Digital Commodities

Fall 2017
P. Christian Ondaatje

Introduction

It is somewhat common for a person to start thinking about what he wants to do with his career. It is somewhat less common for the conclusion to be "Digital Commodities!" Naturally, this requires a paper to explain. First, it will be important to offer an internal definition of the word "commodity," before applying it to the virtual world. It will then be essential to present my view of the world as it relates to these goods. This will in turn motivate the solution: Squire. Designing this system presents some fascinating academic challenges in computer science and mathematics, along with opportunities for innovation in behavioral economics and soft sciences. Along with some fascinating academic questions in computer science and mathematics, designing this system presents opportunities for innovation in behavioral economics and soft sciences. In summary, Squire is a two-sided marketplace for digital commodities, which has the potential to produce a distributed supercomputer technically capable of creating consciousness. This is a worthy goal.

Commodities

It may be useful to first establish how I will be using the word "commodity" and its surrounding vocabulary infrastructure. The word is used colloquially in many different contexts, so it is important that we are all on the same page as we begin to examine the class of *things* I will refer to as commodities here.

What is a commodity? Throughout this paper, I will use the term to denote an item or class of items (physical or virtual) that are **fungible**, **useful**, and **combustible**. First, an item is *fungible* if there is no meaningful differentiation between individual instances of the item. For example, I do not care if my coal comes from Wyoming or China - I care that it burns or fits in my kid's stocking. Marx uses wheat in his famous quotation: "*From the taste of wheat, it is not possible to tell who produced it, a Russian serf, a French peasant or an English capitalist.*"[1]

Second, an item is *useful* if its expenditure creates some value. The idea of value creation is directly linked with the *combustibility* property. For example, I can put gasoline in my car to make it go. Burning a commodity (in the abstract sense) is the process through which its usefulness is realized - whereas before you had some commodity good, afterwards you have some useful outcome and no more good. You can eat your wheat, explode your dynamite, or literally burn your wood. This will be a meaningful abstraction from which we can generalize, even for commodities which do not literally disappear when "burned." For example, building a bridge can be thought of as burning through your steel supply to achieve the meaningful outcome of *no bridge* $\longrightarrow$ *yes bridge*.[2]

Commodification Commodification is the process through which a previously specialized good becomes fungible. While this may be unwelcome news for individual suppliers as margins and brand differentiators vanish, ultimately the result is widespread availability and low costs for valuable goods. Introducing the concept of a **commodity service**, a prime example of this phenomenon is the process through which Über has commodified livery. Previously, automobile transport from point A to point B was a specialized service, offered almost exclusively by a small group of suppliers (Taxi companies) with brand-heavy differentiation. While taxi companies are obviously upset with the change in the status quo, livery commodification has made transport much cheaper and widely available for the market as a whole.

Intrinsic Value It is important to note the difference between commodities that are directly versus indirectly useful. The commodities in the examples above are all **directly useful**; you get something out of "burning" them (in the abstract or literal sense). Commodities that are **indirectly useful** have a more tangled or transactional value, but do not satisfy the combustibility property. Gold for example is largely useless. Outside of its applications in computer circuitry and toilet seats, the mineral can do little to nothing practical for its owner. However, it has plenty of indirect uses - you can give it to someone to build your house, or beat up your neighbors. You just can't burn it. More succinctly, directly useful commodities have **intrinsic value**. I will sometimes refer to these as **natural commodities** to reflect the idea that indirectly useful goods such as gold and

[1] Can't believe Marx is going to be the first citation in an incredibly capitalistic paper, but here it is: Karl Marx, "A Contribution to the Critique of Political Economy" contained in the Collected Works of Karl Marx and Frederick Engels: Volume 29, p. 270.

[2] When I call something a "commodity" or "natural commodity" in this paper, I generally mean that it is fungible, useful, and combustible. Preview of coming attractions: as digital goods become increasingly commodified, we will eventually be able to burn through our reserves of raw compute or storage-time to produce meaningful outcomes.

Bitcoin got to be valuable through some sort of societal convergence, whereas natural commodities are objectively useful.

Digital Commodities

In this paper, we will focus on digital commodities with intrinsic value. While there are plenty of useful and fungible digital *currencies* (such as Monero), these constructs do not satisfy the abstract combustibility property. So while the argument can be made that these cryptocurrencies are digital commodities, their lack of direct uses means that they have no intrinsic value - making them a subject for another paper (wink wink).[3] Hopefully as I give some examples of natural digital commodities the distinction will become clear, and the reader will come to share my intuition.

What is a digital commodity? A digital commodity is a commodity that exists in cyberspace rather than meatspace. While there is no physical fungible/useful/combustible item, there is still an abstract *thing* that satisfies those three properties. I claim that there are two main digital commodities: **raw compute** and **storage-time**.[4] Many would consider bandwidth or latency a likely third, however they are more akin digital infrastructure (which is conceptually more difficult to qualify).[5] These two assets are clearly fungible, useful, and combustible.

