational capacity. We like to say our proposed service will be the Air B&B of CPU power.

We have focused our efforts on providing an infrastructure platform anyone can connect to, which means new DApps can make use of existing DServices already powered by our CGN. It will also be possible to create new DServices, deploy them to the CGN, and use them to power your own DApps.

Importantly, to validate and prove the viability of the design and robustness of our platform, we have initially developed and delivered one much-needed DService, and are readying it to power an actual enterprise application by both moving it to the cloud and upgrading its overall capacity and scalability.

We started with an existing premises enterprise image-processing application called “MediaGen” and ported a large subset of its proven media generation technology stack, thereby creating the first DService. This DService, called “MediaGen” provides on-demand, focused, scalable and secure visual content processing. This will bring far greater power and scale to dApps that use MediaGen, such as Publisher, which have already been put to work in some of the largest companies in the world for many years.

We have already released a functioning proof-of-concept on our CGN TestNet and are now working on a production system capable of supporting even more real-world enterprise applications.

Unlike other similar projects that seem to be building similar functionality from scratch, our approach is to deliver a known and proven enterprise-class application that will be put to immediate, real-world use serving a massive Total Addressable Market at the time of the project’s release.

A cross-licensing deal shall be consummated as part of the pre-IEO close, providing immediate real-world value to purchasers of the CPUcoin utility token.

To support success of the project, we have taken a pragmatic approach in requirements generation to ensure that the infrastructure can be built rapidly in well-defined stages, while focusing on key enterprise use cases.

TERMINOLOGY OVERVIEW

Before getting into a discussion about the problem and our proposed solution, we would like to familiarize you, the reader, with terminology we will be using throughout this whitepaper. For clarity, we have grouped terminology definitions by scope.

DEVELOPERS OF TECHNOLOGIES

CPUcoin	A Cayman foundation company registered under the name DServices Limited and trading as CPUcoin and/or CPUcoin, engaged in the business of developing and operating the Content Generation Network (CGN). This entity will design, build, operate and maintain the CGN. All financial matters associated with use of the CGN will be transacted exclusively through use of CPUcoin, a new cryptocurrency utility token created for this purpose. CPUcoin will create the supply of CPUcoins to operate the service ecosystem by issuing tokens through an IEO. Proceeds of the IEO will be used to complete development of the CGN, to develop the first DService, license certain technology and source code from Equilibrium and for future development initiatives.
Equilibrium	A US-based company engaged in the business of developing and providing premises and cloud-based media processing software. Equilibrium has an established product family and has spent years building product demand with a significant established user base. Equilibrium wishes to expand the reach and capacity of its product line by being among the first to make use of the CGN, DApps and DServices to offer global, highly scalable image processing services in compelling new ways.

DISTRIBUTED APPLICATIONS AND COMPONENTS

DApp	In the client-server model, this is the client component. It contains all user-facing (application code, software system or script, usually user-facing but not required to be, that makes use of the CGN server infrastructure to issue requests to a DService, the server component built to handle the specific work needed by the client.
DService	In the client-server model, this is the server component. It is a unit of server software developed to run in the CGN environment. The DService executes requests made to it in accordance with a well-defined API that gives access to the dApp or family of dApps to perform the specialized, compute-intensive work it needs.
MediaGen, or MediaGen DService	The first DService, licensed from Equilibrium and built on proven technology. MediaGen DService will leverage the CGN to provide the services layer needed to power the MediaGen family of DApps and client applications. The MediaGen DService operates within the CGN ecosystem and provided the services for performing general purpose, near real-time request handling for image, animation, font, audio/video, dynamic processing and rendering operations.

ECOSYSTEM COMPONENTS

Worker Node (Miner Client)	A server or consumer device connected to the network to do work, offering CPU/GPU, threads, RAM, bandwidth and storage to the CGN as available resources.
CGN Work Broker	A service that matches incoming work requests to a specific, available best-fit worker node from a pool of available resources to perform a specific work unit.

Distributed Services Infrastructure

Regional CGN Node	The regional CGN node is a highly scalable supercomputer located in a specific region, configured as a type of computational infrastructure known as a distributed server. The CGN operates this server and ensures its continued availability to DApps, handling all requests made to DServices, the server code that performs compute-intensive work needed by DApps. The CGN node is built to operate at high throughput and automatically scales based on demand. It is capable of efficiently handling many concurrent requests in near real-time. The CGN node is connected to a large pool of contributor-operated worker nodes, which are computers in the homes, business and data centers of a specific geographic area. These computational resources, being a blend of widely varying capability and operating in a wide range of environments, will individually be inherently unreliable, with at least sometimes availability, and having variable but measurable performance characteristics such as maximum computational power capacity, current and average CPU availability, network latency of its internet connection, as well as other important attributes. The CGN Node performs the job of reorganizing this large pool of varying quality computational resources into a coherent system that has superior characteristics to the individual worker nodes by continually making strategic use of all immediately available resources. It can therefore offer a large amount of computational capacity with overall high availability and scalability.
Content Generation Network (CGN)	The CGN is the entire system of all regional CGN nodes worldwide, operating together as a whole. Each DApp uses a provided endpoint to discover one or more best match regional CGN nodes to connect to for operating its DServices, naturally providing edge support with low latency. For powering DApps, the CGN offers a cost-effective infrastructure solution, and provides a compelling alternative to the data centers and cloud computing services typically used for hosting delivery of similar kinds of services.

Payment-Related Terms

Blockchain-powered	The CGN uses the blockchain to transact the provision and consumption of services in bulk, using a utility token called the CPUcoin. The CGN ledgers all transactions, tracks all usage, and regularly collects total amounts for all incoming use fees, and disburses the sum of all outgoing payments to beneficiaries, according to pre-arranged agreements.
Off-chain microtransaction support	A per-CGN node blockchain connected microtransaction system will collect and record all transaction detail information generated by that CGN node at very high rates of throughput, which may number tens of thousands of transactions per second. Use fees and disbursement amounts are recorded in real time for every DService request by the CGN. Custom DService-specific metadata can also be stored with each microtransaction.
Hybrid solution	Despite storing microtransaction data off-chain, none of the essential qualities of the blockchain will be lost. The system will offer the transparency, trusted immutability, and proof of correctness in all ledgering, as blockchain itself offers.
CPUcoin	An ERC-20 compatible utility token, to be issued by our company of the same name, serving as the currency of exchange for all CGN-related transactions.
Wallet	Refers to an Ethereum wallet. Any Ethereum wallet may hold CPUcoin tokens.

Ecosystem Participants

CPUcoin ("we")	Entity responsible for developing, delivering and operating the project.
CGN Operator Entity	Entity responsible for operating a given regional CGN. This entity (generally CPUcoin) will receive a royalty fee for all use of that regional CGN, paid in CPUcoin.
Developer Entity	Entity that developed a DService for use with compatible DApps, client applications or other services. This entity will receive royalty fees paid in Dyncoin.
Equilibrium	Entity that developed the MediaGen DApp and the MediaGen DService. This entity will receive a royalty fee for use of the MediaGen DService, paid in CPUcoin.
Enterprise Account, Enterprise Account Entity, or Business	Any business entity (or individual) that has entered into an agreement to directly or indirectly consume DServices, typically through, but not limited to, the use of CGN-enabled DApps or client applications, paying for all consumption of those services using the CPUcoin utility token. Business enterprise accounts generally will have multiple end users, typically employees, contractors and designees of the business who are authorized to use a DApp or client application on behalf of the business. The business takes responsibility for payment for all services consumed by its End Users.
End User	Any individual user of a CGN-enabled DApp. For Enterprise accounts, each end user is an employee, contractor or designee of the corporation authorized by the business to use its DApp.
Contributing Entity or "CGN Miner"	An Individual or corporate entity who has chosen to operate a Worker Node to offer spare or otherwise available computational and bandwidth resources to the CGN to do the computational work of running DServices, i.e. "mining", in exchange for the opportunity to earn CPUcoin.
Charitable Beneficiary	A charitable cause that in some cases may be given as a donation a small share of the proceeds of each transaction associated with DServices operating in the CGN.
Beneficiary	Any entity type receiving a portion of the proceeds collected for completing work: • The CGN operator entity • The developer entity of specific DServices • The contributing entity who hosts DServices • Any associated charitable beneficiaries.

General

The Service	Refers to the CGN itself as a service, the system which manages and delivers all DService offerings in the form of Infrastructure-as-a-Service (IaaS).
A Service	Refers to server code that listens to the internet, waiting for an incoming request to perform work. In our design, we have coined the term "DService" for this: Any Web-based service that has been adapted to work within the CGN ecosystem.
Request	An HTTP request made through the internet to a DService, asking it to do work.
Response	An HTTP response to a request, made through the internet. This is the reply to a request to a service, containing the results of the work performed by the DService.

SECURITY

Trusted Computing	Trusted Computing is technology that guarantees a computer will consistently behave in expected ways, with those behaviors being enforced by computer hardware and software together.
Trusted Execution Environment (TEE)	A software execution environment that can be trusted to only execute known good code that has been signed by an appropriate authority. It makes Trusted Computing a possibility.
Hardware Security Module (HSM), Trusted Platform Module (TPM)	A hardware device, usually a component physically built into the system hardware itself, responsible for safeguarding and managing digital keys for strong authentication and providing crypto processing. This device makes it possible for a Trusted Execution Environment to prove that its contents are indeed authentic and unaltered.
Remote Attestation	A process through which a computing system can provide confidence in its own integrity to a relying party, permitting it to decide to trust the computing system.
Byzantine Actor	A malicious participant in the system who tries to thwart the integrity of the network or cheat the system by intentionally submitting fake work or by otherwise abusing the protocol.

Basic Background

Audience

This white paper targets an audience of readers with a range of technical knowledge. We have included this section as a basic technical background reference, offering readers an opportunity to understand enough basic material to be able to fully appreciate the remainder of the white paper.

If you have an advanced understanding of answers to the questions listed below, you can skip this section.

- Why is a client-server model so commonly used for building and delivering applications, and what are the benefits are of doing so?
- Why does delivering a services-based application impose a long-term commitment to operating server infrastructure?
- What is scalability as it applies to client-server applications, and what are the consequences of not actively managing it?
- Why does the use of fixed server infrastructure require capacity planning, and what can go wrong if a company fails to do it well?

To skip this section, please continue reading at the next section, Problem Background.

The Client-server Model

Many companies provide a wide array of Web-based applications, often to a user base as a pay-for-use service, but sometimes for operation of internal systems. These applications are delivered in multiple ways, whether it be through a browser, an application that installs onto the user's PC, or as an app that can be downloaded and run on mobile devices, tablets, etc.

The client-server model is a design pattern commonly used today when developing Web-based applications. Use of this pattern is regarded as a best practice approach by software engineering teams that makes the software easier to develop and allows for maximizing available power and capacity.

However, a downside associated with this pattern is the imposition of operations-related responsibilities on companies who wish to deliver a software product and ensure it always operates reliably regardless of demand, and at a reasonable cost.

This paper focuses on ways on to dramatically improve the operational aspects of applications that are constructed and delivered in the form of a client-server system or server-side scalable services, by using an innovative approach to provide the computational capacity required for operating all the server components.

As an introduction, the next section will explain and illustrate the relationship between client and server in this model.

A CLIENT-SERVER SYSTEM

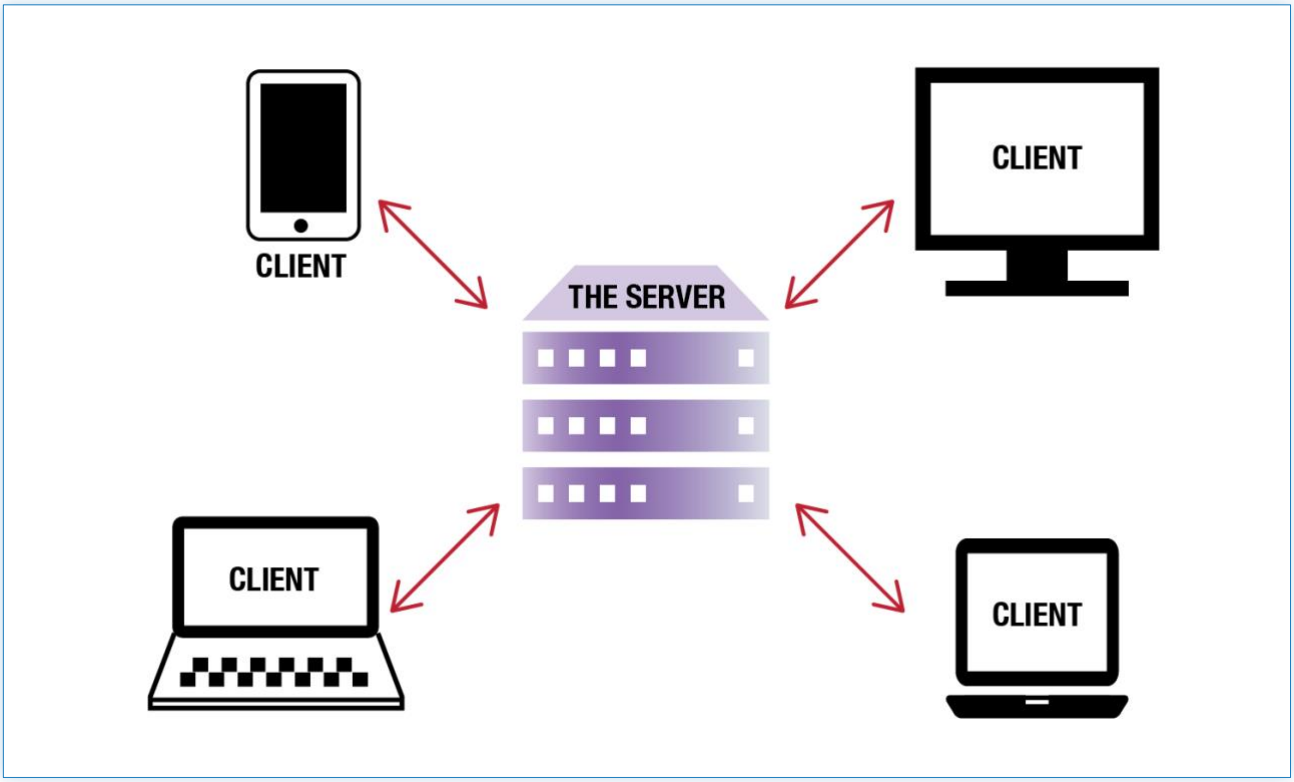


FIGURE 1

THE CLIENT

The client component of a Web application is any piece of software that has been built to interact with the server component, and which is connected to the internet so that it can always reach the server component.

USER-FACING CLIENTS

A user-facing client is simply client software a user interacts with. It provides a user interface, handles user workflows and the like. This type of client software can run in any environment where a user interface may be presented such as in a Web browser, on a phone or tablet, or it can be software installed on a PC. It could also be software incorporated into consumer electronics devices.

SERVICES AS CLIENTS

Other, non-user-facing forms of clients can also exist. The only essential quality of client software is that it is software written such that it requires use of a specific server component, making it a client of that server's service. In complex Web applications with or without a user-facing component, services themselves may be layered. Layering often results in one service playing the role of client with respect to yet another service.

THE SERVER

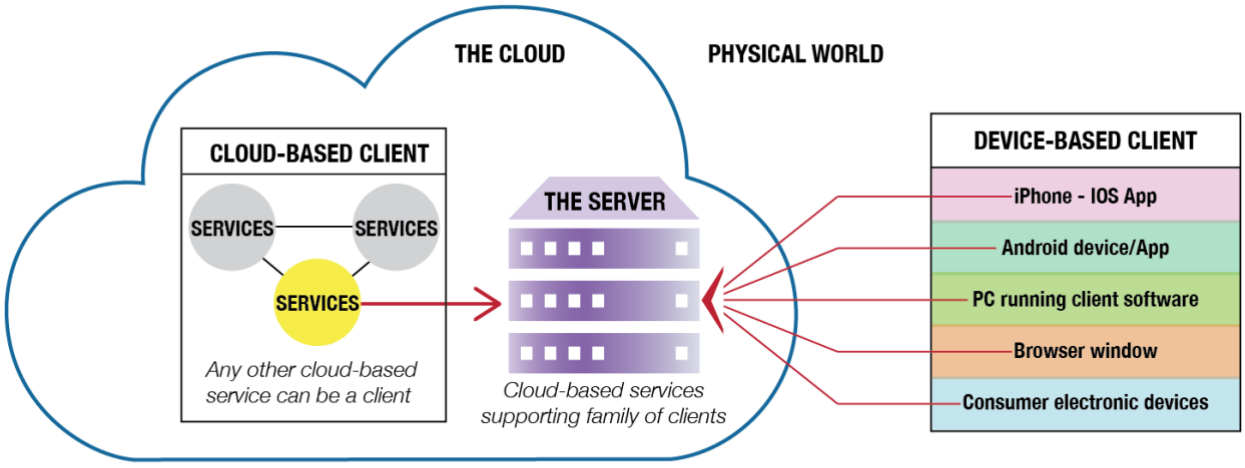
The server is an always-available computational resource that reliably runs the services software component required by the client, continuously processing requests made by all clients of the service at any time. The service layer isolates the client implementation from the services implementation by defining and making use of a clear, well-defined interface that exposes services in a way that is independent of the particular use-case of any of the clients. Any kind of client will be able to make use of the service.

Typically, a server receives and processes requests over the internet using a standard request/response protocol. Upon receiving a request, the service software immediately goes to work interpreting the request, performing the requested work, and issuing a response containing the result of the work back to the client or to the correct destination(s) as defined in the request.

Since the services layer is run in a server environment independent of all the clients and must be available so that clients may use it freely, this helps to define clear requirements of a good server solution:

1. The client software may be run or used at any time, and because the client expects to be able to reach the server component through the internet at any time, server applications must run on a special type of computer, a dedicated server machine that is always running and is always reliably accessible over the internet. High availability is a key requirement of our solution.
2. This imposes critical uptime and reliability requirements on the computational infrastructure that runs the server software. Because of the always available nature, the server is traditionally a dedicated machine (or many dedicated machines) that must remain powered on and must always be standing by, always ready to accept requests from a client. Later we will look at the cost impact of these requirements. Low cost of operation is a key requirement of our solution.
3. To support best performance and smooth operation of the client, the server must strive to be as responsive as possible by being consistent about performing these steps in rapid succession, without introducing delay. The end user will have the best experience possible when all delays associated with the server communications are minimized. The client applications we target are ones that expect the server to perform its work in real time or near-real time, making low latency and responsiveness a key requirement of our solution.
4. To fulfill its duty to clients, the server must have enough capacity to be able to handle requests coming from every client running worldwide. Depending on use patterns associated with the client software, there exists a possibility of periods of time when a high volume in requests will be made to the server, making the ability to scale to demand a key requirement of our solution.

CLOUD-BASED CLIENT-SERVER SYSTEM



Clients can be physical devices such as mobile phones, computers or other environments such as a web browser which has access to the internet. Clients use the internet to communicate with the server.

Clients can also be components of other cloud-based services systems.

FIGURE 2

CLIENT-SERVER INTEROPERATION

RELATIONSHIP BETWEEN CLIENT AND SERVER

1. All around the world there are end users who every day will download, install and run applications of every kind. This is done in almost every current computing environment imaginable. These applications are acquired, usually as a download, from a company that not only has created the application but is also handling the continual operational aspects of providing the application.
2. Somewhere in the world, the company that has provided the application is also running a server, or more likely a great many servers, which are required to provide continual delivery of services needed by the application. The application connects through the internet to reach necessary services, which it knows how to find, and utilizes the services as needed. While the application may be in a geographic region distant from where the servers are located, the two components work together seamlessly, provided that the server component is always reachable and stays fully operational.

BENEFITS TO SOFTWARE DEVELOPERS

1. This pattern forces the developer to follow the sound principle of strategic separation of concerns. In this case there are two concerns. The first concern is the portion of the code responsible for providing the user experience, operating within a specific environment. The second concern is the portion of code responsible for doing the computational work needed by the application and exposing that as a service. This pairing promotes good architectural software design by requiring development of an API to explain exactly how the two concerns are related and how they communicate. When structured properly, this results in well-defined software components that are easier to develop and maintain.

2. Because very powerful computing hardware is available to run the server component, it becomes possible to leverage its power to write services code that performs very compute-intensive work, benefiting the client application. If a client-server model was not used (i.e. the client application attempted to perform all work for the whole system), the resulting product might operate very poorly or may be impossible to develop, due to the severe constraints and capacity limitations associated with a variety of platforms where the client operates. Having a powerful server available to the client enables development of high performance client applications which can carry out impressive tasks to the end user, despite running on very modest hardware.
3. Because the services layer software always executes in a known and controlled server environment, usually only a single implementation will be developed. Developing only a single implementation of the services layer simplifies and lowers the cost of development, testing, deployment and maintenance of the project's code. This contrasts with the client portion of the code which might have many implementations, each specific to a particular platform, and having special code to accommodate that platform's unique capabilities and limitations.

Client-Server Application Example

Let's say your company has developed a hypothetical mobile app, running on iOS and Android, which we will call "Crazy Faces". "Crazy Faces" is an imaginary app that lets you use your phone to snap a selfie, and then after choosing one of several fun special effects, the app will entertain you by creating a video rendition of something humorous having been done with your face. The application will let you share this humorous video clip on social media. Which, or course, will contain a download link so all your friends can get the app and try it on their own face.

The "Crazy Faces" is an excellent candidate for a client-server design, assuming the work required for converting a picture of a face into a video requires a lot of computational power. Therefore, your company would have one or more teams developing the mobile app, while another team would develop the image processing algorithms as a server component that the mobile clients can use for the application to work smoothly.

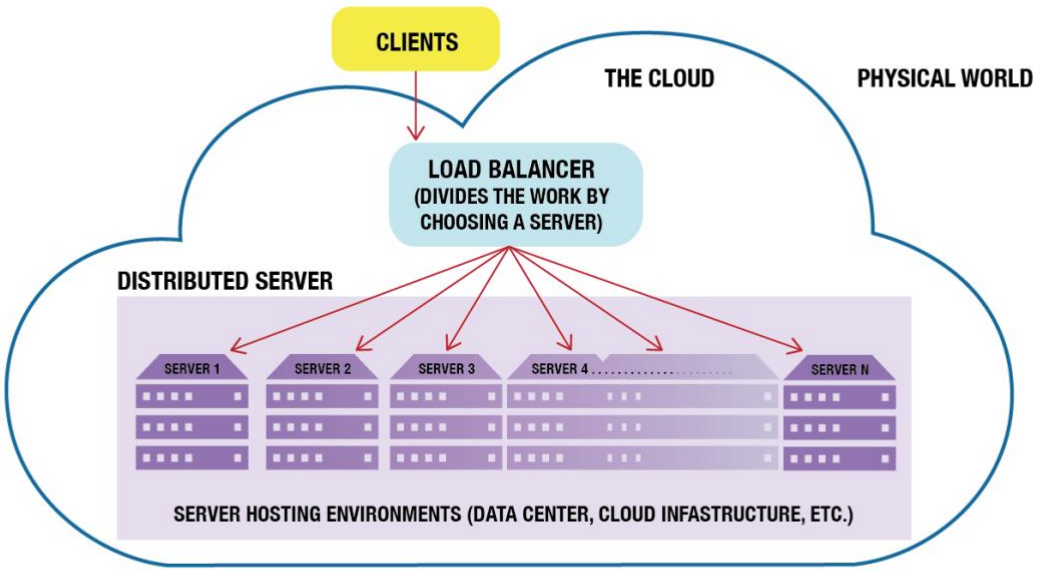
The Distributed Server – A Server Operating at Scale

Scalability refers the web applications ability to function satisfactory and at a high level as demand for its use grows. To ensure scalability, developers must carefully consider operational aspects, such as how to build, monitor, maintain and adjust the given server component.

A distributed server made up of many individual servers will scale overall capacity. A distributed server is implemented as a collection of identically configured machines all running the same services code and interconnected such that they appear to the client as a single, very powerful server.

The volume of requests handled by the distributed server will be proportional to the total number of client application instances in use at any given time. This volume will fluctuate based on the changing use patterns of users launching and interacting with given application. Furthermore, the volume may increase continually for days or weeks if the application happens to become very popular. It is important that these factors be considered when building the distributed server. This analysis is commonly referred to as capacity planning.

ILLUSTRATION DEPICTING A DISTRIBUTED SERVER



A distributed server consists of many individual server machines connected together such that they collectively behave as a single server of much greater compute capacity.

FIGURE 3

SERVICES AND STORAGE

The server infrastructure just described provides an environment where services can reliably be operated, delivering ongoing support to the client application. There are different ways to classify types of service, and there is an important distinction that we must make. This presents a design choice that will have a big impact on capabilities and limitations of our server infrastructure, influencing key aspects of how it is designed and deployed.

Here we will consider a distinction around whether a service uses internal state to influence generated results. This distinction will later be used to take advantage of opportunities to constrain and simplify our proposed solution, scoping it to a more pragmatic final form.

STATEFUL SERVICES

A stateful (or "impure") services layer is one where the same inputs given at different times may produce different outputs. A service is stateful if it has any side effect that can influence future computations to cause a different result to be produced. A service is also considered stateful if it depends on a data source other than the inputs, such as the results of a request to another stateful service. If there can be no guarantee that the same inputs provided multiple times to the same service on any host will produce the same outputs every time, then the service is stateful.

STATELESS SERVICES

A stateless (or "pure") service is one that always produces an output dependent only on only the inputs. Services that use internal state and still fulfill that requirement are also considered to be stateless. Every request/response pair must operate as if in isolation, meaning that the outcome or side effects of any prior inputs will never influence or change the outputs for any future request given the same inputs.