Storage-Time Imagine that your computer is out of space and you have several gigabytes of data (music, photos, national secrets) that you would like to store for a year. You might use Dropbox, Google Drive,[6] or perhaps even AWS if you are in the mood for tax-code style complex pricing schemes. These companies are selling you a digital commodity: storage-time. You pay them as they burn through byte-hours of this resource, producing the useful outcome of saved data.

Raw Compute Raw Compute is another example of a digital commodity. Let's say that my professor has assigned me one million math problems, and told me that I will fail his class on the economics of computation if I do not produce answers within a week.[7] My computer is way too slow to solve all those problems - so I turn to AWS, IBM, or Los Alamos National Laboratory (if I'm well connected). I pay a considerable sum to have my compute jobs taken care of in a few hours, minutes, or seconds (respectively). These companies are selling you a digital commodity too: raw compute. You pay them as they burn through Floating-Point Operation (FLOP), producing the useful outcome of solved problems and passed courses.

Note: I define two units of measurements here. Storage-time is measured in byte-hours, and raw compute is measured (somewhat poorly) in FLOP. For example, I might have a 10 Terabyte-hour storage request, or a 4-PetaFLOP compute job. These measurements don't perfectly reflect the nuance of needs in digital productivity, but they do give us a basic quantitative vocabulary.

---

3 *Illicit Drugs and Financial Privacy*, P. Christian Ondaatje

4 Which I will sometimes refer to as "compute" and "*storage*," respectively. Once again, we are using the word commodity to denote a *directly useful commodity* - a commodity with intrinsic value.

5 Mesh nets could very well have a place in this discussion.

6 Or *Google Backup and Sync* or whatever they are trying to call it now

7 Based on a true story

Why are they valuable? I claim that storage and compute are the natural digital commodities. As we saw in the examples above, these assets have intrinsic value - storage-time can be burned to persist information, and compute power can be burned to answer questions.[8] In the grand scheme of things, information is everything. The permanent sum total of human effort is the creation of knowledge (data). Everything meaningful can be reduced to knowledge - the knowledge of how to find food, create fire, build cities, or explore space is what gives humanity power. This simplification makes the all-encompassing value of digital commodities clear: **Computation creates knowledge. Storage persists it.**

In a timeline-agnostic view of human progress, it makes sense then that nearly all value will originate from compute power.[9] We will eventually have enough food, water, housing, etc. But our lust for knowledge will always outpace its supply. This makes computation not just *a* resource, but *the* resource.

Problem

Accepting this view of the world, we can start to look at the differences between the present and the future. Optimistically, such differences represent room for improvement. Accepting that commodification is good, and that raw compute and storage-time are natural commodities, any way in which they are not yet commodified is therefore a problem worth solving.

What is wrong with the world right now? Let's say I am an average Joe with a large compute job to run (somewhat average Joe). As we saw above, I have a few options - but they are specialized and difficult to access or understand. For scientific-level computation, I need some kind of business relationship with a large institute - an inefficient friction in what should be an efficient market. Admittedly, not many average Joes have large scientific compute jobs to run. But remember, commodification overhauls the supply side as well as the demand side. So in an ideal world, an (actually average) Joe could connect his laptop to supply the global market for computation whenever he's asleep.

This is where the problem becomes clear - sure, there are some existing digital commodity merchants. For certain applications, that may be enough. But broad spectrum availability of fungible digital goods is not there yet. Compare the supply of compute and storage to the supply of food - almost anyone can supply the asset and it is absolutely essential to human enterprise. However unlike food (which can be exchanged by hand), widespread transaction infrastructure for digital commodities has not yet crystallized.

[8] which is all a computer does in the abstract - answer questions

[9] which in turn is just a refined application of energy - the genesis physical commodity

Existing Solutions: Storage The existing old economy solutions are pricey and cumbersome enough not to warrant further discussion. Of much greater interest are the newly born decentralized new economy (DCNE) solutions, such as Siacoin,[10] Storj,[11] and Filecoin.[12] While I'm sure each system's respective evangelists would crucify me for this claim, these are essentially the same thing; blockchain-based decentralized storage markets. On a technical level they are all sound, using similar proof systems as will be implemented in the Squire network. I claim that the biggest problem these technologies have in getting humanity from specialized point A to commodified point B is non-technical.

Behavioral Economics As the story goes, enough monkeys with typewriters will eventually produce the complete works of William Shakespeare.[13] As in simian publishing, the hardest part of digital commodification is making the monkeys *want* to engage with the machines. The technical problems (for storage) are solved or solvable, but the soft science has a long way to go. As we saw with Über, sometimes goods/services that should be natural commodities can languish in specialization for a long time before a proper two-sided market comes along and opens everything up. The brilliant implementations of Web3 storage show that commodification is an eventuality - but the systems' opacity to non-technicals ensure that adoption will be slow. If Joe wants to store all his photos on the inter-nets and the "best" option requires a CS degree to understand, he's just going to use Dropbox.[14]

Existing Solutions: Compute The state of the union for compute power is even worse. *Every* solution is opaque to Joe. AWS and Google Cloud are the least technical popular systems, and even those require a reasonable level of programming/systems understanding to navigate. Even considering the argument that the average Joe has little demand for raw compute, it is clear that the *supply* side is lacking. Joe should absolutely be able to supplement his income by offering his laptop's spare clock cycles to the global market, even if he doesn't know what that means. Digital currencies and Proof-based mining are as close as we get, however the current technical infrastructure and regulatory complications make crypto prohibitively unfriendly to the average consumer - he wants dollars.