STATEFUL VS. STATELESS SERVICES

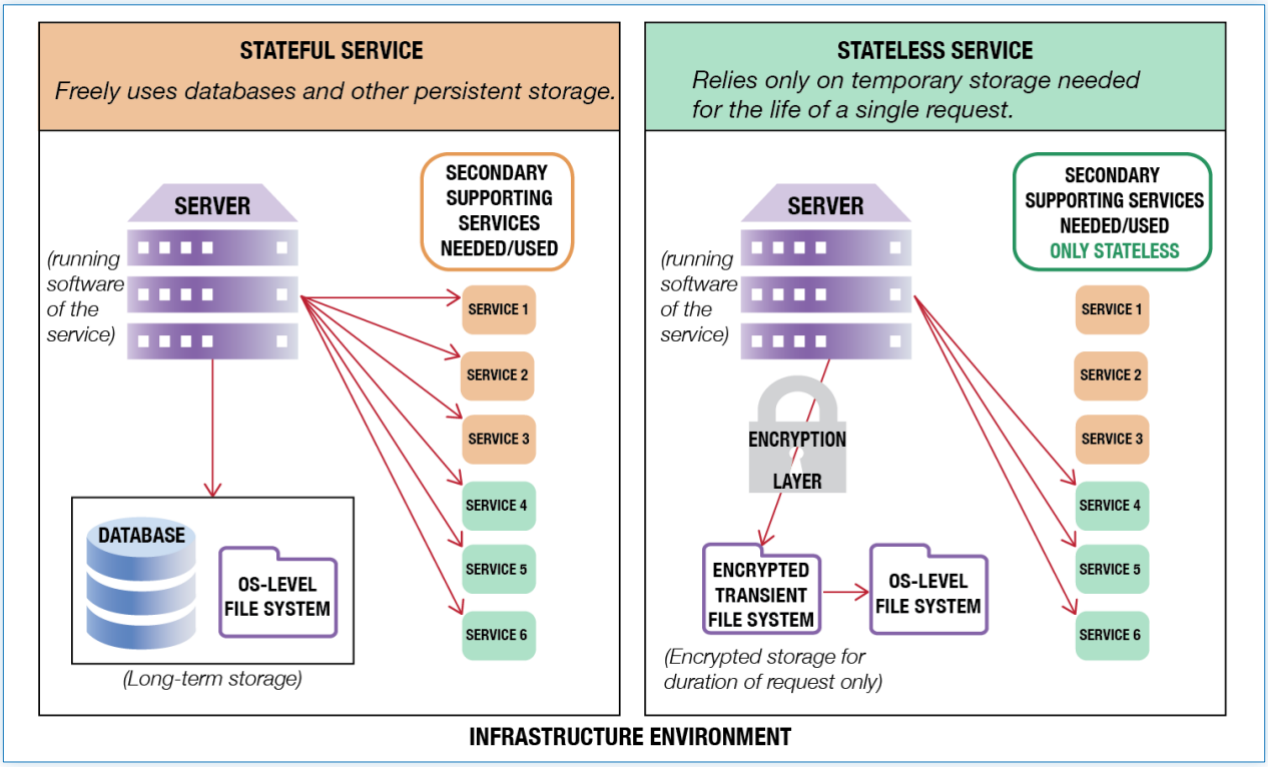


FIGURE 4

The distinction is important because only stateless services allow us to execute the same request multiple times on different hosts and use confirmation that results were the same as a form of peer work verification.

Statelessness does not preclude the ability of the dService to employ caching mechanisms that would allow correct results to be produced more quickly.

It also does not preclude the generation of temporary files, whose paths may be returned as part of the output and provided again as part of a future set of inputs so that the results of past work can be built upon in future work. To facilitate this, we intend to support miner affinity (the ability for the CGN to route a given request from the dApp to a specific miner instance) and support for clustered miners (a number of miner instances running on the same LAN and having access to a shared file system). Both approaches permit the dApp to accumulate intermediate results remotely and make use of them later to minimize transferring of intermediate results between different miner instances.

Types of Hosting Environment

In traditional models where hardware is collocated, all machines comprising the distributed server need to be set up and operated in a place that provides the right environment for the servers to operate, and which provides power and network connectivity to the machines so that they can be configured, connected and left operating as a distributed server. We call such a place a hosting environment. There are three primary types of hosting environment.

Premises

This refers to a distributed server system set up using machines physically located within the business that is using the servers. Usually there will be a dedicated server room, where the servers are physically housed, with air conditioning to keep the servers cool, backup power supplies to keep things running in case of a power failure, and a security system in place to keep employees away and to prevent tampering with the systems.

Data Centers

Data center operators run a business whose core competency is acquiring and operating large number of servers in a controlled environment, guaranteeing uptime and availability.

A data center operator will usually guarantee a high level of service by taking full responsibility for a myriad of critical activities associated with keeping the data running optimally. These critical activities must be done recurrently. Typical duties include (but are not limited to):

- Providing the real estate, building space and property management needs behind operating a large, dedicated facility
- Providing abundant rack space in which servers can reside
- Giving the tenant full control of their machines, allowing complete flexibility of configuration and use
- Providing conditioned power, including backup and fail-over power supplied by batteries and standby diesel generators
- Operating all critical data center systems with redundant backups for greater reliability
- Regulating environmental factors such as airflow, temperature and humidity
- Providing high speed Internet connectivity
- Providing physical protection of all equipment by tightly controlling access to the facility
- Offering an ability to purchase additional capacity for expansion if needed
- Handling fire detection and suppression
- Providing regular maintenance of all systems
- Providing 24/7 monitoring of systems by on-site staff to quickly detect any anomalies
- Ensuring rapid remediation of any problem, such as a hardware failure, by repair or replacement

Data centers are expensive to build and are costly to operate, due to the construction and operation of the expensive and sophisticated systems required to guarantee a high level of service promised to the tenant in the lease contract. All these activities are valuable or even essential for mission-critical applications. However, we are targeting non-mission critical Web applications, which have more modest needs and can be delivered effectively at an acceptable level of service without the costly overhead of an extremely high SLA environment.

The data center operator generates revenue by leasing blocks of dedicated computational capacity in the form of arrays of servers, assigned for exclusive use by the tenant. Compute capacity is usually leased by contractual agreement, where the contract specifies a term, the number of servers to be reserved exclusively for use by the tenant, and a promise to maintain a specified service level (the SLA). This payment model is arguably both inflexible and expensive for web application operators. This is due to the difficulties of estimating the projected volume of use and therefore the number of machines necessary to lease for a given term. This often leads to situations in which server capacity is over-leased.

CLOUD SERVICES

Cloud computing infrastructure is itself hosted within one or more data centers, so this approach has similar advantages and disadvantages as compared to using a data center.

Over-leasing of capacity will remain a problem for the same reasons given for operating servers in a data center. You still need to lease extra capacity to have it available on hand to handle upward uncertainties in the amount of use and the popularity of the client application.

However, these services do offer greater flexibility in the ability to increase or decrease scale by adding or removing machines as needed. This flexibility comes at a cost and may not be as valuable as it first appears. This is because changing the number of reserved machines is not done automatically and is something that must be continually managed by the web application operator. Management cost is easy to overlook. Indeed, many companies can tell a story of exorbitant cloud computing bills, incurred because a server was accidentally left running longer than necessary.

ALTERNATIVES

There are few, if any, good alternatives to the above three standard approaches. Please read on to learn how we plan to address this.

Problem Background

An Ongoing Responsibility

As a growing number of new applications are continuously being developed using the client-server model, the need to address challenges to simplify and reduce costs of deployment and operation of server infrastructure will remain a high priority.

A key point to take away is that the responsibility of the developer of the application does not end after the software has been developed, tested and delivered to the user. To keep this software operating reliably, the developer incurs a responsibility to operate services reliably, a commitment that will last the duration of the service life of the application. If an application cannot reach its required services or if the server infrastructure become overloaded by too many requests, the servers may fail to provide services in a timely manner, and the application will slow down or stop working. Therefore, the developer must invest in enough server infrastructure resources to fulfill on its long-term commitment to continually keep the services layer operating.

Deploying the Distributed Server

The distributed server is a collection of server machines set up in a hosting environment, configured to work together as a single server. For the operator of a DApp or Web application to deploy and operate the software product, server machines must be procured, configured, operated, and maintained in some form of hosting environment. This may be directly, by buying and setting up the infrastructure oneself on premises or in a data center, or it may be done indirectly using cloud-based services providing the needed infrastructure through a subscription model. And then the services software must be installed on those machines and operated for the service lifetime of the application

As we have discussed, there are currently three approaches commonly used to accomplish this. Here is a recap of the three approaches, and their trade-offs, from the point of view of the DApp operator:

Hosting Environment	Operational Approach	Startup Costs	Fixed Costs	Convenience
Premises	• You provide hosting environment • You purchase server hardware • You operate server hardware	Very high	Moderate	Low
Data Center	• You lease hosting environment • You purchase or lease server hardware • You operate server hardware	High	High	Medium
Cloud Services	• Hosting environment is provided • Server hardware is provided • You lease and operate server hardware	Low	Very high	High

Premises

This approach is usually not viable because the developer and operator of an application will not want to hire additional staff having the unique skills needed to design, build, install and operate server machines due to the increased cost and complexity. Startup costs can be very high. The cost to benefit ratio of this approach is often prohibitive to providing a properly scaled environment for operating and maintaining server machines over the duration of the service life of the application.

And as mentioned before, the operator needs to put in place excess capacity, so that it is on hand in case of the need to handle load surges or unexpected rapid growth in popularity of the application, which might not even happen. This increases costs even more.

Application operators often turn to the use of a data center to overcome some of these challenges.

DATA CENTERS

Data centers have a high built-in cost structure because of the immense amount of technology that has been put in place to reliably support the environmental, electrical, security and reliability needs of tens or hundreds of thousands of servers, all of which contribute to high fixed costs, in addition to high startup costs.

The need to put in place excess capacity still exists for the same reasons. Acquiring and operating this excess capacity is even more expensive in a data center.

For greater operational convenience, application operators often turn to the use of cloud services to mitigate some of these challenges.

CLOUD SERVICES

Cloud computing infrastructure is itself hosted within one or more data centers, so this approach has all the built-in cost structure of a datacenter, and then some. The other advantages and disadvantages are similar to what they would be with use of a data center.

However, over-leasing of capacity is still a problem, for all the same reasons given for operating servers in a data center or on premises. You still need to lease extra capacity to have it available on hand in case of unexpected demand in use or runaway popularity of the client application.

However, these services do offer some flexibility, by providing the ability to increase or decrease scale by letting you easily add or remove machines as needed. But this flexibility comes at a great an often-unexpected cost, and therefore may not be as great a prospect as it first appears.

This is because changing the number of reserved machines is not done automatically by the cloud service, and is something that you, the operator, must be continually monitor and manage. This cost is easy to overlook, and it's easy to make mistakes. Indeed, many companies can tell a story of exorbitant cloud computing bills, incurred because a server was accidentally left running longer than necessary.

These problems can sometimes be solved through automating scale by modulating the addition and removal of resources in response to changing demand for services. But attempting to operate such a solution will create a whole new set of problems. For example, such an automated system allocating and releasing server resources could be scripted by the user of the cloud service. Such a resource management system would try to control costs by always running the optimal number of servers at any given time. While this can be done, the effort will usually be thwarted in the end because of the way cloud services price the cost of resources (charging much more for short duration use commitments).

Here are three arguments against the use of cloud computing services:

4. The underlying hardware has all the high operational costs of a data center baked in, and therefore must have a similar but higher cost structure, even when leased for long periods of time.
5. Automated solutions to address over-leasing of capacity are possible, but these require extensive and costly extra work from the application developer. The application developer may choose to develop an automated solution in-house or may choose to outsource this project, but either way it will come at a substantial extra cost and would require its own operational resources. Some cloud providers also offer this capability for an additional fee.
6. Should such capacity-adjusting automated solution be developed, it would most likely be ineffective. The continual acquiring and releasing of server machines will result in shorter lease durations, and will cause onerous, short-term resource pricing models coming into effect, further increasing your

costs. Thus, the workable solution that strives to drive down costs would have the side effect of driving costs up! The result would be more negligible gain or perhaps even a loss.

Alternatives

There are few, if any, good alternatives to the above three standard approaches. Please read on to learn how we plan to address this.

Capacity Planning and Management

When deploying a distributed server, there are two aspects of it which require careful planning:

1. Initial capacity planning to ensure the right amount of server resources are acquired to begin operating the services.
2. Ongoing management of capacity, making changes as needed to adapt to changes in use patterns and business needs.

Each of these imposes its own unique challenges which must be addressed:

Initially Planning Capacity

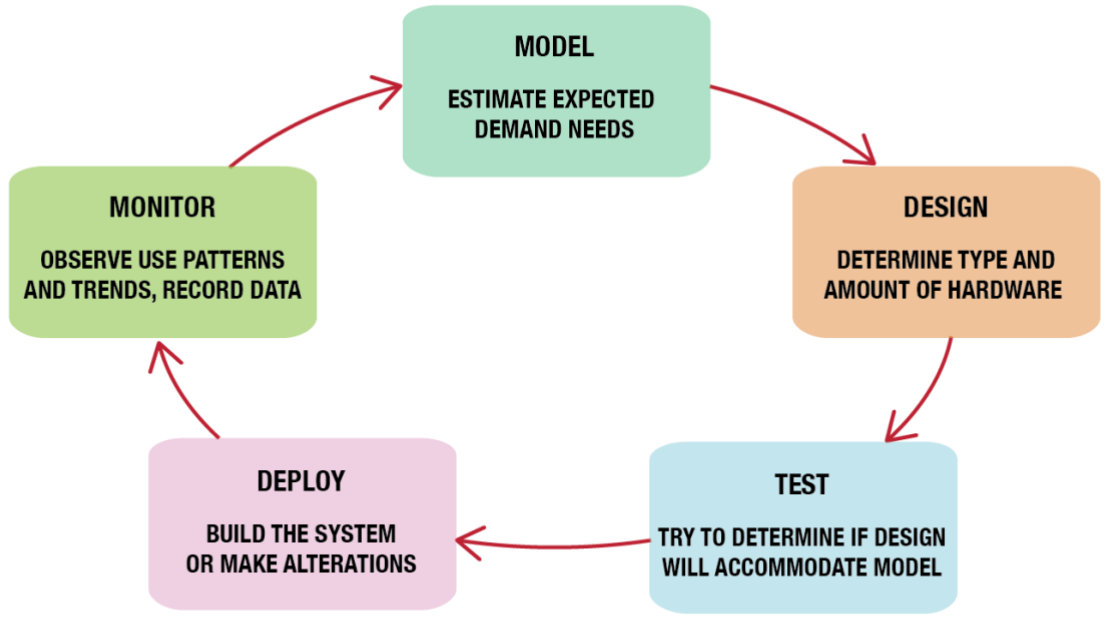
To configure a distributed server, the capacity planner will need an estimate of the number of machines required to process the highest anticipated load of incoming requests from Web applications. This estimate will be based partially on understanding how the application is typically used and anticipating what demands it likely will make when using services. The capacity planner will also need an estimate of the popularity of the Web application. Popularity is gauged by knowing how many copies will be downloaded, and the number of users likely to be using the application at any given point in time. From this information, the capacity planner can estimate the peak demand load requirements of the servers, as well as an idea of how these demand load requirements may change over time.

The success and growth of new web applications are difficult to predict. This makes it inherently difficult to estimate the number of machines required to be dedicated for use to the distributed server.

Actively Managing Scalability

Companies who develop web-based application must also ensure quality delivery of service over time. To do this, the company must have staff who continually monitor use patterns of the distributed server. These staff members typically work closely with the capacity provider (usually another company) to make required changes and adjustments. These adjustments are done to ensure the application continues to work smoothly over the durations of its service lifetime. During this span of time a multitude of variables can force change and drive new requirements for the amount of server capacity needed by the application for it to run smoothly.

THE SCALABILITY MANAGEMENT CYCLE



Responsibilities incurred when deploying and operating a services infrastructure to support of a client app, adapting to changes in business needs and popularity of the app.

FIGURE 5

INEFFICIENT USE OF RESOURCES

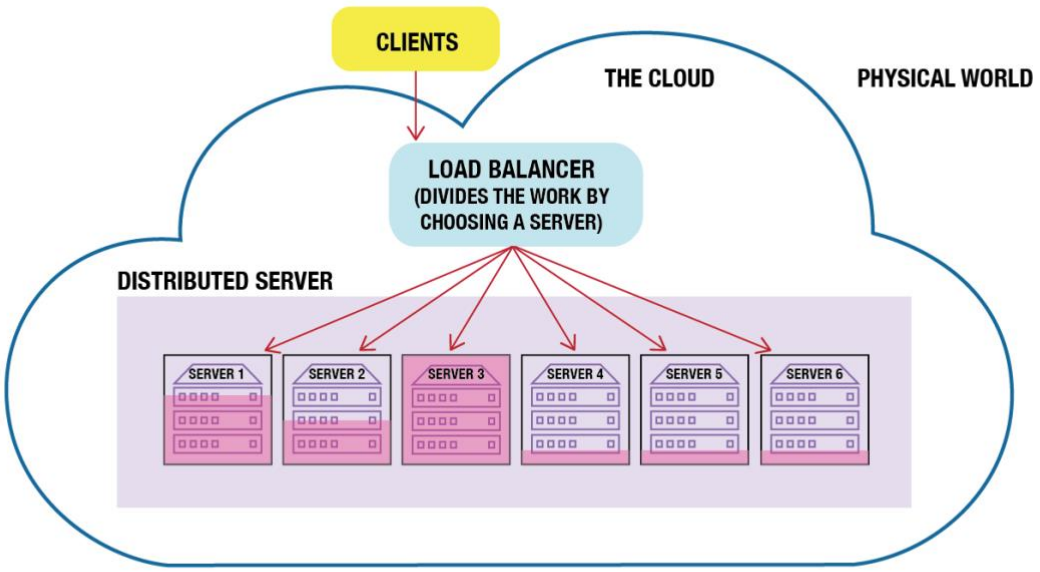
It is an unfortunate reality that a significant number of servers, costly to acquire, deploy, operate and maintain, must inevitably remain idle, operating on standby but reserved for the infrequent load crisis situations that must be supported by available capacity.

To ensure that the Web application will run optimally, the operator must overestimate the application's popularity, leaving headroom for actual use that may be higher than originally estimated. This ensures that the fixed-size distributed server component will be able to support all needs of the Web application without overload or failure. Thus, it's a sure thing that more server machines than actually needed must be acquired, paid for and held in reserve. Failure to do so will result in the distributed server having availability issues when there are large fluctuations in use.

The certainty that there will be wasted (i.e. leased but unutilized) server machines in the distributed server is a disappointing reality that comes along with deploying server resources for applications of unknown popularity. These machines have been acquired and set-up at substantial cost, are reserved for exclusive use, but usually aren't performing work. Idle server resources will be used only if there are unforeseen heavy demands for the given service.

This is one of the problems our focused architecture addresses, eliminating the costs associated with maintaining this necessary overhead. It would be ideal if the operator of the Web application could simply pay only for the server resources that are actually in use. And possibly more importantly, to not have to pay to manage and set up resources at all.

UNDERUTILIZED RESOURCES IN A DISTRIBUTED SERVER



Real-world capacity utilization of server machines inside dedicated distributed server system. It's virtually guaranteed that much of the time many server machines will be idle.

FIGURE 6

SUMMARY

For every (client-server) application ever developed and sold, its developer takes the responsibility of operating a multitude of server machines, systems required to support consistent, continual availability of the application by its users. For the operator to deploy and operate these servers, they must invest in the infrastructure and labor needed to acquire, deploy, configure, maintain, and pay for use of these complex distributed server systems. Initial and ongoing expenses incurred can be significant. Furthermore, keeping these systems operating requires a long-term operational commitment, as they must be operated and maintained for the serviceable lifetime of the given application.

A critical aspect of maintaining these systems is the additional need to actively monitor load patterns, detecting when the system needs added capacity, and proactively addressing those problems. The operator must also assume responsibility for and pay all costs associated with fulfilling on this commitment, ensuring continued long-term operation of the application as use adoption and other conditions change.

Collectively, because of the way infrastructure is currently acquired, these challenges and responsibilities require application development companies to make substantial investments in time and resources. A solution addressing these problems would be very compelling.

PROBLEM STATEMENT

This section seeks to outline and provide a concise description of complementary pair of problems related to global computational infrastructure: cost and complexity caused by inefficiencies in consumption, and an untapped supply of capacity due to inefficiencies in utilization.

INEFFICIENCY OF CONSUMPTION

Reliable computational infrastructure is costly to acquire and operate. This results in difficult-to-meet needs commonly shared by businesses that sell internet-based applications and services:

1. Confidence that the product or service they commit to offer, and support will operate consistently and reliably
2. Need to minimize fixed costs
3. Have variable costs that are both predictable and proportionate to consumption
4. Make efficient use of available resources, ensuring enough availability, without the waste of underutilization
5. Ensure flexibility and awareness, and adapt quickly to changing needs, unhindered by inflexible dependencies.

INEFFICIENCY OF UTILIZATION

There exists a vast amount of untapped global CPU power in the form of computers that sit idle or are not fully in use. These are computers worldwide that are powered on 24 hours a day, but which sit idle most of the time, not being asked to do any work. Often these computers are spare capacity servers kept online and ready for use, but other types of resources often run underutilized.

Let's consider just how much idle capacity there is.

1. There are an estimated 75 million servers powering the internet*
2. According to Gartner and IDC, it is estimated that there are two billion personal computers in use worldwide, even after accounting for retirement of old machines†
3. According to Forbes an estimated 30% of the world's computers at any given moment are powered on but are sitting idle and doing absolutely no work‡

At any given moment, tens of millions of server-grade machines and hundreds of millions of personal computers and mobile devices sit idle. Of those, uncountable millions also have high performance GPU capabilities. Machines having a GPU, including modern forms of server hardware, mid- to high-end PC's with dedicated graphics cards and newer mobile devices, can deliver an immense amount of computational work. When these devices are powered on but sitting idle, that capacity is not being used, and it essentially goes to waste.

* Live internet data estimator
† Computer activity
‡ Idle computing

Requirements and Constraints

General Intent of Solution

In the following sections, we will offer a solution, in which we propose putting in place a new ecosystem which creates an incentive to put idle computer power to use, by paying for its use to reconfigure it into a contributing worker node that participates on an as-available basis to do the work required by the new computational infrastructure or anyone requiring near to real-time high volume content processing and delivery. This infrastructure will be made available as a pay-per-use service component that will offer great value to business.

This proposed system will take idle computers, ones whose availability and capacity are unpredictable, and combine and restructure them into a new aggregate form to provide a service that overall is highly available, responsive, scalable, and flexible in adapting to changing needs, thereby adding value. The system will offer a low cost of use, resulting from the relatively low value of unpredictable, idle CPU time as compared to the high value of always-on CPU power. The system will be self-operating, thereby freeing business users of the burden and overhead associated with operating traditional computational infrastructure.

Put Idle Resources to Work

If only a fraction of that idle CPU power could be captured by financial incentive, harnessed, restructured and put to work in the form of an enterprise-grade computational infrastructure service that is also fast, flexible, scalable and competitively priced, great value could be offered.

Constrain to Services Not Requiring Persistent State

We will build our first iteration of the system to support the operation of stateless services. This scopes the problem by eliminating a substantial area of complexity (around the deployment of data stores and having to cope with requests which may have side effects). But even with this constraint, there remains a huge opportunity, an opportunity big enough for this project to have many compelling uses. We believe that stateful services are a different class of service that will usually be hosted traditionally, such as in a data center or on cloud infrastructure. Operating data storage infrastructure brings with it complexities that we do not wish to introduce into this project. Problems such as how to secure the contents of any arbitrary database, how to provide an ability to back up the data stores, and permitting the maintenance and upgrading the data storage software, etc.

By scoping our services layer to stateless services, we are left with a flexible design which can take advantage of several novel approaches to supporting efficient and optimal work generation using a pool of servers. We will be able to gracefully handle failures incurred by running services on hardware whose characteristics may not be well known. For example, we can recover from a worker node failure by simply retrying the same work on a different worker node. Or we can even send the same work to multiple worker nodes in parallel, creating a race whereby we simply return the result from the first worker node to respond. These are approaches that would not be possible if services were stateful and executing them repeatedly produced different results.

Transition to Microservices

Stateful request handlers often form the backbone of monolithic enterprise and consumer applications, ones that have database and storage characteristics built-in. They utilize their inherent storage of state to provide capabilities for managing or operating stateful systems which hold and act upon continually changing data from the past. These include applications such as social media, online banking, stock trading, cloud-based ERP, CRM, SFA products, and most commerce systems. Any system where the service layer is responsible for creating, reading, updating and/or deleting stored data (CRUD), will depend on a services layer that is inherently stateful.

However, many monolithic enterprise applications are migrating their legacy codebases toward more modern microservices-based architectures. In doing so, stateful services are being broken down into more fine-grained service layers, many of which can be built as stateless. Even applications which are overall stateful, when architected well, have a need for significant amounts of stateless request processing.

FOCUS ON NEAR-REAL TIME DELIVERY

We have chosen to scope our solution to the use cases of interactive, enterprise applications, ones which require low latency and that must operate in real time or near real time. What this means is we will build a service delivery infrastructure that will determine whether to perform work as immediately as possible, by minimizing the introduction of delays in matching, or to perform the work as quickly as possible, even if it means waiting a bit for the availability of better hardware.

This is a key point. While other similar projects have chosen to specialize in creating a market for large blocks of CPU time to power long-running jobs such as rendering, computer learning and other large-scale, computationally expensive scientific research projects, we are taking on a uniquely compelling (and somewhat more difficult) approach, one that will be of great value to consumer and enterprise applications that need to operate in near real-time, to responsively react to user interactions.

Our solution is well-suited for quickly handling short-lived service requests, aiming to achieve client-server round-trip response times as short as a tiny fraction of a second (for very small jobs). Yet our solution can also power longer running services. The ability to support client applications that use a microservices model is one of the areas where our project stands out from the other projects, which either don't address this or hope to support it in the not-so-immediate future, because their solution is not capable of working around limitations in the transaction throughput of the Ethereum blockchain.

Introducing the Content Generation Network (CGN)

A flexible, scalable, pay-per-use distributed server infrastructure with edge connectivity offering businesses a system for reliably operating the enterprise-grade services that deliver a wide range of applications and services to end users, powered by a new utility token and backed by the Blockchain, a new ecosystem ignited by reclaiming vast amounts of idle CPU power and putting it to work.

The CGN

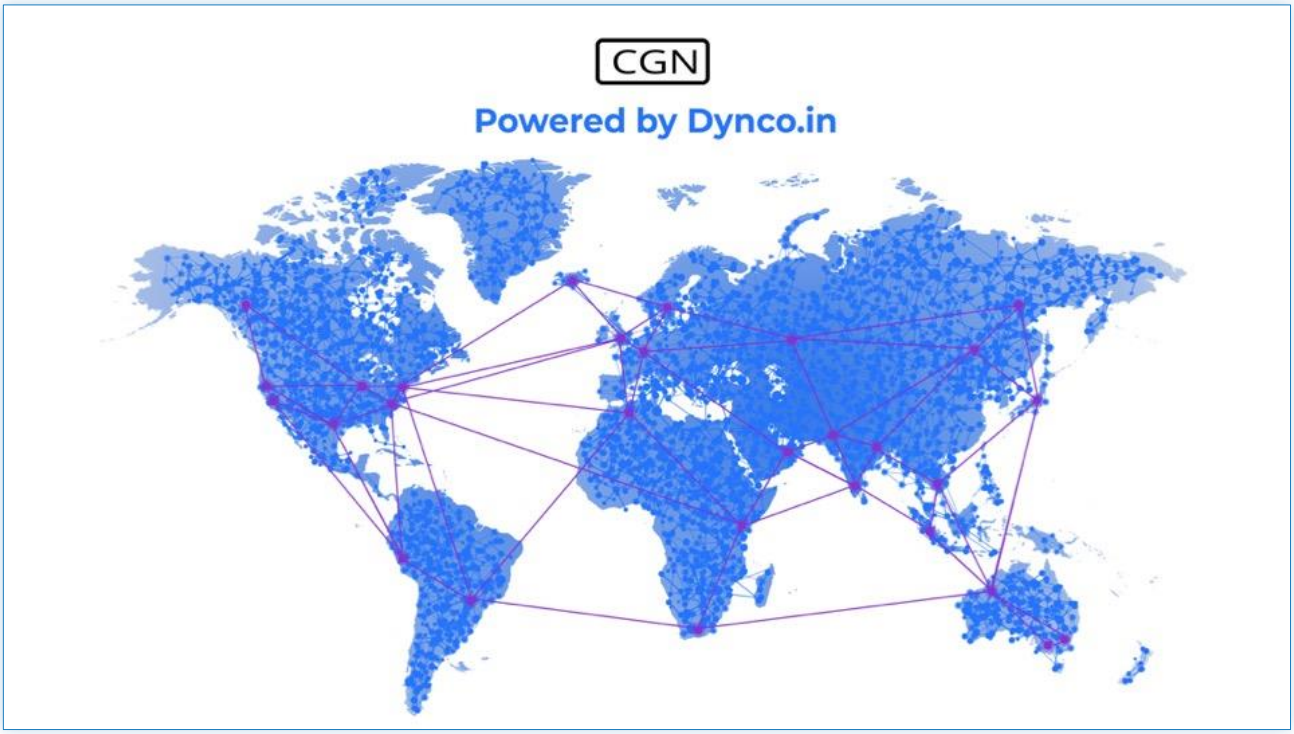


Figure 7

Addresses the Problem of Acquiring, Operating and Scaling Servers

- We will provide an alternate approach for the delivery of Web applications, making available an arbitrarily large number of otherwise idle computers to form a massively scalable distributed server to power Web applications.
- The CGN will have zero fixed costs, as an operator of a Web application pays only for consumption of services and resources actually used. There will never be any costs incurred by idle server time.
- There will be no need to do capacity planning, design a distributed server infrastructure, or operate one. That will all be done automatically. The application developer will be able to focus on development of the best application possible and know that its services will be deployed at arbitrarily large scale. The entire system will work reliably without any intervention or oversight.
- Server resources will be priced at the lowest cost possible by using resources provided by contributors. This coupled with the dramatic labor reductions of no setup or maintenance of one's own cloud

infrastructure and no longer paying for idle services time results in large cost savings over traditional cloud providers.

Robust and Well-Scoped

- Our design addresses real-world needs immediately upon release, with an emphasis on taking a pragmatic approach that does not try to solve every possible problem. We provide the necessary components to enable others to add to our network.
- We have focused the initial solution on addressing a useful set of the most important problems, within a well-defined number of use cases.

Additional Unique Benefits

- We are building a solution that lets client application developers effortlessly deploy a highly scalable services layer, while providing the flexibility to adapt to changing requirements.
- Our system will naturally provide edge support for services. The system will make a best effort to connect each client with an available server that is in close physical proximity to the client. This will result in lower latency and better overall performance as the client communicates with the server.
- We will put vast amounts of currently unused computing power to work, by creating a new sharing economy that incentivizes contributors to put idle CPU power in homes, businesses and data centers to work. We will do so by providing the distributed server capacity needed by Web applications at a reasonable cost and compensate providers of resources from the proceeds of its consumption.
- We will power a thriving ecosystem that drives an economy for the exchange of CPU power by introducing a utility token that meaningfully supports this purpose. This token will be used both to pay for consuming CPU power, and to compensate holders of unutilized CPU resources by paying to commit those resources to system's resource pool.
- The service will be priced at varying tiers of service in terms of fiat currency but transacted using tokens. We believe enterprise users will find this especially appealing, as pricing will be kept predictable by removing the variables associated with having to exchange currency. This will be done through internal currency conversion and will not affected by how the value of the token floats with respect to the fiat currency.
- We will provide the valuable benefit of billing at a fine level of detail. Therefore, we propose a robust payment system that will accept and distribute payments for each single use of a service on a per-request level, regardless of its size or duration. This necessitates a system for handling very high-volume microtransactions. For this, we propose a robust solution which will combine an off-chain transaction management system with the blockchain, without giving up any of the benefits of the blockchain.
- We will support a flexible, sophisticated batched payment model that collects for use of the system but divides the proceeds such that each party having a role in the development and operation of a DService and the associated infrastructure to each be paid an appropriate share of the proceeds collected on a regular basis.
- This model also includes the ability to dedicate portions of collected revenues to charitable causes. CPUcoin intends on building an annual CPUcoin fund for use by third world country organizations that don't have the ability to set-up and utilize cloud resources themselves. Giving back is built directly into our model, and CPUcoin will lead the donations by allocating a percentage of all CGN-earned CPUcoin to a donation wallet for annual distribution to worthy causes.

4.

A Solution Driven by Assured Adoption

In a space with multiple ICO's working on similar systems, each proposing to create some form of a sharing economy for unused CPU power, it's important to understand our project's differentiation strategy. We are prepared to answer this frequently-asked question:

"There are several ICO's working on similar projects
by groups such as Golem, SONM, and iExec.
Explain your differentiation strategy."

Positioning

Figure 8

Well-Scoped

A well-scoped solution is less complex, risky and costly to build, while offering the most important benefits. We are not trying to build a blue-sky or "does-it-all" solution. We do not propose an abstract, loosely bounded project that promises big hopes and offers potential possibilities, and ones which may never gain traction or be used by real businesses because the origin of those ideas is not grounded in real world business problems.

This is where the CGN project contrasts greatly. Our project is not an academic exercise. We offer a well-scoped proposal to build a solution to address a well-understood, existing real-world challenge. Our solution, when completed, will be immediately used and depended upon by a real enterprise class software company to fulfill on its immediate scalability and edge processing needs.

Enterprise commitment to adopt

An enterprise commitment to adopt the project for real-world use will add great value.

This project will be tested and used by Equilibrium for their Content Compliance Cloud buildout, which will adopt the CGN for use with an existing product line. This product line not only has a proven track record, but also has a new Microsoft Cloud Alliance Partnership and a new hosted cloud service with a

large target growth plan. The company will license MediaRich Server from Equilibrium to create the first DService for dynamic imaging infrastructure and enable it to run on our new scalable platform. The existing MediaGen product has already been seamlessly integrated into many solutions, including Microsoft's SharePoint and Office 365 cloud service, targeting over 200 million user installed base. Soon it will be delivered using the first DService, running on the CGN.

Porting MediaGen to run as a DService is not simply a proof of concept but will become a permanent part of the product itself. This will aid to rapidly prove the validity of our design. It demonstrates the value of applying business practices to strategically scope and constrain a solution to its most essential aspects, making it realistic to deliver a high-quality service in a timely manner. We will continue to commit to serve that company's needs by guaranteeing our proposed system to continue to work reliably, providing its important services as the needs of applications scale. The company will show the world that our system can be depended on by other enterprise users who also need to rapidly deploy and scale their own distributed services for the client applications they develop.

A UNIQUE APPROACH TO GENERATING PROJECT CREDIBILITY

This grounded approach brings a type of credibility to our project, which we have not found in most we've seen. The project will be delivered with prior battle tested DService technology and will provide immediate value.

The cross-licensing deal, whereby a new Content Compliance Cloud (connected to many cloud providers), ensures there will be built-in demand and impressive projected volume. The attraction to participate in our project will therefore be more compelling than participating in similar projects of other ICO's working on much broader, un-scoped variants of this type of problem. While they present grand plans and discuss big ideas, few offer any form of known, existing user base. While we have architected our system in a way that ensures it will be heavily used from the outset, other projects have little more than hope that users will actively utilize their system.

ENSURING DEMAND FOR THE CPUCOIN TOKEN

Our resource-contributing worker participants will know up-front about the ecosystem's marketplace, as it will have known size and predictable market characteristics due to assured enterprise use of the MediaGen DService by the Content Compliance Cloud. It will make full use of the ecosystem, as its many enterprise customers must pre-purchase CPUcoin, to be spent as this software make regular use of the initial DService. These enterprise users' dependence on the ecosystem will help drive further development, adoption and growth in use of the system.

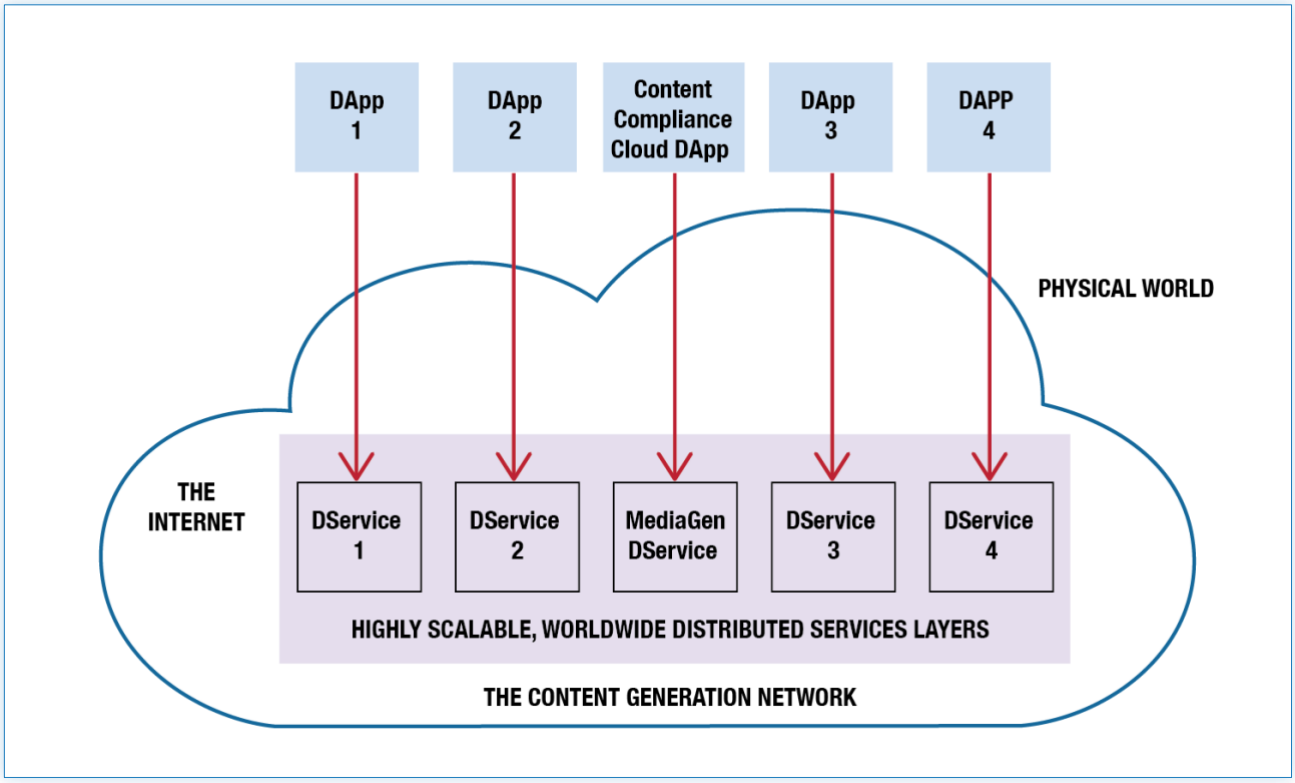


FIGURE 9

INTENDED TARGET MARKET

The solution described in this white paper is targeted at businesses that deliver and operate client-server software application that requires a server to provide a services layer to do services-oriented work for the client. Our solution specializes in use cases where the client software needs work to be in near real time. This means the infrastructure that we propose supports operating a services layer that can process requests in near real-time, providing high throughput and efficiently handling high volumes of short duration requests. This contrasts other solutions that specialize in the execution of long-running jobs, where fast delivery time and low latency are not critical.

We constrain the problem space by proposing a new type of computational infrastructure suited for stateless systems, i.e. systems where the server component does not maintain state on behalf of the client, and where every request is handled in isolation. This means that the server component always operates such that the result of any request is only dependent on its direct inputs and is never influenced by the actions or side effects of any past requests.

While this does place limits on the types of business solutions that our network can accommodate, we contend that there still exists a large need for this type of solution, which is widely used by a range of applications, such as mobile apps, that are simpler than entire enterprise systems. The class of problem our solution targets are data processing solutions, such as those employed by media processing applications, and any client app that transforms data such as gaming applications, mobile entertainment apps, and the list goes on. There are plenty of data and media processing applications for which our solution is an excellent and cost-effective match.

Example Use Cases

What can the CGN be used for? What types of software could be deployed in such an environment? The number of possible use cases is nearly limitless, but we wanted to offer families of use cases well-suited to the technology. There are potentially many existing projects that could immediately benefit by making use of the CGN.

Here are some examples of solutions where it would be easy or straightforward to adapt existing implementations for deployment within the CGN. This is only a partial list of what's possible.

Media

- Media transformation and processing*
- High-speed rendering of all kinds
- Video rendering from models
- Video transcoding*
- Automated video editing and assembly*
- Video subtitling
- Media provenance applications*
- Image recognition
- Optical character recognition (OCR)
- Digital signal processing (DSP)
- Audio analysis and retrieval of music information

Business / General

- Speech to text, text to speech
- Computational tasks and business logic support where result is determined from processing inputs
- General purpose data/format conversions (business-specific)
- An alternative platform for delivering microservices of all kinds
- Workflow tracking and management, with ledgered audit tail*
- Stock market simulations
- Web hosting, including dynamic image resizing*
- Backend for managed data storage services
- Large scale deployment
- Ecommerce enablement with worldwide distributed shopping API for using CPUcoin for shopping and loyalty programs.

Mobile / Gaming

- Server-side processing for generation of in-game experiences
- Server-side processing for generation of in-app experiences (mobile apps)
- Transacting in-app and in-game purchases

Mining & Crypto

- Cryptocurrency mining (GPU & CPU, non-ASIC)
- Ether mining pools
- Automated wallet distributions such as coin drops and crypto interest and incentives

* Indicates use cases to be supported by MediaGen, the first DService to be launched in the CGN.

Global CGN System Overview

Specialized for Distributed and Decentralized Computing

The CGN is both an ecosystem and a new type of cloud computing infrastructure. Backed by the blockchain, it delivers the secure, scalable, high-throughput request handling capacity required for reliable operation of a globally distributed, robust services foundation layer that today's cloud-based enterprise systems depend on.

The CGN delivers services built using common protocols, such as HTTP and REST, making it straightforward to operate existing services within our ecosystem. CGN server components, for which we have coined the term “DServices”, are server-side software components that process work requests issued by cloud-based applications, maintaining minimal state across requests. Running within and orchestrated by the CGN environment at scale, DServices deliver results with reliability, high throughput, including native support for edge delivery.

Attractive Pricing Model

Besides offering expected desirable characteristics such as consistency, reliability, low latency, high throughout and scalability, the CGN offers an additional benefit. Because of the CGN's uniquely predictable and affordable pay-for-use pricing model, we expect it to provide a compelling alternative to traditional infrastructure solutions in use today, ones that often come at high cost because they bill your business the full cost of all resources you have reserved, including resources not in use.

Uniquely Powered by Idle Resources

The CGN acquires and manages its enormous constituent capacity of worker nodes by taking an innovative approach. The CGN amasses and operates a large pool of idle or sporadically available CPU power, opportunistically acquired from a wide range of available machines and devices, which can be anything from mobile devices left on overnight to charge, to the excess server capacity of many data centers and server farms. Much of this hardware may offer high-grade capacity that’s mostly high availability, that’s not a requirement. Many kinds of excess capacity can be useful to the CGN.

The CGN will be able to work with computational hardware that may be inherently unpredictable, i.e. machines that cannot offer any guarantee of uptime or availability and which may fail to finish work, and devices that may offer variable and changing amounts of CPU power to the CGN due to other activities also taking place on the device. The CGN dynamically restructures these varying quality individual resources into a global network of quality computational power that is delivered as a well-managed, coherent service, presenting it to Web applications as a solid, dependable services layer that is highly available, runs worldwide, offers high capacity, is scalable, and providing edge support.

Built-in Support for High Throughput

The CGN uses an off chain microtransaction system, anchored to the blockchain, to record each use of all DServices to do work, no matter the size, ensuring payments are collected for all services consumed, and disbursing funds to all involved recipient entities based on established agreements. Because we support handling requests at high volume, we have also had to do the difficult work of designing our system to natively handle billing transactions at the same high volume, without putting undue strain on the blockchain. This capability is one of the things that makes our project unique compared to other CPU sharing projects.

CGN in Contrast to CDN

In contrast, content Delivery Networks (CDN’s) are an existing technology designed to globally distribute the delivery of static assets to client applications (such as Web browsers, but also DApps) so end users can access infrequently changing Web assets from a server geographically close to their computer. This shortens roundtrip traffic times, and low latency improves user experience. It also helps manage load spikes on the

central asset generation mechanism when many users around the globe try to request the same new asset simultaneously.

The CGN differs from a CDN in that the CGN provides generation in addition to distribution. The CGN is a breakthrough technology that, for the first time, applies the principles of a CDN to dynamic content, decentralizing its generation, and ensuring its consumers will be ability to obtain custom-generated assets as quickly as possible from a server in close geographic proximity to their computer.

The CGN by design itself CDN-compatible, meaning that the CGN itself can be wrapped with a CDN for even more robustness, with the CGN's role to provide a global network of origin servers to the CDN. This is an additional level of scalability that we can foresee adding to our network.

The reverse proxy nature of CDNs allows them to function as load balancers and distribute traffic to multiple CGN origin servers, all while controlling the flow of incoming traffic to maximize performance and reduce server load. Because of their on-edge positioning, CDN servers have visibility into incoming traffic, enabling them to employ application layer load balancing algorithms to improve traffic distribution efficiency by precisely gauging the actual load on each CGN origin server.

The pairing of CDN with CGN provides the best of both world, with the CDN providing global generation of unique, dynamic content and the CDN providing global caching and delivery of that underlying dynamic content.

SUMMARY

In the following sections, we present an alternative solution for the necessary operation of web-based services that is powerful, flexible, scalable, inexpensive, and has a truly use-based cost structure.

Companies that develop and operate client-server Web applications will benefit from the availability of a new class of infrastructure solution that requires little setup and operates itself, letting your company focus on its product, not on deployment and operation of server infrastructure. With the CGN's unique pay-for use model, companies can better manage risk because there are no fixed costs and you only pay for actual use of resources, tying costs directly to the success of your product. By utilizing our innovative solution, paying for unused services will be a thing of the past.

In the next sections, we will explore how the system works.

Global CGN System Technical Discussion

A Global Network of Regional CGN Nodes, Working as One

The CGN is a global association of independently operating regional CGN nodes, each delivering computational power through use of its own pool of local, on-reserve worker node machines, each contributed to the CGN by a user wishing to offer their extra CPU power to earn CPUcoin in exchange. The CGN operates as a worldwide, decentralized, scalable distributed server, hosting DServices that power DApps and Web-based applications.

Each regional CGN node participates in the global CGN, with each operating independently in its specific geographic region. A given DApp or Web-based application may connect to any regional CGN node(s) of its choosing, typically preferring ones in relatively close geographic proximity for low latency. There is no practical limit to the number of regional CGN nodes that can be in simultaneous operation worldwide. Each one adds the substantial power of its own local worker pool to the global CGN's total capacity.

Working with the CGN

When developing a DApp or other Web-based application that uses the CGN to make use of a DService, there are several additional services that the CGN will provide for use by the DApp, such as services for locating and selecting an appropriate regional CGN node (more on this later).

We will provide an SDK and documentation for DService and DApp developers that clearly explains these services along with sample code illustrating key use cases. This, together with a downloadable test CGN environment that any developer can run locally, will help developers adopt the CGN by letting them create, deploy and test working DApps and DServices independently in their own fully-functioning local sandboxed environment.

Selecting a Best-Match Regional CGN

A functioning DApp must be able to decide which regional CGN to use for performing the work of its DService. We will provide a service that accepts the geolocation of the DApp, together with a list of the DService components it wishes to use, returning a list of nearby CGN nodes that have the needed capacity and can fulfill the DApp's needs.

A similar mechanism is used to help a Miner decide which regional CGN node to join for contributing CPU power to do work. In that case, the list is biased toward CGN nodes in need of capacity and the specific DServices offered by the Miner.

Other factors that can be taken into consideration when selecting a CGN node:

- Restrictions imposed by the user of a DApp around data flow through certain specific geographic regions
- Ensuring DApps operating at high volume select a CGN with enough resources to accommodate the workload
- Support distribution of work across multiple nearby CGN nodes when applicable
- Direct work away from any regional CGN node that is temporarily down or is experiencing heavy load to ones that will accommodate the work

CGN Resource Descriptor Registries

CPUcoin will create and operate registries to be by CGN components to retrieve descriptors for specific resources of the CGN. For example:

- A registry of DService descriptor records defining the characteristics of all supported DServices
- A registry describing all operating regional CGN nodes. Each CGN node descriptor will include the account ID of its operator, its physical location and its Web address

Regional CGN Node Overview

Purpose of a CGN Node

The Content Generation Network (CGN) is a global, decentralized network of many CGN master nodes (or just "CGN Nodes"), each of which is responsible for receiving, assigning, executing and returning work results for a DApp within the specific geographic region surrounding that CGN Node. These initially will be operated by our company using small amounts of data center resources leased around the globe, enabling us to quickly offer DApps worldwide access to low latency work execution.

In the future, we may delegate the operation of certain CGN Nodes by allowing them to be run within data center space of enterprises and trusted business entities which whom we have mutually entered into a CGN operating agreement. This will open access to geographic regions strategic to our user base and will offer enterprise users the ability to run their own private, internal CGN Node.

Some of the critical housekeeping functions performed by the CGN are:

- Self-monitoring and reporting of each CGN node's operational status to a central service for aggregated system health monitoring
- Accepting and characterizing newly joined contributed worker nodes (Miners)
- Authentication, formation and management of a session, and providing public information about all associated CPUcoin accounts wallets (wallet private keys are never shared with the CGN)
- Assigning work coming from DApps to an appropriate worker node that has the needed DService
- Handling per-use billing, calculating the fee structure for each request and generating microtransactions
- Error handling and recovery while doing work, including automatic failover from a down Miner to a functioning one
- (In our lowest Miner Tier) performing verification of work results to identify, punish and eliminate bad actors trying to submit fake work into the system, and then reissuing the work to another Miner

Support for Many DServices

It is the responsibility of the CGN to quickly fulfill work requests by assigning incoming work to a Miner with the necessary DService implementation. Our intent is to support numerous future DApp use cases by introducing a wide spectrum of new DServices into the ecosystem over time.

Each DService will be deployed to Miner installations based on the operator's choice of which DServices to operate, which may be guided by the available types provided by the host computer or device.

The CGN itself will be inherently agnostic to the specifics of any one DService. The only requirement is for work requests to be representable using an HTTP-based request/response protocol. Initial support will be for work delivered using HTTP, but we will consider adding other protocols as use cases arise.

Initially, we will play the exclusive role of being the central authority for approval of new DServices for deployment to the CGN. This is so we can prevent bad and unknown actors from entering the system, and guarantee stability of the system as adoption scales up during the early growth stages.