Note: *It can be argued that Ethereum attempts to link the supply of compute to demand, but the structure of the Ethereum Virtual Machine gives it essentially the power of one (very cool and decentralized) computer. While astounding in so many ways, this makes Ethereum an unsuitable marketplace for raw compute.*[15] *Once again, the only solution that comes close (Golem) is also dug firmly into the Web3/Crypto ecosystem.*[16] *As we reasoned above, intellectual barriers to entry must be reduced.*

---

10 `https://sia.tech/whitepaper.pdf`

11 `https://storj.io/storj.pdf`

12 `https://filecoin.io/filecoin.pdf`

13 `https://en.wikipedia.org/wiki/Infinite_monkey_theorem`.
Before the accusations of elitism come flying in - I am most certainly a monkey in this condescending analogy.

14 These concepts are essential to Squire, and much of the understanding in this paragraph arose from discussions with Prof. David Parkes in person and with help from his *Hidden Market Design* paper, though he puts things much more politely.

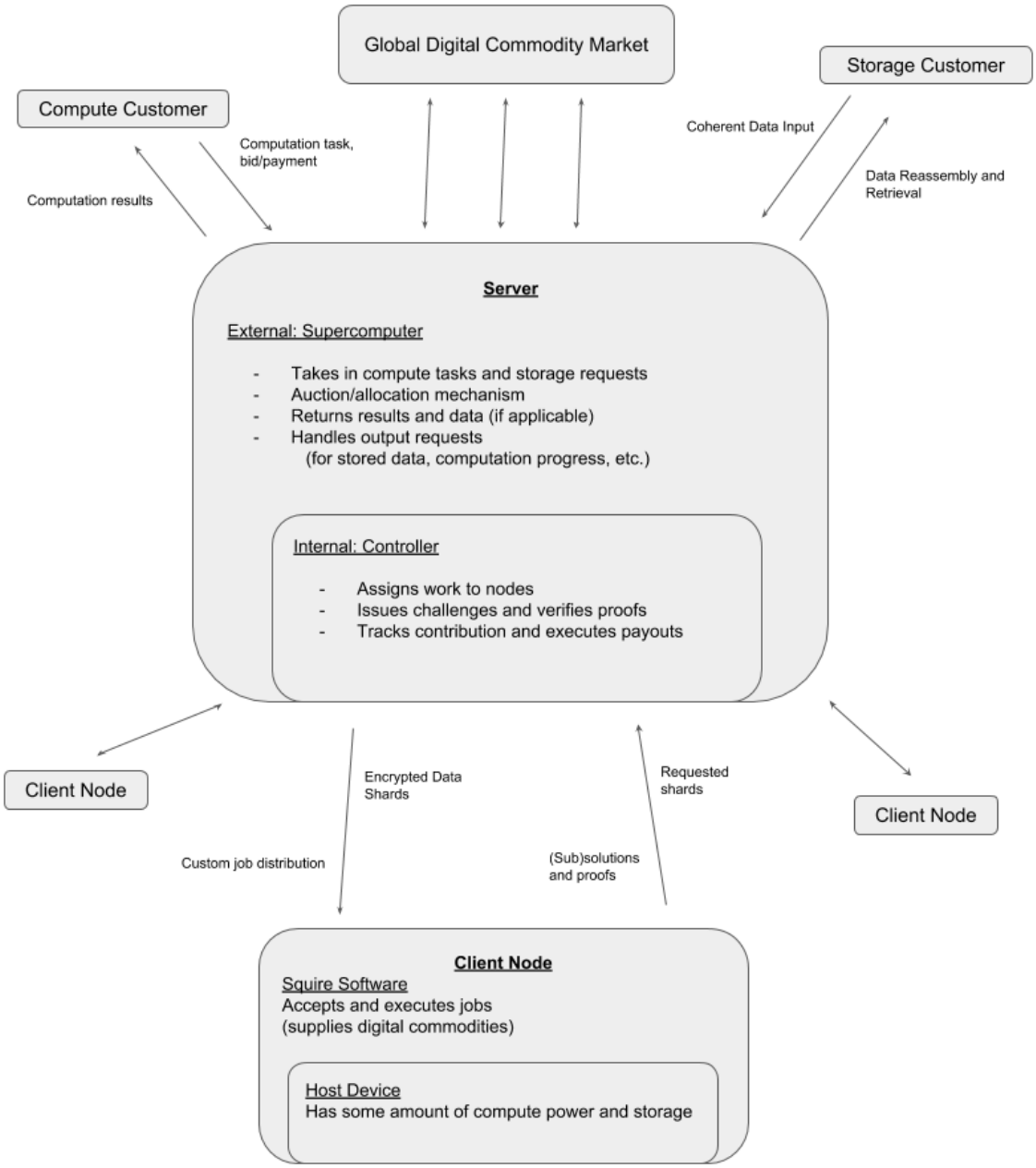
15 Though as a technicality, it is a platform on which such a marketplace can be/is being built. Ether *almost* fits my definition of a digital commodity, since it feels combustible - producing consensus answers from the EVM. However, the Ether isn't destroyed in the process like a natural commodity, so its value is still transactional. Even so, consensus computation is currently a niche good - and it can be replicated in Squire with k-peer computation anyway.

16 `https://golem.network/doc/Golemwhitepaper.pdf`

Solution

I'd like to introduce Squire - a two-sided marketplace for digital commodities. As we examined above, the most important part of such a system is populating the supply side. While the technical challenges are exciting and worthy of discussion, the core innovation comes from the economic structure of the hybrid central-distributed model. By optimizing for humans first, Squire will hopefully be able to achieve much greater scale and succeed where other solutions have failed. Furthermore, by opening with a level of demand already present in mining, I hope to solve the chicken and egg issue inherent to bootstrapping a two-sided market. Squire has the potential to successfully commodify raw compute and storage-time.

The Squire System



Overview The basic element of the system is a relatively small node that contributes compute power and/or storage to the network. This is to be accomplished complex but unobtrusive software that allows individuals to be compensated for resources contributed. The cumulative power and efforts of the network contribute to a distributed supercomputer that can perform useful and profitable tasks. This lends itself to a client-server model.

The Client (user-facing) The Squire client is the software that runs on a node. It leverages and monetizes the spare compute power and storage of the device. While it will default to unobtrusive and simple behavior, the client can be customized to individual preferences. Because this is market design, there is an intersection between computer science and behavioral economics. It is important for the growth of the network and consequently the power of the supercomputer to make participation as easy as possible. So from the user's perspective, all they need to know is that there is some software running on their computer that makes them money. The dirty digi-economic details and market mechanisms will be worked out as a *Hidden Market*.[17] This makes Squire more accessible to the average user, which is part of what will encourage the network to grow exceptionally.

The Client (tech) More technically, the Squire client can do some very cool things. First, there is the core execution: meaningful tasks the software runs on a client device. The baseline of profitability is cryptocurrency mining, but computations are dynamic - the client can run any profitable binary. It interacts with the server to receive tasks, store data, submit proofs, and accept compensation. For computation resources, mining profitability can be viewed as a "reserve price" in an auction - the baseline profitability a bidder must surpass to access the compute power of the network.[18] Iterations of the Squire client may use GPUs, CPUs, ASICS or some inclusive combination to optimize monetization. Furthermore, computation need not be restricted to typical computers. Phones, cars, and of course smart fridges can contribute to the Squire network.[19]

The Server (internal) Internally, the server acts as a controller for each node and the network as a whole. Responsibilities include assigning work, tracking client contributions, and making corresponding payouts (a dangerous game). Since the network will just be pool mining in the early days, tracking client contributions will be reasonably immune to strategization.[20]

The Server (external) This is the supercomputer. If the controller can assign work to client devices, and its owner can decide what that work is, then logically the aggregation of compute resources in the Squire network is "programmable." Since it would take only 4000 modern GPUs to outperform the most powerful known supercomputer in America (Titan), I will claim that this system has a good shot at becoming a **Distributed Supercomputer**.

Note: *this assumes the work being assigned is parallelizable*

[17] `https://nrs.harvard.edu/urn-3:HUL.InstRepos:9121285`

[18] Economics and Computation, D. C. Parkes and S. Seuken, Chapter 6 (Auction Design) and Chapter 10 (Online Advertising Markets)

[19] `https://electrek.co/2017/11/29/tesla-mining-bitcoin-model-s-supercharger-power`

[20] This comes from the game-theoretic properties of strategy-proofness that are the core motivation for Proof-of-Work algorithms and their defense against the Byzantine Generals' Problem. https://wikipedia.org/Byzantine-Generals-Problem

Discussion: Compute

Now that we have outlined the system, it is time to start defending it. I view the world through a theory that I call the "**Law of Incentives**:" if there is a large enough financial incentive for some realizable outcome, it will happen regardless of its morality.[21] For this reason, it is essential for Squire to either disincentivize bad outcomes, or make them unrealizable.

The Cryptographer's Stone Core to this goal is the quest for meaningful Proof-of-Work, which is the holy grail of cryptocurrency. One of the most common criticisms of existing digital currencies is the apparent wastefulness of their mining infrastructures. While the market clearly disproves the knee-jerk intuition that this computation is worthless, it would be a beautiful thing if that computation also produced meaningful knowledge. The concept of "meaningful" is normative, so a more objective truth is that full customizability of questions solvable by blockchains (rather than just "what's the next hash?") would be incredibly valuable. If hashing an iterated nonce is truly the algorithm that generates maximum economic value, then nothing would change. But in the likely event that it isn't, discovery of proof of work algorithm for customizable computation would fundamentally alter the productivity of all blockchains.