To emphasize our focus on delivering a practical, useful product, we have initially developed the first CGN DService, our MediaGen content generator. This DService support an existing enterprise application that has thousands of practical use cases by providing support for dynamic imaging, mobilization and on-the-fly rendering of multi-page documents such as PDF's and much more.

We will also provide tools, environments and documentation to make it easy for developers to try out the system and develop new DServices. We stand behind the importance of our role in driving adoption of the CGN and enabling future opportunities to expand into similar businesses. We will promote the ecosystem by doing our part to help others to use it successfully.

Diagram - CGN Node Detail

CGN Regional Node Detail

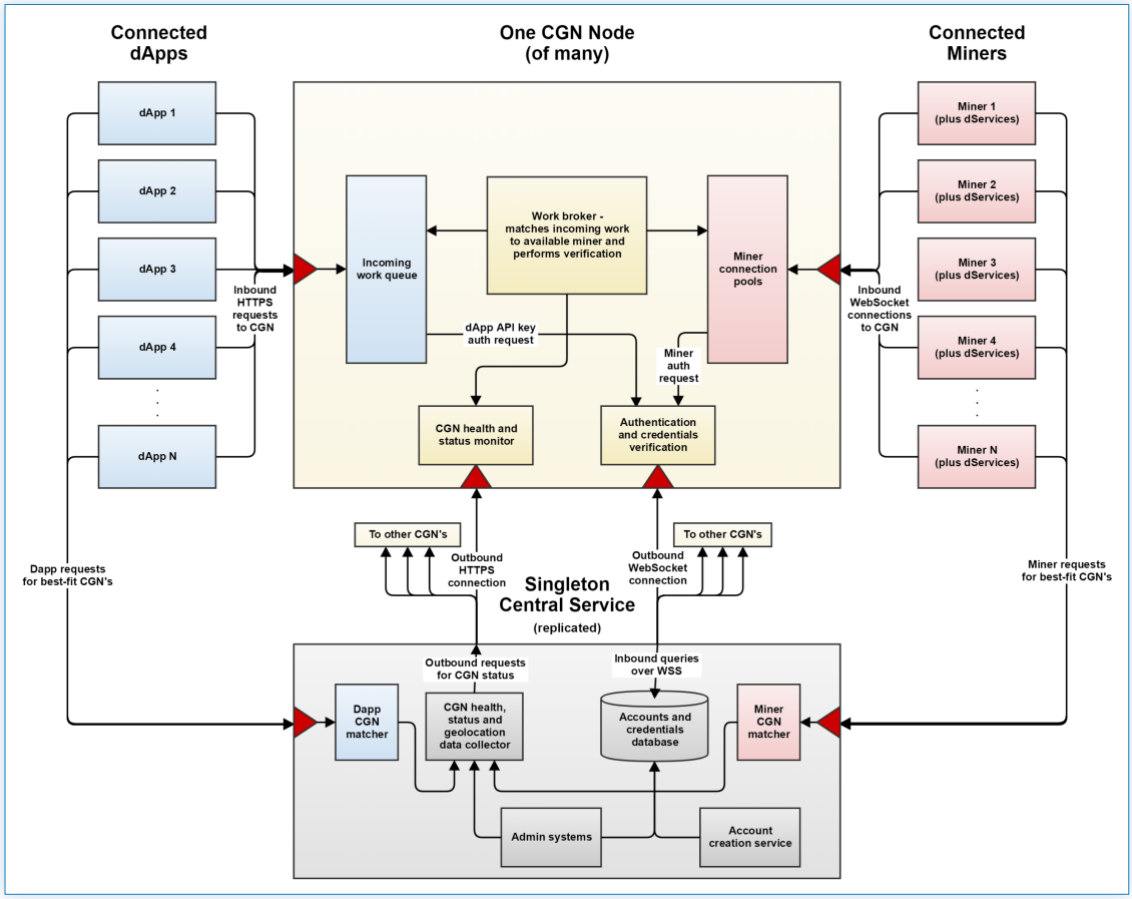


Figure 10

The above diagram describes the inner workings of the CGN in some detail. Please refer to this diagram while reading the following sections.

You can also view a larger copy of it online using this link: CGN Architecture Diagram.

Regional CGN Node Technical Discussion

Anatomy of a CGN Node

The CGN Node is a highly scalable, multi-threaded enterprise-grade server that can be run using one or more local instances (i.e. load balanced). The server exposes four endpoints, each accepting incoming connections on different ports:

1. The Miner connection endpoint (port 8080, using WSS, secure WebSockets protocol)
2. The DApp work request/response endpoint (port 80, using HTTPS/REST request/response pairs)
3. The CGN node status endpoint (port 5551, using HTTPS/JSON)
4. The Central Service authentication reverse-connect endpoint (port 7771, using WSS)

Each endpoint is an independent, highly scalable and high-availability Web server with a single purpose. In addition, there is a Work Broker component that serves to tie together the first two endpoints. Here is how the pieces fit together.

Miner Connection Endpoint

The miner connection endpoint accepts inbound connections from Miner instances that wish to join the CGN to receive and execute work. The secure protocol used is WSS, or WebSockets over HTTPS, with all data sent and received being encrypted on the wire.

The primary function of this endpoint is to receive an incoming connection from a Miner that wants to join the network, to authenticate the Miner against our account records, and to either accept or reject the miner. If rejected, the connection is immediately closed. If accepted, the WebSocket connection is left open as a semi-permanent two-way communication tunnel between the CGN and Miner. This open connection is immediately placed into one of several pools of available miners, organized by Miner tier. The Miner is now available to receive and execute work from the CGN.

Work Request Endpoint

The work request endpoint accepts secure inbound connections using the HTTPS protocol over port 80, with all data sent and received encrypted on the wire.

The purpose of this endpoint is to receive incoming DApp work requests and put them into a fast, in-memory work queue for processing. This is so work arriving faster than it can be processed during bursts of activity will always immediately be accepted. Work is queued by Miner tier (more on this later) so that each tier of service has its own dedicated processor. Work is taken from the queue in first come, first served order and assigned to an appropriate miner from the given tier. After the work is complete the CGN immediately forwards the response to the DApp, which receives the results response and can then go about its business.

When there is a surplus of workers compared to incoming work, the work queue will essentially always be empty. This is because the moment a work unit is put into the queue, if an available Miner is known to be waiting for work another thread will immediately pull it out the work unit and assign it to that Miner. However, there will be a certain threshold to the queue depth indicating that work has been coming in faster than can be executed for too long. If a danger threshold is reached, then we may handle the problem in one of two ways:

1. The CGN may return a 503 (server busy, try again later) to inform the DApp that it should select a different CGN.
2. The CGN may automatically delegate unfulfillable work requests to another CGN known to have available resources, doing so until Miner capacity has been restored. Such forwarding requests would contain an identifier of the unavailable CGN node so that another CGN node will know not to re-

forward the request inappropriately.

Before sending work requests into the system, the DApp must first authenticate with the CGN to receive a session API key. All work requests require a valid API key, otherwise the request will be rejected. The API key identifies the account and tells the CGN who to bill for all completed work. It will not be possible for users without a valid enterprise account to send work to the CGN and all spurious requests will be instantly rejected.

Restricting access to the work request endpoint also helps make the system resilient to DDOS attacks. We are currently researching additional strategies and solutions to further mitigate this.

Work Broker

A central process called the Work Broker connects each incoming work request with an appropriate available Miner and then sends a command to the Miner in reverse through the WebSocket tunnel to execute the work, and then receives the work result from the Miner back through the same WebSocket link. The work broker then decides whether the work was in fact executed and is also correct.

If the Miner can't or doesn't immediately acknowledge receipt of the work, it is considered to be down. The down Miner's connection is immediately terminated and is removed from the available miner pool. The Miner is free to reconnect later once it is ready.

In our lowest Miner Tier, where a trusted execution environment is not available, an additional consideration will be accounted for: If the Miner is found to be have done wrong work (discussed in detail later), then the Miner connection is terminated as above. Additionally, the Miner's account is flagged, and the Miner will not be allowed to reconnect. Finally, the CGN seizes ("slashes") the Miner's at-stake holdings of CPUcoin, and the Miner is left with a zeroed out at-stake balance.

In the minority cases where a Miner fails to complete the work, the incomplete or wrong work is discarded, and the original work request is put back at or near the head of the queue, to be picked up immediately by another available miner. In this way the original DApp work request never results in an error response due to problems with a Miner. The DApp will always receive work results (which may contain an error response if the DService returned an error), but they may be delayed if there were Miner availability issues during processing.

After the work unit has been completed the Work Broker simply sends the results back to the DApp in response to the original request. The DApp will have no knowledge of how the work was completed.

The miner client will have measured total resource utilization while the work was being done, and those results are sent back to the CGN along with the work result. The CGN also performs its own measurement, allowing it to compute latency by subtracting time reported by the Miner.

The work broker finishes by immediately writing a CPU/resource utilization record together with a record of the completed work unit to a high-throughput time-series database that ingests and records all micropayment records for later processing. We use these records at the end of each billing cycle by totaling transactions to date and using the totals to transact Miner payouts and bill DApps for usage of the service. Billing cycle payment totals are transacted in CPUcoin and are permanently recorded on the blockchain (more on this later).

Status Endpoint

The status endpoint accepts inbound secure HTTPS requests to the CGN Node allowing DApps and Miners to make inquiries about the CGN Node itself. Inquiries may include requesting the specific geolocation of the CGN node, information about its current health and more (TBD). This lets our Central Service consider and compare multiple CGN Nodes, allowing it to help DApps and Miners to make a decision about which CGN Node to connect to using factors deemed most relevant.

Central Endpoint

The central endpoint allows our Central Service to connect inbound to each CGN node to make itself available to provide secure account lookup and credentials verification services needed by the CGN.

Central Service Singleton

The Central Service single instance of a scalable, distributed centralized server using a private database, operated exclusively by CPUcoin headquarters. It establishes an outbound connection to every CGN (using the Central Service endpoint) to provide authentication services to all registered CGN nodes. The Central Service is designed to never accept inbound connections for authentication. Instead, it uses our global directory of CGN instances to establish an outbound connection to each CGN node using the secure WSS WebSockets protocol.

The Central Service utilizes a private database also operated by CPUcoin headquarters. It stores account records, hashed credentials and CGN component certificates so Central Service can provide authentication information to CGN nodes and supports our self-service user account management pages. This private database is locked down and firewalled off from the internet, available exclusively to the Central Service and authorized internal database administrators.

Because the Central Service is configured to not accept any inbound authentication connections, it will not be possible for any actor, malicious or otherwise, to reach the private database or access our authentication services. The only way these can be used is through a CGN node. This strategy also makes the Central Service resilient to DDOS attacks.

To reduce the chance of this system becoming a single point of failure, we will use a replication strategy to operate multiple, synchronized copies of this service operating in different locations.

Anatomy of a Miner

The miner client consists of two components, both to be installed on the user's device:

1. A DServices manager runtime, preconfigured with our CGN client software including one or more DService implementations, to be installed into a Trusted Execution Environment, and made available to the CGN for performing specific types of work. The specific list of available DServices will depend upon how the Miner is configured and the platform itself. Not all DServices need be installed into every Miner. The Miner will be able to offload unwanted DServices and/or acquire new ones, providing control over the selection of work the Miner is willing to do which may change over time.
2. The Miner control service, a small background program for the end user to control the Miner VM such as starting, stopping and shutting down the VM. It will also provide a simple configuration for the Miner, such as control over conditions establishing when the Miner will be allowed to do work. It will also provide the end user a means to visualize Miner status information including the amount of work done and number of CPUcoins earned.

The specific tier of Miner, taken together with requirements imposed by Specific DServices and the nature of the platform itself will determine the way in which a Trusted Execution Environment is to be established (more on this later). Therefore, in cases where privacy is guaranteed through restricted physical access, the TEE may be considered sufficiently trusted even with the Miner runtime and DServices being directly installed into the host environment ("bare metal"). In cases where privacy cannot be assured by physical isolation of the device from bad actors, a TEE must be provided on the device using a virtualized environment with an unbroken chain of trust down to the hardware security module (more on this later, too).

DService Segregation within the Miner

Each DService will be segregated from the others so no DService can access any files or data associated with other DServices. This will be done through containerization of DServices (more on DServices later).

TRUSTED COMPUTING

INTRODUCTION

This section is about mitigating the risk of malicious actors being able to abuse the CGN protocol to "earn" CPUcoin by creating a Miner client that can fool the CGN with cheaply generated fake work. This section pertains to our tier of miners that includes consumer devices of all kinds, where a trusted execution environment can't be assured through physical isolation from malicious actors.

Currently it is not known how to prove that a given installation of software has provided correct results to a general computing problem when there is any possibility of the software having been tampered with by a malicious actor. Altered or "fake" software implementations can be written that intentionally produce wrong results.

There are various strategies that use consensus to try to solve this problem by issuing the same work to three or more Miner workers and comparing the results. If there is any inconsistency, usually it can be concluded that the minority outcome is the bad work. But there's no way to guarantee that. If there are many bad actors in the system producing fake work the same way, what's to stop that bad actor from operating many Miners, enough to become the majority work generator when multiple results are compared? The number of Miner workers used to confirm work results can be increased, but the problem quickly devolves into the Byzantine Generals problem, whose solution reaches consensus only if there are at least 3 times as many good actors as bad actors.

This problem is solved well by blockchain technology, but at high cost of essentially every actor in the system having to duplicate the same work. It quickly becomes apparent that this sort of strategy cannot efficiently be used by the CGN to prove work is correct. *Any strategy that does not massively duplicate work cannot be guaranteed to generate correct work in all circumstances.* For a solution geared toward the enterprise, this would be unacceptable.

Rather than pursuing as strategy to detect and correct the fake work that might be generated by altered software, we will instead focus on the task of ensuring only unaltered software can ever successfully connect to the CGN in the first place.

SECURITY HARDWARE

Today's modern servers, mobile devices and some PC's are being built with a special, built-in hardware security device called the TPM (Trusted Platform Module). It supports data encryption, certificate management, and an has ability to establish a chain of trust within the platform. This chain of trust is starts with the installed software, extends through the operating system, the boot loader and the BIOS, and is finally rooted in the actual TPM chip itself, whose contents are completely inaccessible to the end user and cannot be altered or tampered with.

The TPM stores a completely inaccessible private key, which it uses to sign critical data such as hashes of the environment and its installed software. By use of the TPM's public key and knowledge of the good hashes, it is possible for the TPN to testify to a remote server that the environment is completely correct. This is done by computing a hash of the environment and attaching its cryptographic signature to a message containing that data. The signature can be remotely verified as authentic using the TPM's public key then verifying the hash against a set of known correct, unaltered environment signatures.

REMOTE ATTESTATION

The goal of attestation is to prove to a remote party that an operating system and its application software are intact and trustworthy. Using this technology, a remote server can prove that a given machine has precisely the expected software, operating system components, boot loader and BIOS that it requires, and that no unexpected software is present. If anything has been tampered with or altered, the hash will no longer match and the TPM will not sign the message. With the chain of trust broken, remote attestation will fail. Since

configuration of the environment is under our control, we would only need to keep a database of up to date hashes for each known good environment signature.

To apply this to the CGN, at the time a Miner connects to the CGN for authentication, the protocol would first require the Miner to give the TPM-signed system hash to the CGN for verification. The CGN would then compare the incoming environment hash against our known good environment hashes to confirm it's trusted. No adversary could possibly forge such information because it is impossible for any entity to sign any fake system hash as the TPM.

SOLUTIONS

Many modern mobile devices are designed with a Trusted Execution Environment already built in. Our Miner client software can be installed as trusted app on these devices. The code there cannot be tampered with by a malicious actor.

Trusted computing on PC's and servers usually requires both a TPM and a bare metal OS and software installation, where the entire environment is under management of the TPM. This presents a challenge for us, because we will have no knowledge of the other valid software installed into the environment, so we would not be able to confirm the environment against a set of specific valid environment hashes because the number of possible environments is vast.

With recent innovations in virtualization, there are now VM implementations offering support for a virtualized TPM that is also tied to the hardware security module. This lets the VM itself establish a chain of trust from the software down to the virtual BIOS as if it were a bare metal system. This is currently supported by Windows Server with Hyper-V, which we could use for as our Windows virtualization environment. We are continuing to research a similar solution for Linux. This type of solution could enable us to run the Miner client within a virtualized environment using the best OS for suited for the chosen DServices, as well as providing the needed TEE and attestation.

Trusted computing and remote attestation of work are hot areas of research now, and new solutions are continuing to emerge. We intend to utilize the best solution available for each class of device to ensure the integrity of our system cannot be compromised maliciously.

DATA SECURITY

Data security is of the utmost importance to us, and we must take every measure to ensure DApp work data is kept safe in each stage as work makes its way through the CGN to Miners and back to the end user. Much of this security will be provided by use of standard, secure protocols.

Part of our security solution relies on the existence of a Trusted Execution Environment, whether it be by physical isolation from attackers or through use of a TPM. With these security concerns already addressed, we will focus here on strategies to ensure data can't easily be examined on disk or in memory from outside the trusted environment.

The strategy expressed here helps form a basis for how we will designate Miner tiers, which is discussed in the next section.

DATA AT REST

I most cases the Miner client will usually be run within a minimal, lightweight Linux or Windows virtual machine that is factory configured as an opaque "black box" to prevent attempts to gain root access to the VM:

1. The Miner VM will be pre-configured by default with all inbound ports closed except one: there will be a single open inbound "control port" for telling the miner to start, stop or shut down, and for querying its current health and use statistics. Therefore, it will be impossible to establish any kind of OS-level control connection from the host environment into the VM to exploit the known vulnerabilities of general-purpose services or otherwise try to get shell access into the VM.
2. Inside the Miner VM we disable or physically remove any unnecessary administrative tools, services and programs/modules/features not essential to the operation of the core OS for the Miner client and its DServices.
3. To prevent data at rest in the Miner VM from being read through the host file system, in cases where the TEE does not provide encryption, the Miner VM environment will itself use robust block-based encryption such as `dm-crypt` with `cryptmount`. This is supported on both Windows and Linux. For the utmost in security without vulnerabilities, we will use an operating system image that supports `ESSIV`, `LRW` and/or `XTS`. Any attempt to examine file content in the VM image data file on the host will produce only random data.
4. We will use a containerization strategy (e.g. Docker) within the VM to isolate each DService from the others and allow each DService to have its own apparent file system for storage of temporary results. The DService implementations will not be able to observe each other's code or data. This will also prevent conflicts from arising around shared resources such as ports and I/O.
5. In addition, we can overwrite deleted temporary files containing previously completed work with random data. This will minimize what can be seen by a byzantine actor inspecting disk if the file system encryption were to be broken.
6. Similarly, we can overwrite critical memory with random data before freeing it. This will minimize what a byzantine actor can see when inspecting memory in the host environment with the VM running.
7. CPUcoin (DServices Ltd.) controls what is permitted to run in the Miner client runtime environment. We will certify all new or updated DServices before deployment to the network. While we will strive to minimize barriers to adoption, we do not intend to build an open system where anybody can deploy a DService to the network anytime they want. This will eliminate the chance of uncontrolled introduction of unwanted DServices, malicious or otherwise.
8. Each deployed DService must be signed by a trusted signing authority. The Miner client will not launch any DService whose signature is invalid.
9. The VM will be able to update its software installation from our servers as needed to incorporate new features and bug fixes.

Data In Transit

1. The CGN only accepts inbound connections using HTTPS and WSS. A certificate containing the key pair will be installed into the CGN at time of setup and will not be accessible from the outside. We will ensure that CGN certificates are updated before they expire according to best practice.
2. Only CPUcoin and our trusted designees will be able to operate a CGN node. Public users will not be able to operate a working CGN, even if our project were completely open source. The list of known, genuine CGN nodes is kept in our Central Service, and this service only makes outbound connections to known CGN nodes. Any other CGN node, legitimate or otherwise, would not be able to gain access to the credentials database, nor will it be advertised by the network to DApps and Miners as being an available option. Neither DApps nor Miners could discover a fake CGN, and it would receive no traffic.
3. In addition to closing inbound ports on the Miner VM, the local network contained within it will be configured as unreachable from outside the VM. Inbound traffic to the VM from the host environment will be blocked and only outbound connections will occur.

By using these strategies, a potential malicious actor will not be able to inspect network traffic, disk or memory, nor attempt to operate an unregistered or fake CGN node.

TIERED MINER STRATEGY

We want to enable a wide range of participants to be able to join the network as Miners, recruiting a blend of underutilized CPU resources coming from a variety of sources.

Miner Tier Properties				
Tier	Location	TEE?	KYC?	Use Case
Tier-1	Corporate (any)	yes	yes	Private computing (including clusters)
Tier-2	Corporate, data center or bulk hardware	yes	yes	High performance, high capacity computing (including clusters)
Tier-3	Consumer (mostly mobile)	yes	yes	One-off enterprise bulk work where security, privacy and correctness are paramount
Tier-4	Consumer (all devices)	no	optional	One-off consumer space bulk work

In Tiers 1 through 3, byzantine behavior will not be possible. Tier 4 opens participation of the system to everybody.

The type or tier of a given Miner will be recorded in the Miner's account record in our central database. The CGN will organize Miner instances by tier at the time the Miner connects to authenticate, when the CGN looks up the Miner account record.

Each DApp will be able to establish in advance which tier(s) of Miner it wants to work with, and the CGN will respect that preference. In this way, enterprise DApp users can make tradeoffs between the type or grade of execution environment against cost of the service.

All tiers of Miner Terms will require the end user or operator of the Miner to agree to a contract stating the terms and conditions of participation in the service. This will include provisions to forbid tampering with the data of the service, decompiling or reverse-engineering the service, and fraudulently interacting with any of the API's. Violation will be enforceable through legal action.

We envision 4 categories of miner type, or tiers, as follows:

TIER-1: ENTERPRISE-OWNED RESOURCES

This is our highest Miner tier. Tier-1 miners utilize spare capacity provided by the DApp enterprise *itself and most traffic is limited to and executed on the enterprises network exclusively*.

With this tier, the DApp operator can use its own spare compute power to operate Miners running its DService. This can be spare data center space, but that is not a requirement. In-house servers or employee PC's could be used just as well.

Because all Miners would be located within the confines of the enterprise business, matters around securing the Miner from data inspection and tampering are kept completely under the control of the enterprise itself.

This tier could be also be utilized by the enterprise as part of an internal pilot program to try out the network.

Lastly, we will provide a toolkit for DService developers consisting of a downloadable development build of the CGN along with an empty Miner VM that can be deployed and run locally. The developer just needs to drop their own DService into the VM. Developers would operate the components temporarily while testing and debugging their DService, safely and conveniently within their own private network. This toolkit use case also falls under Tier-1.

Tier-2: Data Center Resources

Our intermediate tier. Tier-2 miners utilize spare data center grade capacity provided by a variety of sources including data centers, server farms and embedded systems. Tier-2 resources generally originate from spare or standby server capacity businesses seek to monetize but may also include embedded systems and other sources of bulk CPU power provided by hardware that is not directly accessible to end users.

These Miners are provided to our network in bulk through operating agreements made with each resource providing entity. These entities will be required to go through a KYC process with us, so we know who to hold accountable.

This class of resource is controlled purely by businesses entities and runs only on company-managed hardware. These Miners must be completely inaccessible to end users.

Tier-3: Trusted Consumer Devices

Our most basic trusted tier. Tier-3 miners utilize spare capacity provided by the general public coming from consumer devices such as PC's, laptops and mobile devices. Only devices with a trusted execution environment will be supported. We are investigating approaches to make best use of trusted computing options available for the most common device types and operating systems in this category.

This is the lowest tier that can accept enterprise grade work.

Tier-4: Untrusted Consumer Devices

Our most basic tier, one which lets the public at large participate in our ecosystem with the least friction. Tier-4 miners use spare capacity provided by the general public coming from consumer devices such as PC's, laptops and mobile devices. This tier will primarily contribute GPU, as those types of workload tend to be easier to verify.

Tier-4 miners are the only tier where malicious behavior is possible. However, there are protections we will employ to reduce the risks. These users will first have to create an account with us, including verifying email address and completing of a Captcha test, to block automated account creation bots.

Miners in this tier will receive consumer-grade work, which is also the most tolerant of the low, but nonzero, probability of receiving incorrect results. Examples of this workload include batch processing of images/videos and other high-volume tasks that produce public content, as well as general-purpose GPU computing tasks. There is a large market for this type of work.

Work results produced in this tier will require verification, meaning the system needs to consume additional resources to perform verification. That means payout for resources will be smaller in this tier than the other tiers.

There are two variants of Tier 4, as follows:

Tier 4a

We will require these users to complete a KYC process that will link an actual, known person to the work being done.

Tier 4b

This is the case where a Tier-4a user opts out of KYC. These users may simply not want their activities or earnings associated and tracked on our network; these are potentially the most byzantine of Miners because the people offering these resources will not be known to us.

We believe the above measures, combined with our ability to slash at-stake funds, will present enough obstacles to discourage and effectively punish malicious behavior. While such behavior can be greatly suppressed and virtually eliminated, even using all these strategies it is not possible to 100% guarantee that byzantine behavior will always be detected and corrected.

Ensuring Work Correctness (Tier-4)

The Problem

Our aim is to build a global network that is strong, flexible and secure. We want to support as many DApp use cases as possible.

Our service is targeted at enterprise users, so we anticipate that paying users of our service will be businesses of various kinds. We expect universal agreement around wrong work results. All users of our system would consider that to be very undesirable.

But different use cases will be more or less tolerant to an occasionally wrong work result. These examples illustrate the spectrum of tolerance for occasionally incorrect work results:

- Not at all tolerant: Pure enterprise use cases such as processing sensitive customer and financial data within a banking institution.
- Somewhat tolerant: Consumer use cases such as generating thumbnail images for websites or content for consumer mobile apps. An occasional bad thumbnail is undesirable, but the impact is usually low.

As explained, currently, there seems to be no solution for the problem of zero-knowledge proofs for general computation using byzantine workers for general purpose computing. That's why we have focused on running the Miner software in a trusted execution environment, as explained for Miner Tiers 1-3. But because there are many use cases where this constraint has been relaxed, we have Tier 4 Miners to accept that type of work.

Here are some strategies to minimize the chance of incorrect work being returned to a DApp as much as possible.

Work Correctness Strategies

Here are some strategies we may use to ensure work correctness:

Avoid Using Miner Tier-4

With miner tiers 1-3, the miner VM and its DServices run in a trusted execution environment, either by physical isolation from malicious actors or through use of technology based on a TPM. DApps that cannot accept any risk whatsoever of incorrect work results should avoid use of Tier-4 Miners altogether.

In the next sections, we will discuss strategies we'll use to minimize the possibility of fake work being submitted by byzantine Tier-4 Miners.

Use DServices with a Work Verifier

In certain situations, it may be possible for a DService developer to also implement a verifier. The verifier is a companion to the DService that can examine work inputs and outputs and determine whether or not a work result is authentic. The idea is that verifying a work result must be much less expensive than generating the work result from scratch. The verifier may not need to verify all the work and it may be enough to only verify parts of the work, or only certain aspects of the work.

For DServices that have a verifier, the CGN may select two or more random Miners (as specified in the enterprise account's work verification preferences) to verify each work result. Work results failing verification will be discarded, a different miner will be chosen to redo the work, and the bad miner's account will be flagged, preventing that account from doing further harm.

Handle DServices without a Work Verifier

When it's not possible to verify work due to the lack of a DService verifier, the CGN may instead issue identical work to three or more miners at once, and then check all the results for agreement. If there is a discrepancy, the CGN will try to determine if any of the work results are valid. *The CGN will determine*

which miner was at fault by comparing the differing results against an internal "reference miner". This is a small set of trusted Miner instances running within the CGN that will always do correct work. They are used only as needed to resolve disputes.

This will be a difficult system to break because the byzantine actor would need to dominate the Miner pool of the CGN with identical fake miners for there to be a high probability that the CGN would select three Miners at random that all do the same incorrect work. We have created impediments around account creation and the use of at-stake funds that would make this technically and cost-prohibitive to set up. So long as there are any adulterated miners in the network the CGN will catch them sooner or later and terminate them.

Attempts to Hack the Environment

As we have explained, we will make it very difficult for anybody to alter or inspect the VM environment:

1. The CGN will be completely opaque to the outside. The only place where an attacker can attempt to observe or alter the system is on a Miner host machine, but everything of interest takes place within the VM and is made very difficult to observe.
2. The Miner client communicates with the CGN using an encrypted channel, so work and work results cannot be inspected on the wire.
3. The Miner VM uses disk encryption and its contents cannot be inspected either.
4. Used memory will be overwritten ASAP, so inspecting memory looking for useful information will be difficult.

Attempts to Hack the CGN Protocol

With Miner Tiers 1-3, the CGN protocol cannot be hacked because either the Miner is physically isolated from malicious actors or because the CGN will require of a TPM-signed hash of the environment, and this cannot be faked.

With Tier-4 Miners, the CGN protocol can also potentially be hacked directly without using the Miner VM at all. We believe with all the measures we have presented will make this both *technically and financially unfeasible and not worth doing*.

To illustrate, here is what the attacker would have to do in order to beat the system and make it submit fake work, and the outcome of that:

1. Register a Tier-4 Miner account with the network.
2. Disable/ignore the Miner VM and its control service.
3. Develop a custom program (a "fake Miner") that attempts to establish a connection to the CGN. This would be easy to do if we open-source the CGN as planned. *Security must never depend on the obscurity of an implementation because people can always figure it out.*
4. Unless this fake Miner were willing to put some funds at stake, it would only be able to receive the lowest grades of Tier-4 work that doesn't pay as well as higher grades and which is tolerant of occasionally incorrect results (such as generation of bulk public content).
5. Even so, this fake Miner would very quickly if not immediately be spotted by the CGN as having produced incorrect work. The CGN would confirm this using additional miners (including a reference miner) to verify the work. Once the fake Miner has been caught in the act of forging work, its account would be flagged, and that Miner would no longer be allowed to connect to the CGN. *The entire effort to this point will have been thwarted, and the attacker will be back to square one.*

Data Set Management

Data Transfer Strategies

Node: The term HTTP is used here to refer to the protocol in general. All HTTP communication will be encrypted using HTTPS.

The work input and output data can be exchanged between the DApp Miner in at least two different ways:

1. Data exchange through HTTP request/response to the CGN

   ### *Sending Input Data*

   The DApp may issue an HTTP POST or PUT request that in addition to specifying the nature of the work, may also include an input data file to the DService. The CGN will quickly route the request as-is to the Miner, with the data passing through the CGN. The CGN handles data as efficiently as possible by streaming, and large file sizes up to 2 GB will be possible.

   ### Receiving Work Results

   The work results data may be returned similarly through the HTTP response by encoding the results as a document or .zip file and issuing a HTTP response that transmits the file.

2. Data exchange through a 3rd party storage location

   ### *Sending Input Data*

   The DApp can first publish the work input data to an external file system that could be hosted on a cloud service such as AWS, and then pass the URL of the file location in the HTTP request instead of the data itself. The CGN will route this to the Miner and then to the DService, which would then in turn fetch the data set from the cloud and store it locally on the miner for processing. If the Miner is part of a cluster, all Miners in the cluster would then have access to the input data, which can be shared.

   ### Receiving Work Results

   The DService can then publish the results back to the cloud in a work results file, returning its location in the HTTP response back to the DApp. The DApp could then go read the results.

Note that the two approaches can also be combined.

For use cases where computation of the data takes much longer than transferring it, either solution may be efficient enough.

Data Transfer Speeds

This table shows ideal download times for various file sizes over a range of internet connection speeds. It can be used to estimate the time cost of transferring data sets to and from the Miner.

Upload/download times for various file sizes and transfer speeds:

	Internet Connection Speed		
File Size	10 Mbps	100 Mbps	1 Gbps
1 MB	1 sec	100 msec	10 msec
10 MB	10 sec	1 sec	100 msec
100 MB	1 min 20 sec	8 sec	1 sec
1 GB	13 min 30 sec	1 min 20 sec	10 sec

Tier-3 and Tier-4 consumer grade Miners can be expected to have minimum working Internet speeds of at least 10 Mbps, as we will impose reasonable minimum requirements around availability of critical resources. Many miners in this tier, especially ones located in metropolitan areas offering high home internet speeds, will be able to deliver 100- 500 Mbps speeds down (typically 50 Mbps over Wi-Fi), and 10- 50 Mbps up.

Tier-2 and Tier-1 Miners will generally have very fast connections to the internet, often symmetrical and usually exceeding 300 Mbps and even 1 Gbps or more. Transferring large files, even up to a Gigabyte in size, will only take seconds.

File Sharing and Exchange for DService Instances

We realize there are many use cases where a DService might need to with a growing set of cached files or intermediate results to be saved and reused for more efficient subsequent computation. There are many ways to accomplish this and this is an area where we are currently researching solutions. Here is some thinking we currently favor, with our intent to revisit this again when the project is further along. These capabilities would initially only be available with Tier-1 and Tier-2 Miners.

Miner Affinity (All Tiers)

Miner affinity is the ability for the DService to make a series of requests to the CGN and have them all be received by the same Miner instance. With Miner affinity, it will be efficient to perform a series of computations on data where parts of the data set don't change and can be reused between requests.

One use case for this is a batch job where the DApp wants to add a logo or overlay to many different documents. If the constant data were loaded only on the first request, subsequent requests would only need to perform the overlay step without re-downloading the common assets.

Another use case is when work units are serialized such that some of the inputs for future work are specifically the outputs of prior work already completed. With Miner affinity, each new work request would continue to have access to all the previously generated work inputs or outputs still present in the file system without transferring files back and forth.

Miner Clustering for Resource Sharing (Tiers 1 and 2)

To support Miner clustering, we will provide a means for multiple Miners housed within the same shared facility to be configured together with a shared file system. In this configuration, the Miner cluster would behave as a single Miner with multiple instances, with each miner instance able to access shared input and output data generated or downloaded by any of the other Miners. The shared file system will be organized by DService so each DService can only see its own portion of the shared file system.

With this setup, the DApp would be able to reserve all or a portion of a cluster. It could use multiple instances to run many large jobs in parallel, with all the shared data only being downloaded once.

We have not yet worked out the specifics around the rules and cost structure of Miner cluster reservation, however, we do have a virtual file system (VFS) that already handles locking and sharing of files between different instances as part of our first DService, which we may leverage for Miner clustering. We consider Miner cluster support to be a firm requirement of the CGN and are still working out the details.

FileCoin, IPFS and Others

These are just a couple of several technologies we are investigating for a potential good fit with our ecosystem. These could be used to provide decentralized file storage for DServices. Today, our VFS handles secure read/write to Azure and AWS shared storage and data lake models, and is compatible with virtually any CDN, including Akamai and others. We will provide more specific information when we revisit this area later in the project timeframe.

THE DSERVICE

A DService is the server-side component supports one or more DApps. It is software developed together with the DApp and housed within our CGN Miner client. A DService receives requests made to it in accordance to its own well-defined API, and executes them to perform all forms of specialized, compute-intensive work tailored to the needs of its associated DApp or family of DApps.

The role of a DService is simply to accept inputs, perform work and return the output. There are several protocols that achieve this, but we are focusing at first on protocols based on HTTP such as REST, already commonly in use by microservice-based systems.

A CGN-compatible DService listens on a specific port for HTTP requests, interprets and processes each request and then produces valid HTTP responses. The DApp and DService have full control over the choice of port and URL format, and no specific conventions are imposed by the CGN.

THE DSERVICES MANAGER RUNTIME

This is a software component which we have developed that provides connectivity between DServices and the CGN. It provides the following capabilities:

- Performs a fitness test of the Miner environment for reporting its capabilities to the CGN
- Uses the contributing entity's account and credentials to form a WebSocket tunnel connecting it to the CGN
- Starts a service that waits for work
- Upon receipt of any work, ensures the appropriate DService is running and forwards work to the DService implementation
- Measures resources consumed by performing the work
- Forwards the response back to the CGN together with information about the amount of resources consumed by doing the work, plus any metadata that the DService wishes to record along with the work microtransaction
- Provides a mechanism so a long-running DService request can report regular status updates to the CGN
- Supplies a software update mechanism so new DServices can be deployed to the Miner client and existing DServices can be updated or patched
- Supports error handling and error tracking by returning information about unexpected failures to the CGN for centralized error logging

THE CPUCOIN TOKEN

BASED ON ERC-20 STANDARD

In this section when we use the term "account", we are referring to an Ethereum account, not the associated end user's user account that is also created with us. Ethereum and end user accounts will always have a one-to-one relationship.

The CPUcoin token will be a standard ERC-20 token implementation with a few additional capabilities tailored to our solution. This proposed set of requirements should result in a very simple ERC-20 smart contract with only a few enhancements.

Note that progress in the Ethereum community has come rapidly, and a few new token standards have emerged to supersede ERC-20, some of which have gained wide support. One we are looking at closely is ERC-777, an extension of ERC20. The emerging ERC-777 standard improves on ERC-20 with the addition of operators, new functionality that enhances support for approval of third-party spending. This is a use case the project relies heavily upon.

When we say ERC-20 in this document, that is to be interpreted as ether ERC-20 or a yet-to-be-selected derivative of ERC-20.

Basic behavior (supported by ERC-20 or derivative):

1. CPUcoin accounts created for each ecosystem role will all be the same kind of account, implemented by one family of inherited smart contracts. Each instance of a role must use a unique Ethereum account (no account sharing between roles or instances).
2. Staking of funds will be supported by use of the ERC-20 spend delegation method `approve()`, or by use of operators in ERC-777. Funds delegated by the Miner to the project master account can be withdrawn by us at any time when we must seize a stake. The at stake funds can be viewed using the ERC-20 method `allowance()`. A miner with little or no funds at stake will only be able to get the least profitable work.
3. Enterprise billing will be supported using the same mechanism. We will collect use fees by withdrawing pre-approved funds. We can examine the remaining balance using the ERC-20 method `allowance()`, and automatically handle notification of the enterprise operator (or perform automatic funding of their account) when approved funds are running low.

In addition to the basic ERC-20 (or derivative) implementation:

4. The ecosystem token contract will support the ability to save metadata together with a transaction to permanently link each CPUcoin ecosystem transaction to a hash of all the microtransactions that contributed to it (more on this later).
5. The ecosystem token contract will support the ability to redeem IEO tokens for ecosystem tokens (more on this later). A given number of IEO tokens held in an account can be redeemed simply by calling the `redeem()` method. This will create a single transaction that both mints tokens in the ecosystem token contract and burns tokens in the IEO token contract.
6. The IEO token contract supports custom vesting schedules and automatically releases the vesting tokens over time, a capability ideal for supporting IEO bonus programs as well as employee incentive grants.
7. Both the IEO and ecosystem token contracts will have support for certain contract administrative functionality determined to be needed.

ENTERPRISE ENTITY ACCOUNTS

1. Each user of the GCN will be required to open an Enterprise Account with CPUcoin in in order to access and use DServices. Each Enterprise Account will include the Ethereum address of the Enterprise account. It will be owned and operated by the user for the purpose of paying for services. The private key will never need to be disclosed to anybody.

2. We will help facilitate creation of this Enterprise account on behalf of the Enterprise Account as part of the installation process and for configuring the DApp or other client software. The Enterprise user shall be the owner of this account, as we will not hold any user's private keys.
3. We will provide a self-service account management website as a convenience for Enterprise Account holders to easily inspect and manage their Enterprise accounts, and to support typical business workflows around measuring and tracking use of and payment for services consumed.
4. Under the proposed terms of service for opening an Enterprise Account with CPUcoin (also known herein as the Continuous Service Agreement), each user will authorize CPUcoin to automatically collect pre-approved payment from their Enterprise account following any use of the Service. For maximum efficiency, all collection actions are automated.
5. Where there are insufficient approved funds in any Enterprise account to pay for services consumed, the Enterprise Account holder will, via the terms of the Continuous Service Agreement, also authorize CPUcoin (acting through a licensed third party payment gateway) to debit fiat from a nominated bank account of the user in order to purchase CPUcoin on the open market on their behalf. This would be a standard ecommerce transaction conducted in fiat currency. We will then directly collect these CPUcoin and apply them as payment for services consumed. This will prevent any service interruption due to lack of funds.
6. The user will have the freedom and flexibility to fund their account at the commencement of use of the service, as well as at any time desired going forward. This allows each user the opportunity to purchase CPUcoin in advance at attractive prices for pre-approval, and thereby prevent the automated account top-up from occurring as described above.
7. A detailed record of all transactions will be provided in a monthly statement to the Enterprise Account holder's finance department itemizing all CPUcoin expenditures.

CGN Operator Entity Accounts

An account for receiving CPUcoin proceeds in return for operating a regional CGN, held by its operator. We anticipate that CPUcoin would generally operate CGN nodes, but in the future, we may delegate this responsibility to certain trusted delegates on a per-CGN node basis.

1. This account must be established before deploying each regional CGN node. The private key will be held by the operator of the CGN alone, the CGN itself will not need to know the private key.
2. This account will be used to collect a set percentage of fees provided for operating the CGN.
3. Where CPUcoin is the CGN operator, proceeds are to be used for creating and building demand in the DService marketplace and for promoting mining of the service to gain new contributors.

Developer Entity Accounts

Each Developer Entity will maintain an account to be used for receiving a royalty fee for each use of the DService that it has developed.

1. This account will be set up for each approved Developer Entity as a part of the DService certification and approval process. The private key will be held by the developer alone.
2. For the first DService, MediaGen, this account will be owned by Equilibrium.

Contributing Entity Accounts

Each Contributing Entity (Miner) will have one account for each worker node they operate.

1. This account will be created for the Contributor when they create their Miner account, which is the first step before installing and configure the worker node software on their donated hardware resources. The private key will be held by the contributor alone.
2. The Contributor will be given the opportunity to put at stake in their account a fully refundable reserve of CPUcoin. We do this so that we have recourse should the Contributor become a Bad Actor.
3. We can make this easy for them by letting them purchase the necessary amount of CPUcoin at the time they install and configure the MediaGen worker node software. This would be a standard ecommerce

transaction conducted in fiat currency, at the current exchange rate, to pay for the stake amount in CPUcoin. It could be purchased using a credit card, PayPal, ACH direct bank transfer, etc.

4. The cost of the optional stake will be set in fiat currency so that it does not fluctuate with the price of CPUcoin. The cost is initially planned to be 2 Euro (subject to change). This will be an amount large enough to make swarming attacks not economically feasible, but an amount small enough so as not to pose a barrier to participation.
5. We will provide a self-service account management website to allow Contributors to inspect and manage their CPUcoin accounts, as well as to reclaim the at-stake amounts at-will and refund it to fiat cash.

THE CPUCOIN PAYOUTS ACCOUNT

This is a temporary per-CGN node Ethereum account to be used during the collection/payout process to aggregate related transactions. The CGN itself alone holds the private key to this account. The CGN operator will typically have no need to ever access this account.

This account is used to provide an optimal solution to the problem of M users of the service paying in, with payouts to N providers of the service, where M and N can be numbers in the range of hundreds to tens of thousands. This lets us reduce the number of payments made on the blockchain from order of $O(M * N)$ to $O(M + N)$.

1. This is an account that the CGN alone will use temporarily per billing cycle to automatically collect paid-in proceeds from multiple users of the service for transacting pay-outs to multiple recipient entities. No human should ever touch this account.
2. As a matter of corporate integrity, it is essential that funds in this account never be used for any purpose other than for collections and payouts associated with delivery of services.
3. After each round of collections and payouts has been completed, this account will usually end with a balance of zero CPUcoin because the sum of all pay-ins is always equal to the sum of all payouts. It may end with a small positive in certain situations where a payout is too small to be worth transacting on the blockchain.

USE OF THE ETHEREUM BLOCKCHAIN

INITIAL EXCHANGE OFFERING

We will conduct an IEO in multiple stages to raise funds, and through doing so, will issue in total 5,000,000,000 (5 billion) CPUcoin tokens. These utility tokens will be used for transacting in the computing services market, thereby enabling a new token-based services economy.

For more information on the IEO, please refer to the section Token Model.

SMART CONTRACTS

We will use at least two ERC-20 smart contracts in total, to implement two types of token:

- IEO Tokens (token symbol CPU) – Before the CPUcoin comes to exist and begins trading and is transacted by the system, namely during stages of the IEO, we will need to be able to create, allocate, unlock and reserve tokens representing the buyer's stake in purchased CPUcoin. These can later be redeemed for ecosystem tokens one-for-one.
- Ecosystem tokens (token symbol possibly also CPU, still to be determined) – This is the utility token that will be the lifeblood of the service. Enterprise users will pay for the service in CPUcoin. Contributing Entities and other Beneficiaries of providing the service will also be compensated in CPUcoin for their efforts.

TRANSPARENCY

Our design includes assurance that every work unit operation, no matter how tiny, is accurately recorded and correctly billed for, with the proceeds appropriately divided. Transactions are also stored for later analytics and reporting, with complete openness, transparency and availability of all information surrounding work that has been completed. We will reap one of the most important benefits of blockchain, which is trust and proof that no data is ever tampered with.

SPEED

Because of the nature and design of the system, worker node work requests will be frequent and abundant. This strongly suggests the need for a microtransaction payment model. Due to overhead and gas costs associated with transacting on Ethereum combined with the sheer volume of data we expect to record it currently is not feasible to store all transactions on the Ethereum blockchain. The Ethereum network can only process 25 transactions per second globally and has high costs per transaction, making it ill-suited for microtransactions.

We will need a separate, off-chain solution to support microtransactions.

USE OF A PRIVATE BLOCKCHAIN FOR RECORDING MICROTRANSACTIONS

In this family of solutions, use of a private blockchain technology would greatly reduce the verification overhead by running all the blockchain nodes on our own servers. When we operate the nodes, there will be no bad actors. The proof of work problem associated with verifying blocks then collapses down to a simple “proof of authority” consensus operation, which can be generated instantly. Here are my reasons not to go this route:

1. It’s a bit of a heavy-handed solution for only recording transactions. If we chose Ethereum, that would likely require the creation of yet another token smart contract token for representing tokens that have associated metadata. Or perhaps we could select an alternate blockchain technology that supports fast, searchable metadata stored along with transactions in its native cryptocurrency.
2. This type of solution boils down to essentially a slow, complex and inflexible time series database with native support for chained transaction blocks using hashes. This infrastructure will be more

complex to operate and maintain than a simple time-series database. We can operate a higher capacity, highly scalable time-series database for less cost and effort.

3. On the other hand, there is appeal in getting to state that our project is a "pure blockchain" project.
4. Blockchain is a brilliant choice to fulfill on certain requirements, but not is not always appropriate for every problem.

WHY WE DIDN'T CHOSE AN "EXISTING" BLOCKCHAIN MICROTRANSACTIONS SOLUTION

There are projects that support microtransactions directly within the blockchain. For example, we could try to use something like Liquidity Network. This is essentially a general-purpose system for meshing off-chain microtransaction management with the Ethereum block chain. They have their own utility token (LQD). There are several other projects as well.

We have a few concerns about taking an approach that is dependent on an external vendor:

1. There is no available solution complete enough and ready enough for us to rely on today.
2. It is desirable that our system depend on as few vendors as possible. Because our solution is intended for enterprise use, issue investigation and resolution will be faster when it can be done in house without having to make requests to vendors.
3. Additional concerns include failure to complete product and potential major bugs. We would like to know that whatever we build, we can also operate, debug, fix and improve at a rapid pace.
4. By controlling the requirements, we can tightly scope the microtransaction recording system, simplifying it and making it easy to build and maintain. Our system requirements don't require a heavyweight, general purpose solution. We can build a streamlined system that excels at the type of work it is given.
5. Using a 3rd party microtransaction solution could add a second utility token into the business model, one completely unrelated to CPUcoin. This would likely create some confusion, would add complexity and might unfortunately link our project's success to unpredictable outcomes of an entity outside our control.
6. We do not want to depend on other solutions to solve one of our most interesting and compelling problems, as we consider that to be within our domain.
7. We have a saying: Avoid relying on others to provide technologies or services that we consider to be our own core competency. Do what we do best and reap the rewards and enjoy the rewards of building and operating systems that leverage the best of our expertise.

FUTURE-PROOFING

While we have chosen the Ethereum blockchain for launching our IEO and ecosystem tokens, we are designing our systems to be agnostic to the specific choice of blockchain. This is so that we can easily migrate the system to a different blockchain in the future, should new advances in blockchain technology offer advantages supporting such a change.

This is possible because modularity of the CGN and its components affords a design where software only communicates with the blockchain in isolated places, i.e. inside the payments system where batched rollup of microtransactions is performed (see Batched Collection and Disbursement). In addition, we have avoided depending on specific proprietary blockchain capabilities.

SUMMARY

We have chosen a solution using a minimal yet capable off-chain microtransaction manager for high volume, high-throughput recording, aggregation, collection and disbursement of funds transacted by the system.

This solution is explained in detail in the following section, Off-Chain Microtransactions.

Off-Chain Microtransactions

As already explained, we plan to internally use a readily available, scalable off-chain time-series database platform for storing microtransactions, strategically connecting it to the Ethereum blockchain for recording rolled-up transactions. This solution offers the performance and throughput benefits of a time series database along with immutability afforded by the blockchain. We are confident this design is tightly scoped to our needs and will be straightforward to build, deploy and operate.

Why Microtransaction Support is Highly Desirable

Each request made to a DService, no matter how quickly it runs, will be considered a billable event, which requires us to record it in the form of a microtransaction. These will occur at high volume, making rapid ingestion and storage of these transactions a critical requirement. This transaction data will be available to its corresponding regional CGN node to perform billing. Complete transparency in billing will allow users of the system to view and analyze prior transactions.

Microtransactions will be stored off-chain using a traditional distributed time-series database (such as Elasticsearch or its equivalent) which will be deployed and operated as a constituent component of each CGN node. Additionally, we will open this database to the outside world for read access through an analytics dashboard so that anybody using the system can query it for the data relevant to their needs.

Having an instance of this database per each regional CGN node assures continued scalability as the network grows.

Each microtransaction will be recorded at a fine level of detail, i.e. per-use of the network to consume work. Payment related to the work will be recorded together with a reference to the CPUcoin-to-fiat exchange rates in effect at that moment, enabling us on a later date to accurately compute the instantaneous value of either side of the transaction in the native currency of the end user. A scheduled process will later total up the amount collectible or payable per entity and record these final transactions in CPUcoin.

In summary, we record charges and payouts in fine detail and in real time and we collect those revenues and pay out the payables using CPUcoin later, following a periodic schedule. This results in the lowest possible transaction frequency on the Ethereum blockchain without sacrificing the ability to capture, collect and analyze usage data in fine detail.