Related Work: Primecoin My first run-in with digital currencies came with an unsuccessful attempt to mine Primecoin in 2013 as a then non-technical teenager.[22] Primecoin's hashing algorithm produced new undiscovered prime numbers. Those who have heard of the currency might now scoff - by all metrics it was a complete failure. At the time, though, it promised to be the first blockchain whose mining algorithm offered some kind of meaningful computation. In fact another (much more technical) teenager, Vitalik Buterin, was a big proponent for the same reason.[23,24]

Ultimately though, Primecoin failed to satisfy the corollary to the Law of Financial Incentives. The hashing algorithm produced some nice trophies for the social good, but nothing extra for an individual's good. This meant that any given miner had no improved financial incentive to dedicate computation to the Primecoin blockchain, and it failed to gain traction. The lesson here is clear - there is nothing more important than proper incentives when designing an economic network.

[21] Or "*Law of Financial Incentives*" - Much of which crystallized in CS 136, under Prof. David Parkes: "*Economics of Computation*" and `https://nrs.harvard.edu/urn-3:HUL.InstRepos:4064046`

[22] `https://primecoin.io/bin/primecoin-paper.pdf`

[23] In fact, he first started building Ethereum on top of Primecoin: `https://twitter.com/VitalikButerin/status/929804867568373760`

[24] `https://bitcoinmagazine.com/articles/defense-alternative-cryptocurrencies`, `https://bitcoinmagazine.com/articles/primecoin-the-cryptocurrency-whose-mining-is-actually-useful-1373298534`

Related Work: Golem As we touched on earlier, Golem is a decentralized market for compute power.[25] There is nothing inherently *wrong* with it's technical design, however the cryptocurrency baggage for a non-blockchain system combined with the complexities of P2P transactions mean that it is currently ahead of its time. Viewed once again from the lens of behavioral economics, these represent fundamental obstacles to participation in a platform that aims to connect the typical consumer to global demand for compute power.

Squire as a compromise Truly meaningful PoW is all-important and the person who discovers it will be very wealthy. But as we saw with Primecoin, financial incentives must be aligned. And as we can see with Golem, even if the financial incentives get there the monkeys must be made to type. Squire's hybrid model represents a compromise - sacrificing decentralization for meaningful computation and ease of use, while preserving the economic incentives of distributed digital productivity. It's not even a real sacrifice - most mining is only nominally decentralized, with the bulk of hashing power concentrated in a few pools for almost any given cryptocurrency. So the hybrid model appears to be where compute power is already gravitating, just without any improvement in computation productivity. Squire is not quite generalized PoW, but the effect is the same in the abstract - unlocking the latent value of the mining model and dramatically increasing the worlds accessible supply of compute power. As for the behavioral economics side, there are two main components. The first and most obvious aspect is accessibility of operation; simple setup, dollar denominations, unobtrusive operation, etc. The second is more nuanced - financial incentives to achieve accessibility of thought.

Supply-side Fed Siacoin, Storj, and Golem all suffer adoption problems - one side is not developed enough to deliver on the platform's promise, which hinders the growth of the opposite side. As hinted above, Squire's biggest innovation may be starting with one side "fed." There is already a very profitable application of raw compute - cryptocurrency mining! The first iteration of the system simply assigns all it's nodes the basic job: *hash*. So while the hybrid model decouples computation resources from mining and leaves the network free to solve the most profitable problem of the day, it remains *at least* as profitable as cryptocurrency mining. I claim that this solves the chicken and egg problem inherent to bootstrapping a two-sided market.

Strategic behavior This also has some very beneficial properties for the system's defensibility. Any aspect of the system that is not strategy-proof will quickly be exploited, with (rational) bad actors finding every way to fudge results and avoid extra work. Starting with cryptocurrency mining as the specific compute job assigned to the network allows us to utilize the friendly properties of proof-based algorithms (designed for just these types of problems). So from the outset, we can be sure that payment is indeed correlated to production - PoW cryptocurrency cannot be produced otherwise.

However, the value of the supercomputer will grow with its set of solvable problems. For this reason, it is very important to promote good behavior more generally - which comes down to economic incentives. Since payment will be executed in periods, the controller has leverage over misbehaving nodes. One way bad actors could be discouraged is through "punitive damages" for lost data or incorrect solutions.

Note: *"Checking for incorrect solutions" sounds simple, but it is actually Computer Science's holy grail. Quick and dirty on P vs. NP: we call the set of problems that are easy to solve "P." We call the set of problems that we think are hard to solve, but with easy to check answers "NP." So the central server can only handle problems in NP.*

[25] `https://golem.network/doc/Golemwhitepaper.pdf`

Attack Vectors Common vectors in a system like this include *Whitewashing attacks*[26] and *Sybil attacks*.[27] A Whitewasher behaves badly, then creates a new identity with no negative reputation. This attack is somewhat difficult to execute in Squire's centralized model, since meaningful gain comes from linking a bank account or other non-trivial payment destination. A Sybil attack occurs when one or more users attempt to undermine a reputation system by marking bad actors as friendly. This is obviously not possible with Squire - a fundamental advantage of the hybrid model. Account theft is another danger, however if the payment destination were hashed into a node's submissions, Squire could maintain outbound security. Any hacker that managed to get their account credited would have to have performed meaningful work for the network - which is a bit like robbing a bank by signing up as a teller and then slowly taking the money paycheck by paycheck. There are innumerable ways in which bad actors can cause trouble, but hopefully the examples above present a convincing argument for the relative advantages of the hybrid central-distributed model in securing the network.