Design of the Off-Chain Database

Implemented as a time-series database, a single row of data will be recorded for each microtransaction, with each row being comprised of multiple data columns. Each column will be configured to have a specific combination of these three characteristics, which controls how the column is used:

1. Immutable/mutable – Immutable data is data that cannot ever be changed after it has been written, whereas mutable data is data that can be changed after it has been written. Only immutable data can be made public and be included in the overall row hash for verification.
2. Internal/custom – Internal data is data that originates at the CGN, providing information such as which account and users are involved in the transaction, and the transaction amounts. Custom data that originates at the corporate entity and can be used to record aspects of the work performed that are company use case-specific, for later data mining using analytics.
3. Public/private – Public data is data that will be exposed to the world, such as for validation. Private data is internal data that will be held private. Some private data will also be hashed, with the hash stored as public. This is to differentiate between data about the transaction visible to the world (i.e. for validation) vs. data that may contain sensitive information, which is visible only to the corporate entity associated with the transaction. In the latter case, a hash of the data would be stored in parallel as public data.

This results in 8 combinations, with 5 of them being used:

Column Characteristic			Use
Immutable	Internal	Public	Public, internal information such as account and user id's (not traceable back to a specific user), transaction amounts that must be exposed for verification.
		Private	Private, internal information, such as the original full request URL, and performance metrics recorded for the work for later diagnostic use.
	Custom	Public	Hashes of immutable, custom private field data
		Private	Private, company-specific data to be logged and stored forever along with each transaction, typically for use later with company-specific analytics. Data such as what the user was doing, what workflow or user action this work was associated with, etc.
Mutable	Internal	Public	(never used)
		Private	Changeable information about the transaction, such as per-entity information indicating whether this transaction has been collected for or disbursed in CPUcoin, allowing a transaction to be marked "closed".
	Custom	Public	(never used)
		Private	(never used)

The schema will be designed such that for each recorded row, the following categories of data will be recorded:

IMMUTABLE/INTERNAL/PUBLIC

1. An event timestamp that always recorded in GMT to make it trivial to merge data from different time zones into a single data set.
2. The ID's of each participating entity: the enterprise account, the end user, the node operator, and any associated charitable entities that participated in servicing the request.
3. The usage fee amount to be billed, calculated at the time the request was completed.
4. The disbursement structure of the usage fee in the form of payable amounts for each participating beneficiary entity, as well as the CPUcoin exchange rates in effect at that moment.

IMMUTABLE/INTERNAL/PRIVATE

5. A copy of the request URL (along with HTTP request headers) that was made to the DService.
6. Metrics around performance and resource utilization in servicing the request. These are useful for debugging and problem diagnosis. We will also record information such as the worker node's reported resource availability, and measurements taken on the work performed such as network latency, actual time spent doing the work, and the amount of resources used by the worker node while servicing the request.
7. A collection of fields specific to the type of DService that was executed. This can be any information that the DService chooses to be recorded along with each microtransaction, such as DService-specific debug or diagnostic information about the work that was done.
8. The Ethereum account for each entity involved in the transaction, for which there will be kept mutable payment status information (discussed later)

IMMUTABLE/CUSTOM/PRIVATE

9. A collection of custom field values specified by the Enterprise user to be collected and stored along with the work performed, as received on the request or stored in the HTTP session. This can be business-critical workflow-related information pertaining to the work, or codes that annotate specific use scenarios that the enterprise wished to record.

Immutable/Custom/Public

10. A hash of all Immutable/Custom/Private field name/value pairs.

Mutable/Internal/Private

11. Microtransaction payment status information, per entity (discussed later).

This will all be collected and calculated upon completion of the request and recorded immediately.

The Per-Row Immutability Protection Hash

1. Additionally, we will store a hash of the full contents of all public data in each row. This will be a simple digest of the concatenation of the data of all public name/value pairs, alphabetically by column name, using Keccak-256. This provides a means to later prove the public row data was not tampered with in any way whatsoever. By including the column names, we ensure that the row schema was also not changed. This mechanism guarantees anyone, at any time in the future through public access to the database, can re-hash rows and check that all newly computed row hashes remain unchanged by comparing against the stored row hashes.
2. Because private immutable data is additionally stored publicly in hashed form, the row signature includes the row's private data. Therefore, private data is also protected by the row hash mechanism.
3. The row hash will be stored as Immutable/Internal/Public.
4. For additional protection against software errors, the storage API is designed such that Immutable fields cannot be altered in the time-series database once they've been written, even internally. In other words, these fields would be treated by the database wrapper API a write-once schema at the lowest level within the system. This is to prevent any code in the codebase to tamper with microtransaction data, intentionally or otherwise, during the period after it was recorded but before it has been converted to a blockchain transaction (whereby the aggregate hash of all microtransaction row hashes is stored in the blockchain).

Computing an Aggregate Hash

When microtransactions are rolled up to form a blockchain transaction between an entity and the payouts account, we need to compute an aggregate hash for all rows contributing to the blockchain transaction. This can include potentially thousands of rows, so we want this hashing of hashes to perform well. In addition, we require that the final aggregate hash value be the same regardless of the order in which rows were considered.

This can be accomplished simply by using the XOR operator to combine all the Keccak-256 row hashes. This provides more than adequate protection because it would statistically be effectively impossible to alter one row and also come up with an offsetting change to another row whereby the two new row hashes would XOR to the same 256-bit value, cancelling out the changes and resulting in the hash value that was stored on the blockchain.

Storage Considerations for Updateable Microtransaction Payment Statuses

1. We will also store in the time-series database some dynamic fields together with each microtransaction row to keep track of the current payment status for each microtransaction that we can update as microtransactions get processed by the rollup system. This state transition information does not participate in the immutability hash mechanism.
2. There will be a family of columns for each entity involved in the transaction to keep track of all accounts involved and the collection or payout status for each. Here are the payable beneficiary entities that must be represented on each microtransaction row:
    - The enterprise account entity representing the business whose users consumed the DService
    - The worker node operator entity that performed the work of the DService
    - The application developer entity who authored the chosen DService and receives use royalties
    - The CGN operator entity for the chosen regional CGN

3. For each of the above entities involved we would store:
    - The entity account id (immutable/internal/private)
    - An initially empty field to be later filled in with the CPUcoin transaction ID once this account has paid in or been paid out to in a complete CPUcoin transaction whole rollup included this microtransaction row.

Data Security

All enterprise-specific data is stored privately, meaning only that corporate entity can ever read the data. It will not be made public, and anybody performing blockchain validation would never see this data, only its hash.

Data Integrity

All public data (and hashes of all private immutable data) will be exposed to the public so that anybody can validate the hashes of this data or validate rows against hashes of aggregate data stored on the blockchain.

Data Retention

Because the microtransaction data will consume a lot of storage, CPUcoin may elect not to retain data going back more than 12 months. Any enterprise user who wishes to keep their data for a longer period will be able to download and store their data on their own or pay for a long-term storage solution if available from CPUcoin or one of its partners.

Also note there will be size constraints imposed on all logged data, but we will support the logging of reasonably large custom field values, on the order of up to a few kilobytes per field. This is mainly for support of custom, stored analytics data.

Batched Collection and Disbursement

Batched Rollup of Microtransactions into CPUcoin transactions

This section explains the process of processing batches of large numbers of pre-recorded microtransactions by reducing them down to individual blockchain transactions to be recorded as CPUcoin transactions.

Here is a summary for how we will implement this:

Perform batch processing on a periodic schedule

A scheduled process is set to on a regular schedule such as monthly. We may adjust the billing frequency or spread out the process as needed in order not to flood the Ethereum network with transactions once the system has many users.

The objective of this process is to sift through all pending microtransactions and try to resolve them with actual CPUcoin token transfers. Each logged microtransaction row has been written with all important values precomputed. Each row specifies a (usually very tiny) amount of CPUcoin to be paid in by a single Enterprise user, and a list of amounts to be paid out to all associated beneficiary entities, with the pay-in amount always equaling the total of the payout amounts.

Collect outstanding receivables

The first thing the batch process does is filter all pending microtransactions, grouping them by Enterprise account. Then it totals all pay-in amounts for each Enterprise account and generates a list of amounts payable. While processing the microtransactions to sum pay-ins, it also builds up an aggregate hash by XORing the hash from each row into a 32-byte (256-bit) buffer that starts off all zeroes. When complete, both a total and a composite will hash have been generated for each Enterprise account.

Next, it iterates through this list and generates a single CPUcoin transaction to be paid from the Enterprise account into the CGN payouts account. To accomplish this, the CGN uses ERC-20 method `transferFrom()` to collect funds that the Enterprise account user has already pre-allocated using their self-service account (using of the ERC-20 `approve()` method).

When generating the CPUcoin transaction, the aggregate hash is also passed as extra data, to be logged onto the blockchain together with the transaction. That anchors this transaction to all its constituent microtransaction rows.

If `transferFrom()` were to fais due to insufficient funds, the CGN will immediately purchase or obtain the needed shortfall amount of CPUcoin either on the open market or directly from CPUcoin's corporate account. The Enterprise user will be billed using a common ecommerce payment gateway, using a payment method that the Enterprise user has already consented to us using.

Any remaining unresolved payment-related issues will result in the Enterprise user account being flagged for later follow-up, which also can be automated in most cases. It is anticipated that the need for human intervention should be a rare outcome.

Regardless, at the ends of this step, a CPUcoin pay-in will have been made for every Enterprise user having an outstanding balance, either from the user's account or from our own sources. All participating microtransaction rows are updated to a status indicating payment has been handled for this entity. In addition, the blockchain transaction id is written back to the microtransaction row for ease in later mapping fully paid microtransactions to a corresponding aggregate CPUcoin transaction.

Remit outstanding payables

This process is nearly identical to how we handle pay-ins for Enterprise users. Except here we re-filter the same set of microtransactions by looping over all involved beneficiary entities of every kind, filtering rows and grouping by beneficiary entity.

As above, the remaining steps are handled similarly: microtransactions are grouped by the entity and then both a total and an aggregate hash are computed for each beneficiary entity. Then a payout transaction in CPUcoin is generated on the blockchain using the ERC-20 method transfer(), together with the aggregate hash to be logged together with the transaction.

In the event an amount payable is below a certain minimum threshold, the payout won't be made in this billing cycle. This is to prevent generating transactions where the gas cost is significant compared to the transaction value. That will result in a small positive balance remaining in the payouts account, which will be resolved later.

At the end of this step, all (or almost all) beneficiary entities will have been paid, and the corresponding microtransaction rows will have been updated with a payment complete status and a blockchain transaction id for every entity on each row.

Upon Completion

At the end completion of all payment batch operations, the payout account will end up with all (or almost all) funds having been accounted for and will therefore always have a final balance of zero CPUcoin (or a very small positive balance to be resolved later).

We will measure the time it took (per-enterprise account, per-beneficiary, and overall) perform batched rollup and executing each transaction in CPUcoin. We will use this data for diagnostic purposes, and to ensure that as the system grows, we can adapt the solution if the batching mechanism begins to take too long.

How Blockchain Ensures Correctness and Prevents Tampering with Microtransactions

Our solution offers proof of correctness and ensures that any tampering of data in the microtransaction database would be detected. This provides the same immutability benefit afforded by blockchain transactions, even though the benefiting transactions in this case are stored internally, off-chain.

This insurance comes from the aggregate hash that we store on the blockchain along with every CPUcoin transaction composed from all contributing microtransaction rows. This aggregate hash is influenced by all data from each microtransaction row that participated in the summation query when computing the total CPUcoin amount for that entity.

In conclusion:

1. The internal microtransaction database will be constructed so that at the API level, immutability is enforced. Every measure possible will be taken to ensure there is no way to get into the microtransaction database and change immutable data. This will prevent errant code from inadvertently altering past data.
2. If immutable data of any microtransaction database row were somehow tampered with, that data will no longer match the hash stored on the same row. This will be publicly detectable by validators.
3. If the stored row hash were somehow altered to match the new data, then the blockchain-recorded composite hash associated with the CPUcoin transaction that included that microtransaction would no longer match. This will also be publicly detectable by validators.
4. Thus, we provide the necessary means for detecting all problems with data integrity, ensuring that validators can discover if anyone ever changes past data kept in the microtransaction database.

Off-Chain Microtransactions

We will use a readily available, scalable off-chain time-series database system to store microtransactions, and strategically connect it to the Ethereum blockchain. This solution will provide the performance and throughput benefits of a time series database, with all the benefits of proof of correctness of the blockchain. We are confident that we can create a system that is tightly scoped to our needs that will not be complex to design, build and deploy.

Why Microtransaction Support is Highly Desirable

Each request made to the DService, no matter how quickly it runs, will be considered a billable event, requiring it to be logged in the form of a microtransaction. These will occur at very high volume, and the need for rapid ingestion and storage of these transactions is a critical requirement. The transaction data will always be available, to be accessed both by the regional CGN nodes to perform billing. Complete transparency in billing will allow users of the system to view and analyze all prior transactions.

Therefore, microtransactions will be stored off-chain in a traditional distributed database (using a time-series type of database such as Elasticsearch or equivalent) which will be deployed and operated solely by us. We will open this database to the outside world for read access through an analytics dashboard so that anybody using the system can query it for the data relevant to their needs.

For scalability, an instance of the microtransaction database would most likely be a constituent component of each regional CGN node.

The microtransactions will be recorded at a fine level of detail, i.e. per-use, but will be recorded in fiat currency so that the cost of services is expressed to each paying customer in a familiar currency. For transacting on the blockchain in CPUcoin, a scheduled process will roll up totals collectible or payable, measured in fiat currency, convert the amounts to a CPUcoin value at the exchange rate at the time of each microtransaction, and then conduct the total amount transaction using CPUcoin, and record it on the blockchain.

In summary, we record charges and payouts in fine detail and in real time, using fiat currency. And we collect those revenues, or pay out the payables, using CPUcoin on a periodic schedule. This results in a better transaction frequency match for use with the blockchain, without sacrificing the ability to capture, collect and analyze usage data in enough detail.

Design of the Off-Chain Database

Implemented as a time-series database, a single row of data will be recorded for each microtransaction, with each row being comprised of multiple data columns. Each column will be configured to have a specific combination of these three characteristics, which controls how the column is used:

1. Immutable/mutable – Immutable data is data that cannot ever be changed after it has been written, whereas mutable data is data that can be changed after it has been written. Only immutable data can be made public and be included in the overall row hash for verification.
2. Internal/custom – Internal data is data that originates at the CGN, providing information such as which account and users are involved in the transaction, and the transaction amounts. Custom data that originates at the corporate entity and can be used to record aspects of the work performed that are company use case-specific, for later data mining using analytics.
3. Public/private – Public data is data that will be exposed to the world, such as for validation. Private data is internal data that will be held private. Some private data will also be hashed, with the hash stored as public. This is to differentiate between data about the transaction visible to the world (i.e. for validation) vs. data that may contain sensitive information, which is visible only to the corporate entity associated with the transaction. In the latter case, a hash of the data would be stored in parallel as public data.

This results in 8 combinations, with 5 of them being used:

Column Characteristic			Use
Immutable	Internal	Public	Public, internal information such as account and user id's (not traceable back to a specific user), transaction amounts that must be exposed for verification.
		Private	Private, internal information, such as the original full request URL, and performance metrics recorded for the work for later diagnostic use.
	Custom	Public	Hashes of immutable, custom private field data
		Private	Private, company-specific data to be logged and stored forever along with each transaction, typically for use later with company-specific analytics. Data such as what the user was doing, what workflow or user action this work was associated with, etc.
Mutable	Internal	Public	(never used)
		Private	Changeable information about the transaction, such as per-entity information indicating whether this transaction has been collected for or disbursed in CPUcoin, allowing a transaction to be marked "closed".
	Custom	Public	(never used)
		Private	(never used)

The schema will be designed such that for each recorded row, the following categories of data will be recorded:

IMMUTABLE/INTERNAL/PUBLIC

1. An event timestamp, always recorded in GMT so it would be easy to merge data from regional CGN nodes in different time zones into a single data set.
2. The ID's of each participating entity: the enterprise account, the end user, the node operator, and any associated charitable entities that participated in servicing the request.
3. The usage fee amount to be billed, calculated at the time the request was completed.
4. The disbursement structure of the usage fee in the form of payable amounts for each participating beneficiary entity, as well as the CPUcoin exchange rate at that moment. This will all be collected and calculated upon completion of the request and recorded immediately.

IMMUTABLE/INTERNAL/PRIVATE

5. A copy of the request URL (along with HTTP request headers) that was made to the DService.
6. Metrics around performance and resource utilization in servicing the request. These are useful for debugging and problem diagnosis. We will also record information such as the worker node's reported resource availability, and measurements taken on the work performed such as network latency, actual time spent doing the work, and the amount of resources used by the worker node while servicing the request.
7. A collection of fields specific to the type of DService that was executed. This can be any information that the DService chooses to be recorded along with each microtransaction, such as DService-specific debug or diagnostic information about the work that was done.
8. The wallet for each entity involved in the transaction, for which there will be kept mutable payment status information (discussed later)

Immutable/Custom/Private

9. A collection of fields specified by the Enterprise user to be stored along with the work performed, as received on the request. This can be business-critical workflow-related information related to the work, or codes that annotate specific use scenarios that the enterprise wished to record.

Immutable/Custom/Public

10. A hash of all Immutable/Custom/Private fields.

Mutable/Internal/Private

11. Microtransaction payment status information, per entity (discussed later).

The Per-Row Immutability Protection Hash

1. Additionally, we will store a hash of the full contents of all public data in each row. This will be a simple digest of the concatenation of the data of all public columns, alphabetically by column name, using a lightweight hashing mechanism. For each column we will hash both the column name and the column data. This ensures that the public row data was not tampered with in any way whatsoever. By including the column names, we ensure that the row schema was also not changed. This mechanism ensures that anyone, at any time in the future through public access to the database, can re-hash rows and check that all computed row hashes remain identical to the stored row hashes.
2. Because private immutable data is additionally stored publicly in hashed form, the hash of private data will be included in the row hash. Therefore, private data is also protected by the row hash mechanism.
3. The row hash will be stored as Immutable/Internal/Public.
4. We will design the storage API such that Immutable fields cannot be altered, even internally, once they've been written. In other words, these fields would be treated as a write-once schema at the lowest level within the system. This is to make it impossible for any code in the codebase to tamper with microtransaction data, intentionally or otherwise, during the period between it having been recorded, but before having been converted to a blockchain transaction (whereby the aggregate hash of all microtransaction row hashes is stored in the blockchain).

Storage for Changing Microtransaction Payment Statuses

1. We will store associated data along with the microtransaction data to be used to maintain state information about the microtransaction that can change over time. This information does not participate in the immutability hash mechanism.
2. There will be a family of columns for each entity involved in the transaction to keep track of all wallets involved, and the collection or payout status for each. Here are the beneficiary entities that would be represented on each microtransaction row:
   - The enterprise account wallet for the business, whose users consumed the DService
   - The work node operator who performed the work of the DService
   - The application developer of the chosen DService
   - The CGN operator for the chosen regional CGN

3. For each of the above entities involved we would store:
   - The entity wallet (immutable/internal/private)
   - An initially empty field, to be filled in with the CPUcoin transaction id, right after this wallet has paid in or been paid out to in a complete CPUcoin transaction that incorporated this microtransaction row.

Data Security

All enterprise-specific data is stored privately, meaning only that corporate entity can ever read the data. It will not be made public, and anybody performing blockchain validation would never see this data, only its hash.

Data Integrity

All public data (and hashes of all private immutable data) will be exposed to the public, so that anybody can validate the hashes of this data or validate rows against hashes of aggregate data stored on the blockchain.

Data Retention

Because the microtransaction data will consume a lot of storage, CPUcoin may elect not to retain data going back more than 12 months. Any enterprise user who wishes to keep their data for a longer period is responsible for downloading and storing their data on their own or paying for a long-term storage solution if available from CPUcoin or one of its partners.

Also note there will be size constraints imposed on all logged data, but we will support the logging of reasonably large pieces of data, on the order of up to a few kilobytes per field. This is mainly for support of custom, stored analytics data.

Batched Collection and Disbursement

Batched Rollup of Microtransactions into CPUcoin transactions

As explained earlier, microtransactions are always recorded in real time using fiat currency. But CPUcoin is used in a much less frequent manner, for regular settlement of billing and payouts according to a regular schedule that is recurs less often. i.e. daily.

Here is a summary for how we will implement this:

1. Query the microtransactions database for all transactions applicable to each specific entity over the batching time period (one calendar day).
2. Accumulate the total amount in fiat currency.
3. Convert that to a CPUcoin amount using the exchange rate associated with each microtransaction.
4. Submit that as a CPUcoin transaction to the blockchain to handle collection or payment for that periodic batch of DService requests provided or delivered, also recording the aggregate microtransaction hash.
5. Mark microtransaction rows that participated in the total payment as having been paid/collected, for that entity. We will immediately update all rows of the time series database that have been handled to prevent double-billing or double-payout for DServices.

CPUcoin Transactions for Payment in Aggregate

The CPUcoin transaction amount will be computed by calculating the rolled-up sum in fiat currency, and then converting that total to CPUcoin at the going conversion rate. We will then transfer that CPUcoin amount between the appropriate wallets.

Microtransaction rollup Procedure for billing in CPUcoin.

We will roll up timespan-based batches of transactions and submit CPUcoin transactions for them. We would perform these steps for one paying user (enterprise user) at a time, looping over all the accounts and performing the collections steps just for that account, and then perform the disbursement steps for all accounts.

Here are the steps to perform, in detail, for each paying enterprise account, to be executed once every collection period:

Iterate over all enterprise accounts having transactions in this CGN

1. To kick off the batch run, we would use a scheduled task so that the batch job is run at a regular interval that covers a predetermined, precise regular time span (such as exactly one day).
2. Identify all enterprise accounts having transactions in this CGN over the past batching period (one day) and iterate over those accounts.

Do pre-collection top-up (per-enterprise account)

1. First, we determine the total amount to be collected for the total of all use of a given DService, that was incurred by all users in a given enterprise account.
2. We then compute the sum total of use fees in fiat currency by running a microtransaction database summation query.
3. Then convert this summed total amount in fiat currency to a CPUcoin amount using the CPUcoin exchange rate at that moment (as stored on each row).
4. While processing the query, we will collect microtransaction row hashes from all rows included in the query and compute an aggregate hash from those values.
5. Next, we reserve funds, and top-up if needed. We use the reserve() method on the wallet to attempt to set aside funds for to be used for payment. If this fails, we top-up.

6. To implement top-up, we automatically purchase CPUcoin on the open market on their behalf (using the pre-arranged payment method) at the current market rate. We then go back to the previous step, and again attempt to use reserve() to set aside the funds for payment.
7. This catches the case of somebody transferring CPUcoin out of a wallet during the brief time between top-up and reservation of funds. This would be detected as a second failure to reserve(), and would again be corrected by repeating the top-up step.

Collect use fees (per-enterprise account)

1. We will then transact the actual payment for services in CPUcoin. To do that, we create a single CPUcoin transaction that transfers funds from the Enterprise Account wallet into the payout wallet, for the previously calculated total amount to be collected.
2. When creating the CPUcoin transaction and entering it into the blockchain, we store the aggregate hash computed earlier so that the CPUcoin transaction keeps a permanently recorded, immutable aggregate hash from all contributing microtransaction data, thereby anchoring microtransactions to the blockchain.

Complete for all Enterprise Accounts

1. Iterate to complete the above steps for all enterprise accounts. The payout wallet is now filled with all funds to be disbursed.
2. After collecting funds from all enterprise accounts, we next do all associated payouts.

Do payouts to beneficiaries

1. To implement handling of payouts, we batch each payout by beneficiary type and ID. This results in a double nested loop, as we must run these steps for each type of beneficiary. For cases where there are multiple beneficiaries of the same type (such as contributing entities), we must identify each participant entity in that time-period, and then loop over each individual entity. This is easy to implement as a series of queries to the microtransaction database.
2. For each individual entity we perform the following steps:

Steps performed per-beneficiary

1. Run a microtransaction database query that queries all payments to be made to that beneficiary entity and computes a total sum in fiat currency.
2. Then convert this summed total amount in fiat currency to a CPUcoin amount using the CPUcoin exchange rate at that moment (as stored on each row).
3. While processing the query, we will collect microtransaction row hashes from all rows included in the query and compute an aggregate hash from those values.
4. Create a CPUcoin transaction that transfers funds in CPUcoin to the beneficiary wallet from the payout wallet.
5. When creating the CPUcoin transaction and entering it into the blockchain, we store the aggregate hash computed earlier so that the CPUcoin transaction keeps a permanently recorded, immutable aggregate hash from all contributing microtransaction data, thereby anchoring microtransactions to the blockchain.
6. Proceeds paid out to the regional CGN operator wallet will be handled in the same manner.
7. Proceeds paid out to associated charitable cause wallets will be handled in the same manner.

Upon Completion

At the end completion of all payment batch operations, the payout wallet will end up with all funds having been accounted for and will therefore always have a final balance of zero CPUcoin.

We will measure the time it took (per-enterprise account, per-beneficiary, and overall) perform batched rollup and executing each transaction in CPUcoin. We will use this data for diagnostic purposes, and to ensure that as the system grows, we can adapt the solution if the batching mechanism begins to take too long.

Ways CPUcoin Can Generate Income

CPUcoin may be paid in multiple stages of the payout process:

1. CPUcoin will receive the CGN operator fee for operating regional CGN nodes.
2. CPUcoin will receive a payout royalty fee for its DServices
3. CPUcoin will receive a payout for any worker nodes it operates. CPUcoin will operate enough worker nodes to guarantee enough resources are available within the CGN for it to run reliably.

How Blockchain Ensures Correctness and Prevents Tampering with Microtransactions

Our solution offers proof of correctness and ensures that any tampering of data in the microtransaction database would be detected. This provides the key benefits usually associated with blockchain transactions, even though the benefiting transactions in this case are stored off-chain.

This insurance comes from the aggregate hash that we store on the blockchain along with every CPUcoin transaction composed from those microtransaction rows. This aggregate hash is influenced by all data on every microtransaction row that participated in the summation query used to compute the CPUcoin amount. We always do it at the time that we perform any summation query for computing a total amount that will be payable or collectible in CPUcoin.

In conclusion:

1. If the immutable data of any microtransaction database row is ever tampered with, the data will no longer match the hash stored on that row. This will be publicly detectable by validators.
2. If the stored row hash were to be changed to match the new data, then all CPUcoin transactions that included that microtransaction data would contain a hash that no longer matches the aggregate hashes of all microtransaction rows. This will also be publicly detectable by validators.
3. Thus, we provide the necessary means for detecting all problems with data integrity, ensuring that validators can discover if anyone ever changes past data kept in the microtransaction database.
4. The microtransaction database will be deployed so that is impossible for the codebase to change already-written immutable microtransaction data. Every measure possible will be taken to ensure there is no way to get into the microtransaction database and change immutable data.

THE DEAL

Each deal struck between an enterprise company and CPUcoin will establish one of two major financial aspects of transactions for all work performed:

1. The cost structure for fees billed for doing the work.

The second aspect of the deal will be established in advance by CPUcoin, and will be tuned to optimize the system:

2. The payout structure for disbursing collected fees.

COST STRUCTURE

It's important to note that regardless of which cost structure is in effect, the microtransaction-based billing system works the same. Payouts are completely unaffected by which model is selected. Only CPUcoin and the enterprise account are impacted by the choice in model, and the impact lies only in the balance between the cost of computational resources consumed, and what the enterprise customer pays for them.

PER-USE VARIABLE COST

This section describes how the enterprise customer will be charged for work on a per-use basis.

We log the actual cost incurred for all work done, in the form of microtransactions. There are several factors that can affect the cost of the work done:

1. The service level tier at which the work being requested should be performed, on a per-request basis. This is expressed as a number composed of three components: the minimum number of threads, the minimum CPU power, and the minimum bandwidth requested.
2. A maximum price cap on the work with tier of service downgrade, on a per-request basis. To support this, we will invoke a work estimation component of the DService which quickly examines an incoming request and estimates the cost of the work being requested. If the cost exceeds the price cap, the CGN will substitute a lower tier of service than was requested that can do the work at or below the price cap. If the estimated cost is still higher than the price cap, then the CGN will consider machines that currently have a heavy load and therefore could offer a deep discount on the price of the work, since they will charge less for work due to being performance-bound.
3. A maximum price cap on the work, covered by a subsidy (if established in advance for the enterprise account). This is an alternative to the other price cap model which guarantees the quality tier of service at our expense. It is a bit kinder on the enterprise customer. It would perform a lot better because the estimation component would not need to be invoked, and the enterprise user would always get the requested tier of service.
4. A full subsidy (if established in advance for the enterprise account). There are cases where CPUcoin may wish to subsidize an enterprise company's work requests outright. Some examples would be:
   - Handling all requests for free in test environments. The remainder of cases refer to production usage.
   - Handling very short requests for free
   - Handling all requests for free during a period while we are testing or tuning the service as it gets used by an enterprise customer.
   - Handling all requests for free because the enterprise customer pays for the service through a payment channel outside the CGN (such as monthly, by check, in fiat currency).

SUBSCRIPTION (PER-DAY FIXED COST)

This section discusses how the enterprise customer will be charged on a subscription basis.

We log the actual cost incurred for all work done, in the form of microtransactions. Since there are no factors affecting the cost of the work, we just do the work based on pre-agreed terms and log the actual cost of the work.

But at the end of every day, when doing batched roll-up collections and payouts, while we will determine the total cost of the work done for the day, we instead collect a specific, fixed daily amount that corresponds to the agreed-upon monthly transaction amount.

However, we must fund the payout wallet every day with the actual cost of work consumed that day. To do that we must compute the difference between the actual cost of the work and the daily subscription amount collected and handle the difference. If the actual work cost more than the subscription amount, then CPUcoin would subsidize that amount by paying in the difference. If the actual work cost less than the subscription amount, then we still collect the full subscription amount but pay out the difference to CPUcoin.

In this model, there is no per-request way for the enterprise user to select a specific tier of service or to set a price cap. Instead, there is no price cap, and the tier of service will have been established in advance for the enterprise account. We just perform the work at the pre-established tier of service, log the actual cost of the work as measured by the DService, tally all work at the end of the day, and determine the under- or over-payment against a daily subscription amount, with the difference either coming from or going to CPUcoin.

Payout Structure

The payout structure is independent of the cost structure, and is based upon fixed, pre-determined ratios that will be established by CPUcoin.

We have explained before that payouts P will be split between parties as follows:

Entity	Payout Fraction (%) (from 0 to 100%)	Value of Payout	Explanation
Contributor	C	$P * C$	Contributors get a fixed percentage of all payouts.
Developer	D	$P * D$	Developers get a fixed percentage of all payouts.
CGN Operator	$1 - (C + D)$	$P * (1 - (C + D))$	The CGN operator gets the remaining amount.

(Where C + D obviously must total less than 1).

Each of the above three entities may optionally select a charitable wallet, with a specified percentage of the proceeds that go into that wallet. Therefore, the right column expresses an amount that will potentially be split between that entity's wallet and the charitable wallet of their choice, at a ratio specified by that entity.

Tuning

CPUcoin must carefully tune the model so that it makes economic sense to all parties. The model will be tuned by analyzing use and making precise choices to optimize the following parameters:

- Pricing, in fiat currency, for the various service level tiers
- The constants C and D

The choices for constants C and D must be carefully made, while also making every effort to ensure that:

- Enterprise customers will pay an amount less for the service than it would cost to provision in a data center or using a cloud service. The cost of the service must be compelling to enterprise customers.
- Contributors will be paid an amount for contributing computational resources that is competitive against the other alternatives (such as SONM, Golem, iExec, and Bitcoin or Ethereum mining, as well as other upcoming possibilities)
- The developer will be paid a fair royalty for use of the software.
- The CGN operator will be paid a small but fair fee to compensate it for acquiring and overseeing the reliable operation of all hardware comprising the CGN internally.

It will likely require a fair amount of experimentation along with many adjustments based on empirical observation to find the best balance for these parameters.

Strategy to Ensure Dependable CGN Operation

Initial Start-up of the Service

We will build test versions and a production version of the system using data center-grade hardware, either in a data center or by use of cloud computing resources. When we roll out the production system, it will be initially stocked with enough computational capacity to meet the needs of all enterprise users of the system at that time. We will operate the initial supply of worker nodes, also on enterprise-grade hardware, at our own expense.

Ensuring Sufficient Compute Capacity

One of our commitments is to ensure high quality of service for enterprise customers using the service. Therefore, we will have a means to fulfill on this promise.

To do so, we will monitor all regional CGN nodes for capacity and utilization, looking for oversupply or impending undersupply of resources, and actively adjust by bringing enterprise-grade hardware online where needed, at our own cost (we will also remove unnecessary capacity to control costs). This will help buffer the system from initial lack of participating worker node contributors, as we promote use of the system and build its popularity and reputation.

As more and more volunteer worker nodes come online, we anticipate that the need for our providing worker hardware will lessen, but we will continue to play the role of ensuring reliable operation of the CGN by adding and removing capacity to keep the system operating smoothly.

Ensuring Liquidity in the CPUcoin Markets

Our design is dependent on high liquidity of the CPUcoin, meaning that at any moment, it should be easy to purchase or sell CPUcoin in fiat currency. Therefore, we will take measures to ensure that the bid/ask spread (the difference in the exchange rates for buyers and sellers) between CPUcoin and a fiat currency are kept to a reasonable minimum.

For stable operation of the ecosystem as a utility, we will try to ensure that there are one or more market makers in CPUcoin operating in the markets, willing and able to purchase and sell CPUcoin against fiat currency at any given moment, with a reasonable spread.

Managing CPUcoin Price Stability

As a matter of prudent fiscal policy, after conclusion of the IEO, CPUcoin may from time to time elect to burn CPUcoin tokens in order to try to minimize price volatility or to support its price, should market conditions warrant.

Systems for Managing Operations

The Public-Facing CPUcoin Site

This public site will provide the following capabilities:

- A self-service portal so users may view their account to see holding and view reports detailing CPUcoin spent or earned.
- A download site so that contributors can acquire the DService worker node software installer.
- A convenient way to find a CPUcoin token exchange, for buying and selling CPUcoin.
- Support for creation and management of contributing entity accounts, worker node accounts and worker node wallets.
- A services layer that worker nodes and DApps can use to get information about nearby regional CGN's to join.

An Internally-Facing Operations Portal

This CPUcoin internal will provide the following capabilities:

- Support for approval of Miner Accounts, as well as banning or suspending misbehaving users.
- Support for creation and Management of Enterprise User and End User Accounts.
- A services layer that regional CGN nodes can connect to for access to user and enterprise account information as well as verification of login credentials, access to CPUcoin wallet balances and quick lookup of fee-splitting agreements.
- Operate a registry of DServices and their privately negotiated fee distribution structures.
- Provide internal dashboards for monitoring the health and vital statistics of systems.
- Provide DevOps tools for deployment of a new regional CGN node.
- Provide DevOps tools for administration and management of existing regional CGN nodes.
- Provide DevOps tools for capacity management analysis, and the ability to quickly add capacity to a regional CGN node in need.

THE RELATIONSHIP OF CPUCOIN TO RESOURCES

INTRODUCTION

CPUcoin is a decentralized cryptocurrency that is equivalent to a specific deflation-adjusted amount of computational power having no central owner. CPUcoin is used as a means of transacting in computational resources within the CGN platform, which runs autonomously.

In this section we will explain how this relationship is established and maintained. This is an important topic because an effective utility token must be unquestionably linked to its corresponding utility resource in a durable way so that the token will have well-defined, lasting utility in its ecosystem.

It's also important that this model be simple and understandable. The model must also be as future-proof as possible. This means we must, to the best of our ability, proactively take measures to ensure that the relationship between the token and the utility of its resources will remain stable for the long-term.

Finally, the model must be implemented universally. We will fulfill that by building this calculation into our token smart contract. API methods will compute and return the current resource cost in effect at the time.

COMPUTATIONAL RESOURCE TYPES

CPUcoin represents three types of computational resource, which, through the ecosystem, may be either consumed or provided:

1. CPU power
2. GPU power
3. Bandwidth (data)

As a starting point for determining equivalency between amounts of disparate types of computational resources, we first researched market pricing for these computational resources by analyzing the AWS prices set for them. We then normalized the results to represent pricing in standard, interchangeable units of the resource.

AWS pricing, as of January 31, 2019:

Normalized cost of computational resources (AWS)			
Resource	Unit	Cost (€)	Explanation
CPU	1000 MIPS/hr	€0.000263	This represents one full hour of CPU time for a single core of CPU running at 1000 MIPS or 1 KMIPS (one billion instructions per second), which is based on the CPU clock frequency and average number of instructions per clock cycle. This is determined for a given miner's CPU by running a CPU benchmark test.
GPU	1 TFLOPS/hr	€0.325	This represents one full hour of GPU time for a single GPU unit providing one tera-FLOP (one trillion floating point operations per second) of computational power. This is determined for a given miner's GPU by running a GPU benchmark test.
Bandwidth	1 Gigabyte	€0.0877	This represents one gigabyte (GB) of data transmitted (sent or received) over the Internet. This is determined by measuring actual data usage at the miner.

The CPUcoin utility token will represent a store of value that can be exchanged for any of these types of computational resource. The model is inherently symmetrical and neutral. This means the amount of each of these resources exchangeable for one CPUcoin will be the same regardless of whether the resource is being consumed (i.e. when purchased by the end user using a dApp) or is being provided (i.e. when a CGN miner is reimbursed for those resources consumed on behalf of the end user's dApp). Furthermore, the amount of exchange will be independent of with whom it is being exchanged. The purpose of the CPUcoin is to represent and permit the exchange of raw computational power as a pure commodity.

Defining CPUcoin

We are defining one CPUcoin as being equivalent to the amount of computational resources that could be purchased for €1.00 on AWS on January 31, 2019. That results in the following amount of resources per coin:

CPUcoin equivalency to computational resources (January 31, 2019)			
Resource	Amount	Unit	Explanation
CPU	3290	KMIPS-hours	3290 KMIPS of CPU power consumed or provided for one hour.
GPU	3.08	TFLOP-hours	3.08 TFLOPS of GPU power consumed or provided for one hour.
Bandwidth	11.4	Gigabytes	11.4 GB of data consumed or provided.

The steadily decreasing current price of each type of resource will be updated daily in the token smart contract.

Representing a Blend of Compute Resources

When the ecosystem is launched, the CGN will encounter miners that are substantially different from AWS servers in spec, offering widely ranging variations of CPU and GPU power, capability and configuration. However, the CPUcoin must be able to represent the ability to transact in all forms of CPU and GPU.

To adjust for CPU and GPU configurations that are superior to or inferior to the AWS benchmark, we will simply apply a premium or a discount to the base resource amount of one CPUcoin, using premade lookup tables that express differently performing resource variants as ratios against the benchmark. For missing data, we will interpolate between existing data. This approach lets us fairly exchange CPUcoin for resources relating to any CPU or GPU configuration that may be encountered by the CGN, using the basic model established here.

Compensating for Deflation

Moore's law is the observation that the number of transistors in a dense integrated circuit doubles about every two years. This translates into the capacity of new hardware of comparable cost doubling about every two years. This is equivalent to saying that the cost associated with a given measure of capacity is cut in half about every two years.

The fact that the value of computational resources steadily declines over time is a crucial consideration that we must account for in a pricing model for the CPUcoin. To support a stable economy for our ecosystem, the relationship of CPUcoin to the computational resources that it represents must remain steady over long periods of time. If we didn't account for this, there would be big disincentive to holding CPUcoin for the long term because the amount of its computational power would be worth increasingly less in the future. Confidence in holding a resource can only be ensured if the resource can maintain a steady value. Therefore, our model must compensate for the inherent deflation in the value of computational power.

With the value deflation stated as a halving of value every two years (730 days), the value of computational resources would decrease daily by the following ratio:

$$\text{Daily Deflationary Factor } D = \frac{1}{2^{\frac{1}{730}}} = 0.999050933963$$

Because this effect is widely believed to be both persistent and predictable, we can compensate for it by applying an equal amount of inflation:

$$\text{Compensating Daily Inflationary factor } K = \frac{1}{D} = 2^{\frac{1}{730}} = 1.000949967619$$

Therefore, the amount of resources represented by one CPUcoin must be adjusted daily as follows, by relating the amount of resources R(N) on day N to the amount of resources R(0) on day 0 as follows:

$$R(N) = R(0) * K^N$$

Our ecosystem's pricing engine will be designed and built to promise and guarantee that this adjustment will continually be made. By making a gradual adjustment continually (i.e. computing it daily instead of over longer time intervals) we will minimize any disruption caused by perceived hikes or discontinuities in pricing.

To illustrate the effect, here is a table showing the increase in the amount of resources tradeable for one CPUcoin on different dates:

CPUcoin resources equivalency, adjusted over time								
Resource	Unit	Amount (day 0)	Amount (after 1 day)	Amount (after 1 week)	Amount (after 1 month)	Amount (after 3 months)	Amount (after 1 year)	Amount (after 2 years)
CPU	KMIPS-hrs	3290.4	3293.5	3312.3	3385.5	3584.0	4653.3	6580.8
GPU	TFLOP-hrs	3.08	3.08	3.10	3.17	3.35	4.36	6.16
Bandwidth	Gigabytes	11.4	11.4	11.5	11.7	12.4	16.1	22.8

Now that we have shown how the CPUcoin is fundamentally linked to computational resources, we are ready to discuss pricing of the token in our token sale.

Adjusting for Change

The situation may arise where new developments in CPU, GPU and bandwidth may cause alter their cost structures to change radically. Therefore, we will have the ability to administratively update the resource cost coefficients within the token smart contract keep the rate of price change representative of actual current costs. Such changes will never affect costs in the past, only the rate at which costs change in the future.

What About Storage?

DServices will have access to a limited, but reasonable amount of temporary local storage to be used for purposes of sharing results between subsequent requests, or between clustered miners. Such storage is not intended for long term use, and the system will automatically delete leftover files as they expire to reclaim space. For DServices requiring access to long-term storage, we suggest using any form of cloud storage, and communicating the location of input and output files between the DService and DApp by exchange of URLs. Local storage may be used to temporarily cache those results to minimize retransmission.

THE SALE

Note: This section is preliminary. All numbers, dates and amounts specified herein are subject to change.

INTRODUCTION

After reading the previous pages of this white paper, you may have concluded that what we propose to build is quite an ambitious project. To recap, we will build a global, infrastructure-as-a-service ecosystem powered by CPUcoin that is specialized in deploying and operating DServices at scale, doing so through careful and efficient orchestration of unused CPU time, obtained from millions of servers and computers of all kind operating around the globe. We will sell those services to existing and new enterprise customers, and will work hard to develop and expand the user base of this new ecosystem.

This project has the potential to be disruptive to technologies such as AWS, Azure, and data center hosting, by providing a cost competitive IaaS alternative solution. However, these data center hosting costs can be offset by installing our CGN resulting in payouts based on volume. Just like with solar in a home reducing an electric bill, legacy CPU resources from any source can off-set their total net cost, therefore, we expect partnerships with AWS and Azure, as well as, other data centers to be formed to capitalize on this opportunity.

Our ability to fulfill on the design by building and operating a minimum viable version of the system can be accommodated with a modest level of funding. However, we don't want to stop there. For this project to provide an enduring and lasting enterprise solution that will thrive and grow for many years to come, we must also invest continually to help developers adopt the system, thereby migrating an ever-greater number of existing applications and enterprise users off standard cloud services and onto the new infrastructure. The amount of funds needed to achieve ubiquity in the marketplace through partnerships, effective DService ecosystem investment and DApp integrations is hard to calculate. One thing we can say for certain is that the greater the amount of funds raised, the better CPUcoin will be able to market, promote and develop the CGN ecosystem for its long-term success. Therefore, we are raising as much funding in the crowd sale as the market believes we should have to fund this project for the short, medium and long term.

FACTORS INFLUENCING TOKEN PRICING

We intend to price CPUcoin competitively in the IEO. To be competitive with cloud providers, we will attempt to undercut them by providing the same amount of computational resources, but at a discount to fiat.

A workable price for CPUcoin, for purposes of establishing its price in the IEO, will be constrained to a range that is limited on both the low side and the high side as follows:

- On the low side, if CPUcoin were valued at less than the potential income that could be generated by using one CPUcoin worth of computational resources to do other types of mining such as mining Ethereum, then the miner would choose instead to dedicate the resources to do that other type of mining instead and would not participate in our ecosystem (for GPU mining). For CPU-based mining, there are few alternatives so ours may still be very appealing at low prices.
- On the high side, if CPUcoin were valued at more than the cost of procuring one CPUcoin worth of computational from cloud providers (such as AWS), then the end user would choose to use AWS or another cloud provider to deliver the service and would not participate in our ecosystem.

It is our intention that mining CPUcoin be significantly more profitable than mining Ethereum. Our calculations show that with CPUcoin mapping to €1.00 of resources, mining CPUcoin will be more profitable than mining Ethereum so long as the price of ETH remains under €877 ($1000). At the time of this writing, ETH was trading at €234 ($262).

This information can be used by token buyers to evaluate an opportunity to purchase CPUcoin at any given price.

DATES

For clarity: whenever a specific date is given, we are referring to the calendar day beginning at midnight on that date in Greenwich Mean Time (GMT). The start and end of a day will be marked by the span of time starting at midnight (00:00:00:000000) and ending just before midnight (23:59:59:999999) on that day. If you are in a different time zone, an event starting or ending on that date will occur for you at the date and time based on converting the given GMT date and time of the event into your time zone.

TOKEN MODEL

In this section we will explain the token economics ("tokenomics") of the IEO in detail.

The IEO process will result in the distribution of IEO tokens. The IEO token is an ERC-20 token having certain properties necessary for implementing the rules of the IEO. These are used for temporary "bookkeeping" purposes during the IEO and are expected to eventually be redeemed for CPUcoin.

CPUcoin IEO tokens will have the following properties:

1. The sole purpose of the IEO token (symbol CPU) is to facilitate the IEO to and enforce its rules.
2. IEO tokens support the ability to remain locked, and unlock using a time-based, continuous unlock schedule.
3. IEO tokens are non-transferrable and cannot be moved to another wallet.
4. While locked, IEO tokens do not offer any specific rights, and must be held.
5. After having been unlocked, the holder has the right to convert/redeem/exchange IEO tokens one-for-one into CPUcoin utility tokens, which may be traded and/or used in the CGN ecosystem. This transformation is a one-way operation and cannot be undone.
6. There are no other uses for IEO tokens other than the above use cases.

There are three scenarios for IEO tokens that can be granted during the IEO process:

1. Private and presale tokens – These are granted through purchase in each of the presale tranches of the IEO. These are IEO tokens, each representing a future right of exchange for a CPUcoin token, that a buyer has purchased during the presale period of the IEO. These tokens are issued as initially locked, and they will unlock over time in accordance with the presale token unlock schedule.
2. Main sale tokens – Also called "ordinary tokens", these are granted during the main sale of the IEO. These are IEO tokens that represent the right to be exchanged for CPUcoin tokens, that a purchaser has purchased during the presale period of the IEO. These tokens are issued in an unlocked state.
3. Bonus tokens – These are additional IEO tokens to be granted in the presale tranches in differing amounts to token buyers in each tranche. There are two ways bonus tokens can be granted:
   - As an incentive bonus for very early participation, buyers will be granted bonus tokens as a percentage of the number of tokens purchased (percent bonus).
   - As a quantity incentive, buyers who purchase larger blocks of tokens will be granted bonus tokens based on the purchase size (quantity bonus).

   The specific details for each tranche will be set based on market conditions. Please refer to our website CPUcoin for more information.

The IEO will take place in two stages:

1. The token presale tranche series
2. The token main sale auction

We will mint **5,000,000,000 (five billion)** tokens in the IEO, to be divided into four pools:

1. **8%** of tokens, or **400,000,000** (four hundred million) tokens, will be set aside as partner swap tokens. These will be used to provide opportunities for co-promotion and strategic partnerships. Due to the enterprise nature of this IEO, we believe that the need to forge strong partnerships will be essential.

2. **10%** of tokens, or **500,000,000 (five hundred million)** tokens, will be reserved for employee vesting pool.
3. **10%** of tokens, or **500,000,000 (five hundred million)** tokens, will be allocated for advisors and initial buyers, and Extra Bonuses. Up to **45,000,000** of these will be made available to be issued as Additional Bonus tokens (1.5% * 12 months * max number of presale tokens).
4. **72%** of tokens, or 3,600,000,000 (three billion, six hundred million) will be sale and bonus tokens. Of these, **240,000,000 (five hundred million)** tokens will be sold in presale tranches. Of the remaining **3,360,000,000 (three billion, three hundred sixty million)** in this pool, a minority portion will be allocated and set aside as bonus tokens, with the actual number TBD based on purchase patterns. The majority remainder of these tokens will be sold in the token main sale.

The Token Presale Tranche Series

Before the main sale, we will sell a limited number of tokens in a series of tranches. In each tranche, the token will be offered for sale at a set price, with clearly stated rules around how bonus tokens will be granted. Each tranche will have its own unique pricing and bonus structure, so please refer to our website CPUcoin for more information.

After completion of the presale tranches, we must compute the number of tokens remaining to be sold in the main sale, as follows:

$$N = SUM(all\ tokens\ purchased\ \ and\ all\ bonus\ tokens\ issued)$$

We will engage the smart contract to permit frozen tokens to be unlocked over time, in accordance with the corresponding token unlock schedule(s).

STEP FOUR: Compute the number of tokens P to be sold in the main sale, as follows:

$$P = 3{,}600{,}000{,}000 - N$$

The Token Main Sale Auction

After completion of the presale tranche series (the exact date is TBD) we will begin the main event sale. The main sale will be structured as a series of a computed number M of repeating block sales of **10,000,000** tokens, to be distributed over each **29**-hour period, repeating continually until all P tokens have been sold. The final 29-hour sale will be for the remaining number of tokens, which will be for an amount equal to or less than 10,000,000 tokens.

The number of periods M is computed as follows:

$$M = P \,/\, 10{,}000{,}000 \text{ (rounded up to nearest integer)}$$

To help equalize opportunity for buyers worldwide, a 29-hour sale period will be used to regularly shift sale times around the globe in order to provide a convenient time of day in each time zone for buyers to participate in the final hour of bidding, roughly once every 3 ½ days.

In each sale period, buyers will be able to purchase any desired amount of value of tokens until the end of the period. The total amount contributed in total by all buyers will be added up at the end of each sale period. We will then compute an average token price at the end of each period using the aggregate of currency used for all purchases in the period. This price will determine how many IEO tokens each participating purchaser will receive for their purchase in that period.

Sales in the main sale will continue 29 hours at a time until all remaining tokens have been sold. At that point, we will allow the IEO to be permanently and irrevocably closed.

Unlock Schedule

Presale and bonus tokens will be unlocked daily over a 360-day period, starting from the date of purchase. Date of purchase is considered to be either the date of purchase by SAFT, or the date on which funds were committed in an exchange purchase during the presale, whichever is applicable.

Bonus tokens unlock following the same schedule as the presale tokens on which they were granted.

The following table illustrates the percentage of tokens that will be unlocked at the end of each 30-day period:

Presale and bonus token unlock schedule (days since purchase)		
Period	Elapsed days	Percentage Unlocked
0	0	0%
1	30	8.333%
2	60	16.667%
3	90	25%
4	120	33.333%
5	150	41.667%
6	180	50%
7	210	58.333%
8	240	66.667%
9	270	75%
10	300	83.333%
11	330	91.667%
12	360	100%

Locked tokens will not be transferrable out of the wallet. Tokens that have been unlocked can be transferred freely. As time passes, more of the locked tokens will become unlocked and will thereby become transferrable.