Random Verification One simple method computation results could be checked is through random verifications. Since it distributes the binaries and inputs, Squire Command can simply check random submissions by re-executing the job itself. Doing this for every job would obviously make client nodes redundant, so it could instead be applied randomly to submissions with probability p. The exact parameters would have to be tinkered with, but theoretically a proper dynamic value of p would create a strong enough financial incentive for good behavior to minimize incorrect results. Most importantly, this method can expand the class of computation Squire is able to perform past NP.

k-peers Server-side verification is actually a pretty bad idea in practice, since the server would have to do an enormous amount of computation for problems outside of NP. However, a similar verification process can instead be assigned to Squire nodes like any other compute job. If several random nodes produce the same result, it is extremely likely that answer is correct.[28] Similar to data storage, there is a trade-off here between capacity and reliability. More redundant execution equals less raw power in the supercomputer. However, if each computation is run k times on many different nodes in some random/collusion-proof way, we can be much more confident in the solution. In addition, the parameter k could eventually be determined by the market - with easily verifiable problems requiring low k and exponential jobs paying for high k. When combined with the punitive incentive structures discussed above, this becomes a reasonably secure system.

[26] `https://en.wikipedia.org/wiki/Reputation_system#Attack_classification`

[27] `https://en.wikipedia.org/wiki/Sybil_attack`

[28] Assuming most of the network is not compromised or colluding.

Discussion: Storage

As we saw with the market design for raw compute, there are exciting academic challenges to be solved in building the Squire system. The same applies to storage. While the chicken/egg launch dictates that compute functionality (and not compute privacy) is more important to implement in the short term, that doesn't take away from the timeline-agnostic importance of storage-time.[29] Furthermore, privacy is much more important for storage than computation, which means that we ~~get to play with lots of cool toys~~ will utilize several algorithms in highly scientific ways.

Basics For anyone that spends too much time thinking about distributed data, this is probably intuitive: it should be encrypted, sharded, and redundant. First, you don't want everyone on the network reading your files, so it makes sense to distribute encrypted data only.[30] Second, there are many benefits to sharding - It improves security, increases availability, and enables standardized data sizes.[31] This means that a node with 10MB of storage to offer can contribute to the network even all the files being distributed are 10GB. Finally, this is an incentives market so we are herding cats - eventually one of them is going to do something weird like lose all your data. For that reason, it is important that the network stores many copies of any given file. Sharding makes this easier to enforce. If a shard is lost, it can be replaced with a copy from a partner node. Ultimately, we can abstract away the concept of a file and just think about properly storing shards.

Strategic behavior Jumping right in, there are many ways in which a bad actor could minimize work and maximize income by acting ... badly. The first is simply accepting storage jobs and never actually saving the data. Such delinquency is easily punishable, but gets to the core of the problem - how can we make sure a node is actually holding onto the data a client needs? A naïve solution would be to download the data periodically and check it, but that is incredibly wasteful. A better idea is to save a hash of the data before assigning it to a node, and have the node periodically verify the hash of the data. This is a step in the right direction, but once again a bad actor could easily avoid doing meaningful work. All a node would have to do is download once, calculate and save the hash, then discard the data. It would simply submit the hash periodically and move on with its life, happily collecting payments along the way. Of course we could punish this with a financial disincentive, but that doesn't change the fact that the damage is done - potentially mission-critical data has been lost. Furthermore, such a fraudulent node might go years accepting compensation for data it didn't actually hold before an archive request comes along to expose the ruse.

---

[29] Technically SIA/Storj hosting qualifies as a chicken/egg solution for storage, but it is minimally profitable at the scale of consumer devices.

[30] This does not affect the storage proofs we will discuss here, since we can just calculate local hashes on the encrypted data. Also this presents a good opportunity for homomorphic encryption algorithms. That wasn't important to include, I just wanted to sound smart since all the smart kids are talking about homomorphic encryption.

Homomorphic. Encryption.

[31] Economics and Computation, D. C. Parkes and S. Seuken, Chapter 5 (Peer-to-Peer Systems)

Blockchain timestamping This is where things get interesting, and the relevant literature comes into play. My first idea was to utilize some popular blockchain to timestamp hashes. Since the world doesn't know the signature of the latest block until it is mined, it can be hashed together with the data to prove that the client node had the data after a given time. This fixes the "hash and trash" strategy outlined above.

k-peers (again) This system could then be abstracted to the same k-peers system as above - instead of the central server storing and checking data (which would defeat the purpose), some number of random partner nodes would check the validity of each others' hashes. Once again, mistakes result in financial punishments, and the system is fairly secure if the network is not compromised and partners are random. It is important that nodes do not discover the identities of their "partner" nodes, since they will likely have the same verification partner multiple times. This could lead to collusion if nodes were allowed to communicate, so verification should be quarantined on the central server.[32]

Related Work: Siacoin After studying the related work, there is clear room for improvement on the timestamping algorithm above. Siacoin's solution to the strategization above involves random hashes.[33] Instead of storing the data and constantly checking against a local (or partner) copy with a shared timestamp, a SIA client precomputes tons of hashes of random segments of the data before distributing it. It secretly stores these proofs and periodically issues "challenges." When the time comes to check up on the data, it tells the hosting node to hash a certain segment of the data. If the host produces a valid proof, it almost certainly has your data. The only way around this is to compute every possible hash of combinations of data in the file, which is exponential and thus computationally impractical. Note that there is no need for peers in this algorithm.

Related Work: Storj Storj improves further with the introduction of *salts*.[34] By hashing the data together with a large random number or "salt," you effectively send the solution space to infinity. All the client has to do is store the salts, the data range that produced them, and the correct proof (hash). It can calculate tons of proofs before issuing challenges one at a time, checking that a client produces the correct hash. This means that generating a proof without the data is not just impractical but impossible* - you'd be better off trying to hack the server to steal the answers.

Proof of Storage-Time These are "*Proof of Storage-Time*" algorithms.[35] They are much cleaner than our computation "proofs," which may provide some intuition as to why no generalized meaningful Proof-of-Work algorithm exists yet. My intuition is that generalized compute includes problems not in P or NP - therefore a general computation PoW blockchain cannot exist without a similar k-peering mechanism. While sad, it does mean that Squire will continue to be viable as long as our current understanding of mathematics holds up.

[32] I wanted to include the timestamping peers algorithm because I was quite pleased with myself for coming up with it, but then I read the Siacoin and Storj whitepapers and realized that I'm actually not that smart.

[33] `https://sia.tech/whitepaper.pdf`

[34] `https://storj.io/storj.pdf`

[35] Or *Proof of Storage* for short. This cannot be abbreviated as PoS. Not for the reason you think - PoS would confuse things with *Proof-of-Stake* algorithms, so we will use PoST.

Impact

Having now presented the Squire system, let's start thinking about its potential impact. Aside from the obvious profitability of operating a large exchange,[36] there are some surprisingly achievable milestones for Squire that could fundamentally alter the global digital commodity ecosystem.

Global Data Part of what is burning up the world's data supply are the hundreds of articles constantly reporting how rapidly we are exhausting it. While they may be incentivized to exaggerate our future storage needs, a Seagate-sponsored IDC report claims that "by 2025 the global datasphere will grow to 163 zettabytes."[37] This is 163 hundred-billion terabytes. That seems like an insurmountable number, but a closer examination of the potential of the Squire system shows what distributed infrastructure can do. Without using any datacenters, and limiting participating nodes to personal computers (excluding phones, smart fridges, etc.), we can actually get on that scale.

Imagine for sake of argument that in 2025 the average personal computer has approximately 10TB of storage.[38] Approximately 420MM personal computers were sold globally in 2017.[39] Using that number as an underestimate, what would happen in 2025 if 50 percent of the world's personal computers were connected to Squire? If each node offers half its storage to the network, Squire's total capacity would land on the order of 10 to 11 ZB! Keep in mind, the Seagate report has a financial incentive to overestimate and the distributed estimate only includes personal computers.[40] Regardless, this should hopefully convince the reader that a system like Squire has the potential to handle a *significant portion of the world's data needs.*

FLOP This is where it gets even more exciting, and we don't even need to go to 2025. While there isn't a perfect generalized measurement of compute power yet, Floating-point Operations per Second (FLOP/s)[41] appear to be entering the public consciousness - and can be somewhat abstracted to give a decent picture of relative speed. A FLOP is just a calculation. How many calculations can this computer do every second? That's how powerful it is - essentially horsepower for computers.[42] When I started this research at Wolfepack, the Wolfe operated around 5.1 TeraFLOP/s.[43] A PetaFLOP is 1000 times greater than a TeraFLOP. The fastest supercomputer in the United States, Cray Titan, operates at about 27 PFLOP/s. That is fast. It should be, given how expensive it was to create. However IBM Watson, the Titan, and all supercomputers like it are fully on the centralized, old economy side of things. If we got just 4000 Wolfe-powered nodes to connect to the Squire Network, it could completely overpower the United States' fastest supercomputer!

---

36 FEES!

37 `https://seagate.com/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf`

38 This is not scientific, but hopefully the reader agrees it is within the realm of possibility. A hefty portion of personal computers shipping today are in the 0.5-2TB range.

39 "Personal computer" defined as a laptop, desktop, or tablet:
`https://statista.com/statistics/272595/global-shipments-forecast-for-tablets-laptops-and-desktop-pcs`

40 We are not even considering datacenters, IoT devices, etc. - where most of the world's storage capacity exists.

41 A terrible acronym in more than one way. Obviously, the letters don't line up. Also, the colloquial plural (FLOPs) or singular (FLOP) forms are strange when expanded. Everyone should please try to use FLOP/s for speed and FLOPs for quantity (like "how many FLOPs will it take to compute that?"). FLOP is also acceptable as a quantifying adjective: "that could be a 1.21-GigaFLOP job." "FLOPS" is right out.

42 Horsepower is actually the worst unit of measurement. Want to know how much horsepower one horse generates? Approximately 14. There's no adult in charge - which is why everyone will probably still be using "FLOPS" in twenty years.