Wallet Limitations

A unique CPUcoin IEO token wallet will be required for each individual presale purchase transaction in the IEO. This is because each presale transaction will have its own unlock schedule with a unique start date. A single wallet will not be capable of administering multiple simultaneous unlock schedules for different holdings within the wallet. To participate in the IEO multiple times, the buyer will be required to use a new wallet for each transaction.

Main sale tokens may be purchased using either a new wallet or an existing presale wallet.

Any transfer of unlocked tokens between two wallets will not have any impact on the token unlock that may be underway in either wallet.

Additional Policy

CPUcoin reserves the right to terminate the IEO at any time for purposes of ensuring smooth and stable operation of the markets.

After token sales have concluded, any unsold tokens may be burned in order to control supply.

Post- IEO: Redemption of IEO tokens

As already said, the IEO will be conducted through the purchase and distribution of IEO tokens, which are temporary tokens that act as a placeholder or "proxy" for CPUcoin tokens until the CPUcoin smart contract has been created.

However, the CGN operates using the CPUcoin utility token. The two tokens are instinct and separate, having different duties altogether. They are not interchangeable in terms of use. However, each IEO token holder is entitled to redeem IEO tokens them for CPUcoin ecosystem tokens any time after they have been unlocked.

The reason we have separate tokens is to separate the concerns of handling smart contract programming of the IEO from programming of the utility token. The two implementations are unrelated, and it's ideal (and best practice) to ensure that the IEO programming does not add any technical burden or complexity risk to the utility token's smart contract. Hence, we have a separate contract for the IEO token to specialize in IEO concerns. And that means we must provide a means to convert or redeem IEO tokens into CPUcoin ecosystem tokens.

This capability will be provided through the wallet holder calling a method on the smart contract. We will provide a webpage for doing this which with work together with private key storage and transaction signing wallet tools such as MetaMask or equivalent. We will provide documentation later explaining exactly how to perform the steps.

The following steps are meant to be representative of the final steps. The actual steps may differ in detail from what is stated here, but are expected to be very close to the following:

1. Install the MetaMask (or similar) browser extension. Establish your Ethereum wallet account and private keys as you like, and log into your account. Be sure "privacy mode" is disabled because it prevents Web pages from giving you access to functions of your wallet, which you will need to do the redemption.
2. Navigate to our token redemption webpage, which we will already have provided to the public.
3. Click on "Redeem IEO Tokens"
4. You will be shown your current CPUcoin holdings for both IEO tokens and ecosystem tokens (if any). Your IEO tokens will be displayed as the number of locked tokens and the number of unlocked tokens you hold. If a vesting schedule is in effect for your tokens, that will also be displayed.
5. If you are in possession of any unlocked IEO tokens, you will be given the opportunity to redeem them for ecosystem tokens, and how many of them to redeem.
6. Once you elect to redeem IEO tokens, MetaMask will prompt you to confirm the transaction. If you are using MetaMask with a hardware wallet such as Trezor or Ledger (recommended), you will also be prompted to confirm the transaction on your device.
7. When the transaction executes, it will cause tokens from the IEO token smart contract to be burned, and the same number of tokens in the ecosystem smart contract to be minted in your account.
8. After the transaction has completed and confirmed, the end user will see the requested number of IEO tokens as having been subtracted from their IEO holdings and the same number of ecosystem tokens having been added to their wallet.
9. Blockchain transactions are irreversible. Care must be taken to perform token redemption only when one knows for sure that is what they want to do.

CONCLUSION

Today's massive adoption of cloud-based client-server applications be it web-based software, software installed on PC's or one of the 2-3 million mobile apps available, has resulted in tremendous growth of the need for cloud infrastructure, required to reliably operate the services that power these applications.

To operate these services, application developers must acquire and operate servers, usually in a data center or using cloud services such as AWS and Azure. These services, while convenient, are surprisingly expensive to deploy and operate. Data centers have huge capital costs associated with their sophisticated operational systems. Cloud infrastructure services, themselves hosted in data centers, have the same underlying expense structure and are usually even more costly to operate and use.

Nearly a hundred million servers and more than two billion PC's and mobile devices are powered on and running at any given time, with roughly 30% of those resources sitting idle at any given moment. The amount of powerful, available but untapped computational power is so vast, it's difficult to grasp.

The Content Generation Network is our proposed new Infrastructure-as-a-Service (IaaS) offering. We will pay you (and anybody) to let us use the idle time of servers, mobile devices and PC computers. We will pull together these unused resources and restructure them into a vast, highly scalable, reliable, edge-capable, CDN-compatible, robust network of highly available computational resources. We will then lease this infrastructure in the form of a distributed services delivery infrastructure to the operators of cloud-based applications, who will use our powerful, convenient and reliable systems to operate their services at scale, and at much lower cost.

We have coined the term "DService" to describe the services component of cloud-based applications and DApps that run in the CGN. Anybody will be able to develop their own DService and run it in the CGN.

A new utility token called the CPUcoin will power the service. As the native currency of the CGN, it will be the digital asset used to pay for consumption of services and to make payouts to providers of idle computer time.

We will strive to isolate users of the ecosystem from price fluctuations in the CPUcoin, ensuring that the cost of the service will not increase or decrease in price when the CPUcoin price goes up or down in the markets. We will do this by pricing our services in fiat currency, and by actively converting between fiat currency and CPUcoin on behalf of users of the system. This is very convenient to enterprise customers, who wish to see predictable costs that won't fluctuate, and who may not want to get involved in cryptocurrencies for procuring computational infrastructure.

While the CGN can be used to power long-running jobs like video transcoding and CGI rendering, we have taken the technology a step further by solving a problem that no other solution addresses: support for microservices. We have overcome the transaction rate limitation of blockchain through strategic use of a custom blockchain-anchored microtransaction system, freeing us of costly time delays and enabling us to provide a services layer that is so fast, applications will be able to make requests to services with results returned in near real-time. This makes it possible for applications using the CGN to deliver smooth, responsive user experiences.

An enterprise commitment to adopt the project for real-world use will add great value.

This project will be tested and used by Equilibrium for their Content Compliance Cloud buildout, which will adopt the CGN for use with an existing product line. This product line not only has a proven track record, but also has a new Microsoft Cloud Alliance Partnership and a new hosted cloud service with a large target growth plan. CPUcoin will license MediaRich Server from Equilibrium to create the first DService for dynamic imaging infrastructure and enable it to run on our new scalable platform. The existing MediaGen product has already been seamlessly integrated into many solutions, including Microsoft's SharePoint and Office 365 cloud service, targeting a base of over 200 million installed users. Soon it will be delivered using the first DService, running on the CGN.

Porting MediaGen to run as a DService is not simply a proof of concept but will become a permanent part of the MediaGen product itself. To ensure adoption, we will kick start the ecosystem through a cross-licensing deal with Equilibrium, which will provide a sub-set of its battle-tested MediaGen image processing engine to enable CPUcoin to deploy the first DService. Management believes that this single customer alone will drive a monthly spend of 4.2 million CPUcoin per month, growing to more than 50M in CPUcoin monthly spend by the end of 2021.

But there is a lot more to it than that. Not only will we provide an initial major user for the service, but we will work hard to drive continued adoption and growth of the ecosystem through our plan to invest in and partner with other potential DService providers and DApp Developers to promote an ever-increasing number of applications to run in the CGN. This will expand initial demand for CPUcoin, the lifeblood currency of the ecosystem.

CPUcoin is poised to revolutionize Infrastructure-as-a-Service, by transforming unused resources into a powerful new distributed server system and platform for developers, where they can launch and deploy their own DServices in a blockchain empowered world.

The CPUcoin CGN: A cutting-edge Infrastructure-as-a-Service that reduces current costs and lets you earn money with unused CPU power. This is the missing DServices platform for DApps. Pay as you go with CPUcoin.

For more information about CPUcoin or the Token Sale, please visit CPUcoin.

THE TEAM

To achieve our ambitious goals, CPUcoin has assembled a diverse and multi-talented group of people who all share our common vision. The CPUcoin team combines people with leading industry expertise and proven records in serial entrepreneurship and large-scale multi-billion-dollar infrastructure solutions. The core-team working on CPUcoin is comprised of pioneers in the software and gaming industry, SaaS, ecommerce and top-tier advisory firms.

CPUcoin is convinced that our world-class team along with our industry specific advisors will enable us to meet and exceed our goals.

THE CORE TEAM

Sean Barger – Founder & CEO

With over 30 years of management, engineering and production experience in the software industry, Mr. Barger is a serial entrepreneur with a history of pioneering award-winning products in games, software and SaaS. As the original founder of Equilibrium Mr. Barger is the visionary behind product lines for managing and delivering media content at scale including the award-winning DeBabelizer and the MediaRich Dynamic Media Processing platform. Continuing to innovate, Mr. Barger is now pioneering CPUcoin which is creating the world's first Content Generation Network and DServices platform for decentralizing Infrastructure-as-a-Service.

David "DJ" Jennings – CTO

DJ has an extensive history of managing and growing large scale projects. Prior to joining the CPUcoin team, DJ held the position of Software Architect at NetSuite where he led technical growth. DJ was technically responsible for all ecommerce products. He spent eighteen years architecting the ecommerce platform and product offerings from the ground up. His previous experience also includes the role of lead architect at Adobe where he was responsible for the animation, rendering and compositing technologies of Macromedia Director and Shockwave. DJ is the technical architect of the CGN and will lead and oversee its development and growth.

Matt Butler – VP Engineering

Matt has long history in software engineering, building award-winning enterprise class servers and integrating them in various environments. Matt also bring extensive experience managing software development teams and leading technology innovation. He will be the principal developer of the first DService which will run on the CGN.

Brian Rice – Databases and DApps

Brian has been leading teams of IT professionals for 35+ years. His projects have ranged from bringing new products to market to planning the development and evolution of internal software systems within an organization. Brains previous database design experience makes him the ideal person to be leading the database & DApp design architecture for the CGN.

Oliver Jensen – Strategy & Project Management

Oliver brings with him a background in top-tier management and technology consultancy. Prior to joining the CPUcoin team Oliver held the position of Senior Consultant specializing in Blockchain and Crypto at PwC. Oliver has consulted on some of the world's leading blockchain initiatives, including the blockchain-based partnership deal struck between IBM and Maersk.

ROADMAP

The CPUcoin team subscribes to the following high-level overview roadmap, which shows estimates for several key milestones.

A detailed milestone-based technical roadmap will be produced following the presale. That roadmap will depict the different CGN build stages and will provide a comprehensive timeline detailing all proposed CGN functionalities at each stage.

Q1 2018	• Concept Development and Design of Hybrid Decentralized Architecture
Q2 2018	• Research, Competitive Analysis & Strategic Plan
Q3 2018	• White Paper Creation, Business Model Design & Token Model Design
Q4 2018	• Release CGN Proof of Concept
Q2 2019	•Release CGN TestNet
Q2 2019	•Launch first DService 'MediaGen' as part of the TestNet
Q2 2019	•Private Sale Tranche & Strategic Parterships
Q3 2019	•IEO Presale on Top-Tier Exchange
Q3 2019	•Mutiple Exchange Listings & IEO Press Tour
Q4 2019	•Mainnet Development - Design of Fully Decentralized Architecture
Q1 2021	•Initial Mainnet Launch

APPENDIX A: MEDIAGEN – THE FIRST ENTERPRISE DSERVICE

A MEDIAGEN DSERVICE TECHNICAL OVERVIEW

The MediaGen DService can be conceptualized as a collection of internal components and subservices, best visualized as a top-down diagram illustrating the relationships between components in layers. Some components may span multiple layers, while other components may be contained within a single layer (see Figure 1).

THE MEDIAGEN DSERVICE

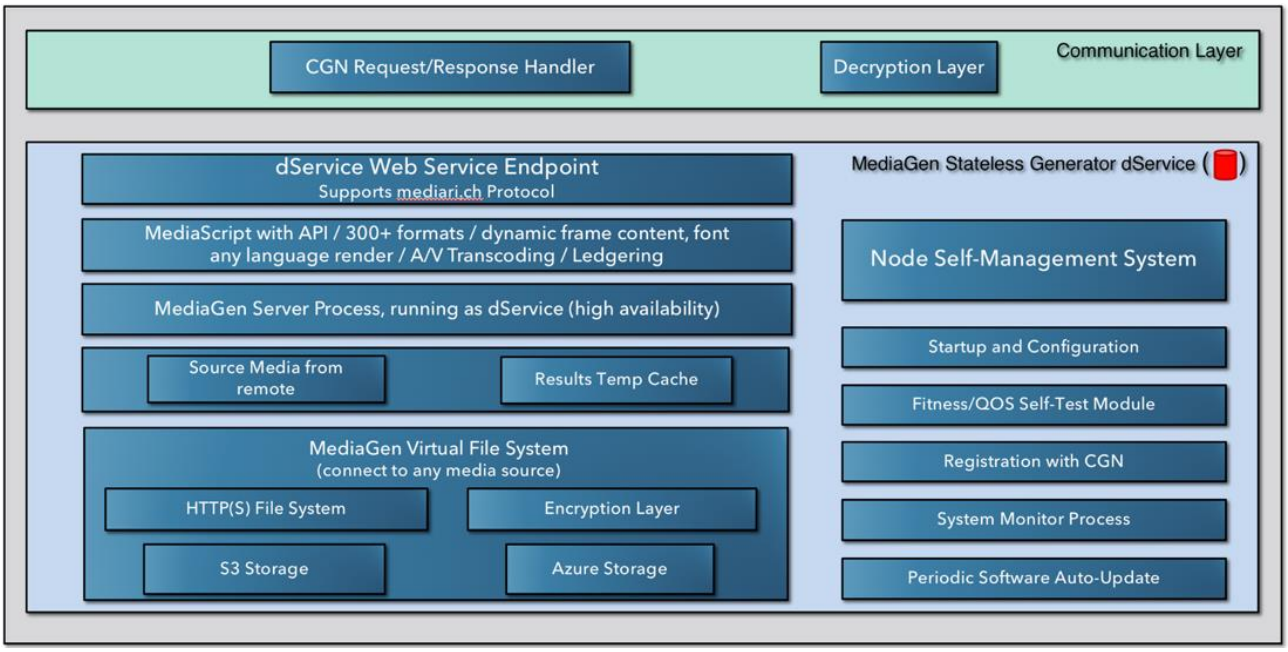


FIGURE 11

MEDIAGEN INTERNALS

The topmost layer exposes a CGN-compatible endpoint for receiving work requests, and maps incoming work to the MediaGen Stateless Generator server process. This is how MediaGen will operate as a natively CGN-compatible DService.

The storage layer manages retrieving source document files, as well as writing out the response document file. All input and output documents and media to be transformed, generated, published are specified in the form of file locations on the internet. Every input and output document will have a HTTP(S) Web address. While one could use a simple Web server for document storage, it is a far better practice to use one of the many available enterprise cloud storage services providers to perform the storage functions. These solutions support sophisticated security and permission models which in an enterprise deployment may be required to keep the work secure. For example, AWS S3 pre-signed or Azure Storage SAS could be used. MediaGen supports configuring storage to suit your needs.

Note: in some cases, the resulting output document may be streamed back to the requestor rather than being placed into cloud file storage.

A final, critical input file, also delivered from cloud storage, is the MediaScript file. This is a custom piece of software code that, when executed, performs the requested document or image transformation task by following a series of specified steps. MediaScript is the server-side programming language that was designed

specifically for this purpose, allowing all kinds of MediaGen image processing tasks (such as annotation, visual transformations, enabling visual effects, etc.) to be defined.

MRL Specification

The MRL is a specific format of URL that is understood by MediaGen and defines the job to be performed by providing the sources of all inputs to be processed. This endpoint can be called, via the CGN, from any website, app or service that wishes to process content and deliver it anywhere, using the MediaGen service, the first service to be built for the CGN.

The most basic MRL looks like this:

http://www.CPUcoin/mgen/io?script=https://www.sample.com/ptag.ms

www.CPUcoin	An endpoint for the CGN, exposing the MediaGen DService endpoint
script	Web URL of the MediaScript file to retrieve and use to execute the request

Additional MRL encoded arguments can be appended to the MRL in the form of a query string. An MRL with a query string looks like this:

http://www.CPUcoin/mgen/io?script=https://www.sample.com/ptag.ms&args="https://sample.com/tomato.jpg",128

args="tomato.jpg",128	This expresses script-specific parameters (any number of parameters can be used within a single MRL)

A MediaScript can also accept arbitrary query parameters on the MRL. For example, the following MRL might be equivalent to the one just above:

http://www.CPUcoin/mgen/io?script=https://www.sample.com/ptag.ms&source=https://www.sample.com/tomato.jpg&width=128

This form of MRL is more readable, in that each parameter is supplied with a name that serves to indicate the purpose of the parameter.

MediaScript

The MediaGen DService employs a rules-based publishing engine that allows media assets to be combined and repurposed without restriction. Nearly any transformation that a developer can conceive of can be performed programmatically. The expressive power of MediaGen is delivered through MediaScript™, a powerful scripting language. MediaScript is based on the same ECMA Script standard as JavaScript, with unique extensions added for media processing. If a Web producer or developer is familiar with JavaScript, they will be able to learn MediaScript in a matter of hours.

MediaScript supports many of the basic ECMA Script objects and programming constructs, but its strength lies in its unique media processing functions. These functions allow users to create image and video manipulation templates that provide precise control of a website's look and feel, even as the images and videos that comprise the site are changed and updated. The product comes complete with sample code and examples for easy implementation of zoom & pan, image and video repurposing, mobile delivery, metadata manipulation, and other functionality.

In summary, MediaScript an extremely flexible and powerful programmatic tool for describing virtually any media processing task, which describes and drives all MediaGen job requests.

MediaScript Example

The following simple MediaScript loads an original TIFF image named car.jpg, rotates the image 33 degrees clockwise, then saves the resulting generated asset as a JPEG file:

```
var image = new Media();
image.load(name @ "car.tif");
image.rotate(angle @ 33);
image.save(type @ "jpeg");
```

The Equilibrium Content Compliance Cloud

The CGN and its DServices can be used to power much more than just user-facing applications and DApps. It can be useful to construct layers of high-level services built upon more basic core services. The Equilibrium Content Compliance Cloud is just such a service, having been built atop the CGN using MediaGen DServices to support its broader functions.

https://www.equilibrium.com/equilibriumnew/content-compliance-cloud.html

The MediaScript Programming API

This link takes you to a document explaining the MediaRich programming language and, contains detailed documentation for the full MediaRich API:

https://eqn.tv/Xz9r3

The MediaGen DService will be scoped to offer a broad subset of MediaRich processing operations, exposing only the API's necessary for general-purpose document transformation operations. Initially we will not support the video and streaming media processing API's of MediaRich.

Therefore, a significant and large subset of the API will be made available in the MegiaGen DService, but not every capability of the MediaRich core will be supported initially.

DISCLAIMER

PLEASE NOTE THAT ANY ACQUISITION AND USE OF TOKENS IS BY NATURE SPECULATIVE AND INTRINSICALLY CARRIES SIGNIFICANT FINANCIAL RISKS, INCLUDING, BUT NOT LIMITED TO, THE POSSIBLE LOSS OF ALL VALUE IN PURCHASED TOKENS. PRIOR TO PURCHASE, CAREFULLY CONSIDER THE POTENTIAL RISKS AND, TO THE EXTENT NECESSARY, CONSULT A LAWYER, ACCOUNTANT, AND/OR TAX PROFESSIONAL OR OTHER ADVISORS TO EVALUATE THE RISK ENTAILED. DO NOT OVERCOMMIT. WE MAKE NO PROMISES NOR WARRANTIES WHATSOEVER OF POSSIBLE GAINS OR RETURNS, IN CASH OR IN ANY OTHER FORM. YOU CAN POTENTIALLY LOSE ALL YOUR CONTRIBUTION. WE ALSO DO NOT GUARANTEE IN ANY WAY THE SUCCESS OR FULL EXECUTION OF OUR APPLICATION OF PROJECT.

TOKENS ARE A UTILITY TOKEN TIED TO OUR APPLICATION. THEY ARE NOT, AND SHALL NOT BE USED IN ANY WAY AS, EQUITY, ASSETS, BONDS, SECURITIES, DEBENTURE, COLLECTIVE INVESTMENT SCHEME, DERIVATIVES OR ANY OTHER FINANCIAL INSTRUMENTS.

YOU UNDERSTAND AND AGREE THAT THE PURCHASE OF TOKENS IS NOT MEANT TO GENERATE ANY PROFIT, INTEREST, GAIN, DIVIDEND NOR EVEN TO MAINTAIN THE SUBSTANCE OF YOUR CONTRIBUTION. YOUR PURCHASE OF TOKENS IS HEREBY ONLY INTENDED TO ALLOW US TO DEVELOP THE APPLICATION OF OUR PROJECT IN VIEW OF EXECUTING ITS VISION AS MENTIONED IN THIS WHITE PAPER.

YOU UNDERSTAND AND AGREE THAT THE TOKEN IS NOT A SECURITY AND DOES IN NO WAY MAKE YOU A SHAREHOLDER OF OUR COMPANY. NOR DOES IT GIVE YOU ANY RIGHT ASSOCIATED TO THE SHAREHOLDING OR THE MANAGEMENT OF ANY COMPANY, INCLUDING THE APPLICATION OF OUR PROJECT, OR ANY OF OUR AFFILIATES OR SUBSIDIARIES.

YOU UNDERSTAND AND AGREE THAT WE DO NOT OWE ANY DEBT TOWARDS YOU WITH RESPECT TO TOKENS AND WILL HAVE NO OBLIGATION TO BUY YOU BACK ANY TOKENS YOU PURCHASED OR TO REFUND YOU IN ANY WAY, NOR TO ENSURE YOU ANY ADVANTAGE OR DISCOUNT WHATSOEVER IN RELATION TO, OR IN EXCHANGE OF, TOKENS YOU PURCHASED.

YOU ARE SOLELY RESPONSIBLE FOR SEEKING LEGAL, BUSINESS, TAX, REGULATORY, ACCOUNTING AND FINANCIAL ADVICE IN THE JURISDICTIONS RELEVANT FOR YOU WHEN PURCHASING TOKENS. YOU SHOULD NOT INTERPRET THE CONTENTS OF THIS WHITE PAPER AS LEGAL, BUSINESS, TAX, ACCOUNTING, INVESTMENT OR OTHER ADVICE.

YOU ACKNOWLEDGE AND AGREE THAT THE TOKEN SALE IS NOT AN EQUITY, ASSETS, BONDS, SECURITIES, DEBENTURE, COLLECTIVE INVESTMENT SCHEME, DERIVATIVES OR ANY OTHER FINANCIAL INSTRUMENTS OR A SALE OF DERIVATIVES WHATSOEVER. YOU UNDERSTAND AND AGREE THAT THE TOKEN SALE IS NOT DONE UNDER THE SUPERVISION OF ANY REGULATOR. THESE TERMS DO NOT REPRESENT A PROSPECTUS FOR THE ISSUANCE OF AN EQUITY, ASSETS, BONDS, SECURITIES, DEBENTURE, COLLECTIVE INVESTMENT SCHEME, DERIVATIVES OR ANY OTHER FINANCIAL INSTRUMENTS.

YOU ACKNOWLEDGE AND AGREE THAT NEITHER THIS WHITE PAPER, NOR ANY OF THE TOKENS, HAVE BEEN OR WILL BE REGISTERED OR FILED UNDER THE SECURITIES LAWS OR REGULATIONS OF ANY JURISDICTION OR APPROVED, RECOMMENDED OR DISAPPROVED BY ANY SECURITIES OR OTHER REGULATORY AUTHORITY NOR HAS ANY SUCH AUTHORITY CONFIRMED THE ACCURACY OR DETERMINED THE ADEQUACY OF THIS WHITE PAPER AND TOKENS. DSERVICES LIMITED IS NOT REGULATED BY THE CAYMAN ISLANDS MONETARY AUTHORITY (CIMA).

ALL TOKEN PURCHASERS WILL HAVE AN OPPORTUNITY TO REVIEW POTENTIAL RISKS INVOLVED IN PURCHASING CPUCOIN TOKENS AS CONTAINED IN THE RISK DISCLOSURES SCHEDULE OF OUR STANDARD TERMS AND CONDITIONS WHICH EACH PURCHASER MUST ENTER INTO IN ORDER TO PURCHASE AND RECEIVE CPUCOIN TOKENS. WE STRONGLY RECOMMEND THAT EACH PURCHASER REVIEW THOSE RISK DISCLOSURES IN DETAIL BEFORE PURCHASING CPUCOIN TOKENS.

IN ADDITION, ON 23 APRIL 2018, CIMA ISSUED AN ADVISORY ON THE POTENTIAL RISKS OF INVESTMENTS IN INITIAL COIN OFFERINGS AND ALL FORMS OF VIRTUAL CURRENCY. CIMA STATED THAT TOKEN PURCHASERS SHOULD THOROUGHLY RESEARCH VIRTUAL CURRENCIES, DIGITAL COINS, TOKENS, AND THE COMPANIES OR ENTITIES BEHIND THEM IN ORDER TO SEPARATE FICTION FROM FACTS. FOR FURTHER INFORMATION ON THE CIMA ADVISORY, TOKEN PURCHASERS ARE ENCOURAGED TO VISIT THE FOLLOWING LINK:

https://www.cima.ky/upimages/noticedoc/1524507769PublicAdvisory-VirtualCurrencies_1524507769.pdf