43 The TFLOP is 1,000,000 times the MegaFLOP - which was first executed by the Anaheim Ducks.

Parallelization Now before the technical readers yell at me, it's important that I add some nuance to that statement. Parallelization is a very important concept, and much of this decade's explosion of compute power is attributable to improvement in parallel computation hardware. it has led to many of this century's greatest technical developments. In your laptop, there are most likely two main computation devices - the CPU and the GPU. By and large, the CPU is great at handling non-parallelizable, or "serial" operations. You can think of it like a little Will Hunting quickly solving complex math problems under your keyboard. The GPU, on the other hand, is like a classroom full of third graders. Or more accurately, a stadium full of third graders. You would never give them the same complex math you're giving Will. However, if you have millions of small addition problems, you should absolutely divide the work up and use the third graders. Each one of the little buggers can solve a portion of the workload independently - in "parallel." As fast as Will is, that could take him ages. In recent years, CPU clock speeds have stagnated. However, our stadiums have added tons of seats and the third graders have become eighth graders - GPU power is growing at an incredible rate.

Distributed Supercomputer Now imagine if you had hundreds of stadiums full of advanced mathematicians. The world's accessible compute power would grow astronomically.[44] This is the perfect opportunity for our new economy central-distributed model. If every computer owner has an incentive to connect their devices to the FLOP commodity market, then by the Law of Incentives the system will naturally become incredibly powerful. We saw that just 4000 devices can outpace the Cray Titan. If 200,000 were to connect, the distributed supercomputer would have a theoretical bandwidth of one ExaFLOP - the amount of compute power we estimate it would take to run a prior-free human brain simulation.[45] This means that the compute power theorized to be necessary for Artificial Intelligence *already exists* - it just has to be connected. For this reason, I believe the first digital consciousness will come from a distributed supercomputer. Such an event would be computer science's version of proving God exists - it's us.[46]

[44] For fully parallelizable operations - though it's possible we could achieve a middle ground with a low-orbit satellite mesh. Maybe not with an atmosphere, but more on that in another piece probably.

[45] `https://en.wikipedia.org/wiki/Mind_uploading#Computational_complexity`
(Spiking Neural Network, huge parallelization asterisk again)

[46] A great piece to get a basic understanding of the significance of AGI and why that discovery might quickly be followed by a reduction to "minor deity" status: `https://waitbutwhy.com/2015/01/artificial-intelligence-revolution`

Conclusion

We started by examining the concept of a commodity, and how it applies to the digital world. We showed that raw compute and storage-time are natural digital commodities, with inherent value. The markets for these goods are woefully underdeveloped however - despite the incredible impact such markets could have. Almost all of human knowledge will be produced by computation in the long run. Storage-time persists that knowledge, allowing humanity to build on its base of information and achieve compound growth in aggregate intelligence. To that end, there is no more important effort than improving the accessible supply of compute power and storage. To do so is to fundamentally increase our capacity for intelligence.

That goal is my core motivator. The shortcomings of our current position at stagnating Point A are strikingly clear to me, and every effort must be made to get us to commodified Point B. The financial rewards are obvious - I believe the first dollar-denominated trillionaires will come from the new economy.[47] Besides the four-comma potential in successfully generalizing the Proof-of-Work model to all computation, Squire could greatly enrich its creators (and the globe) on a level that transcends economics. Cryptocurrency's decentralized new-economic properties have already recruited a significant portion of the world's compute power. In the absence of generalized PoW, the hybrid central-distributed model can mimic mining's financial incentives and attract the population's productivity to inherently valuable computation. As we showed earlier, that quantity of raw compute power *could* fuel Artificial General Intelligence. If Squire has the potential to create consciousness and rid us of our solace, it is a pursuit to which I am content dedicating my career.

[47] Keep an eye out:
`https://etherscan.io/accounts`
`https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html`
`https://bitslog.wordpress.com/2013/04/17/the-well-deserved-fortune-of-satoshi-nakamoto`
We won't know if it's an XMR whale though.

Miscellaneous References

https://en.wikipedia.org/wiki/Commodity

https://golem.network/doc/Golemwhitepaper.pdf

https://storj.io/storj.pdf

https://sia.tech

https://economist.com/node/18185752

https://storagenewsletter.com/2017/04/05/total-ww-data-to-reach-163-zettabytes-by-2025-idc

https://dfinity.org/tech

https://sia.tech/whitepaper.pdf

https://youtube.com/watch?v=lik9aaFIsl4

https://dash.harvard.edu/bitstream/handle/1/4064046/Shneidman_Rationality.pdf

Jeffrey Shneidman and David C. Parkes. Rationality and self-interest in peer to peer networks. In *2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03), 2003.*

Economics and Computation, D. C. Parkes and S. Seuken, Cambridge University Press 2018.

Seuken, Sven, Kamal Jain, and David C. Parkes. 2010. Hidden market design. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10): July 11-15, 2010, Atlanta, GA, ed. American Association for Artificial Intelligence, 1498-1503. Menlo Park, CA: AAAI Press.

