L1s vs. L2s, Rollups vs. Integrated, General-purpose vs. App-specific

Thank you to Mert Mumtaz and Neel Somani for valuable feedback and review.

Introduction

This short post will describe the concrete tradeoffs of:

- L1s vs. L2s
- Rollups vs. integrated
- App-specific vs. general-purpose

We need a better understanding of what architectures to build and when. Otherwise, we'll continue to get a mish-mash of confusing infrastructure that no user can understand or interact with. This is the most common mistake I see:

Reminder to pls stop conflating app-specific & rollup

App-specific <> general purpose

Rollup <> integrated/monolithic https://t.co/cy5qMPp4jd

— Jon Charbonneau (@jon_charb) November 24, 2023

As noted in Eclipse's introductory post ahead of their upcoming mainnet launch:

There's often a false dichotomy presented between the modular rollup vision vs. the ability to have a single chain with massive scale, parallelized execution, and shared state. "Modular" is often conflated with "app-specific", which would lead you to believe that rollups mean a world of many fragmented and low-throughput chains. We challenge that idea.

Rollups and L2s are not bad UX. Fragmented

rollups and L2s are bad UX. Properly designed rollups and L2s should improve UX

.

Rollups vs. Integrated

All chains can eventually adopt the best tech (e.g.,DAS + ZK) if it proves useful. As discussed in my last reportDo Rollups Inherit Security?

, the distinction we're left with then is roughly the following:

- "Rollups" a.k.a. "Modular"
- Build logically separate chains which post data to their host chain (DA layer). They reuse the host chain's consensus.
- "Integrated" a.k.a. "Monolithic"
- Integrate everything into one protocol with its own consensus. Don't post data to a separate host chain. (Even if the DA layer and execution layer are in some sense logically separate pieces of the shared protocol).

Solana and Eclipse represent the parallel paths, as similarly shown in Syncracy's [Solana Thesis

](https://www.syncracy.io/writing/solana-thesis):

As I discussed in my recent Uncommon Core episode with Hasu, both approaches will have long-term value.

Solana takes the approach of bundling everything into one consensus. It pursues minimum latency (slot times currently average ~400-500ms with hopes to hit 200ms in the future) while maintaining a large validator set (~2,000). This incredible achievement required several technical breakthroughs.

However, these two goals (max decentralization + minimum latency) are inherently in tension. It's incredibly challenging to keep this consensus set stable while running at max speed and throughput. TowerBFT has no formal safety or liveness analysis, and it's unclear whether proof-of-history is currently useful and resilient in an adversarial model or can simply be removed. The economics of a low-latency system also of course increase the incentive to centralize.

Eclipse takes the approach of unbundling consensus. Rollups can have a smaller hand-picked sequencer set (potentially

even run by a single operator) to operate a controlled environment. This can increase reliability and reduce latency even further, offering a Web2 product with the benefits of crypto rails. Code, a payments app deployed as somewhat of an L2 on Solana using durable nonces, is similar in its desire for instant and reliable payments. Beyond the great UX of near-instant latency, pushing the lower bound even further is also necessary for high-value low-latency financial applications.

Rollups can then post their data to another decentralized consensus set for more robust verification at a slower time-scale

. For example, Celestia has 15s block times with single-slot finality, which actually isn't even so different from Solana! Solana has confirmations ~400ms, then finality is reached after 32 slots (~12.8s).

There's no free lunch here. There's a potential tradeoff on the properties of the real-time validator set (e.g., Solana has far more validators than rollups do sequencers) vs. the guarantees provided (e.g., controlled resilient environment, even lower latency, etc.). The proper level of commitment given (and at what time scale) is a spectrum. Open engineering questions remain, and the best fit will likely vary by use case. It goes without saying that cost is also essential here, so scalable DA layers such as Celestia (which Eclipse uses) will be needed.

Eclipse obviously won't replace Solana. Each makes different tradeoffs and pursues a different market. Solana remains the heart and soul of SVM development, and it is likely to see many new applications deployed there as a result. However, it's clear that there will be more than one SVM chain long-term (Pyth already even is). The future isn't single-chain, and the SVM is simply amazing tech. Eclipse is starting that trend of exporting it to L2, but I'd expect others to realize the value here and follow their lead.

L1 vs. L2

I'm using L1 and L2 here in the more popular sense to include rollups, validiums, etc. As covered in Vitalik's Different types of layer 2s

:

Two-way validating bridges are almost enough to make a chain a validium. The main remaining ingredient is a social commitment that if something exceptional happens in Ethereum that makes the bridge no longer work, the other chain will hard-fork in response.

What makes an L1 vs. an L2 is effectively how it deals with forks. A validium would revert if its L1 reverts a block, and it would hard-fork if its base layer hard-forks. To upgrade the L2, some form of L2 governance must live on the L1 as a bridge contract which is legible to the L1.

Now, why would we use such a thing? Does it make sense for a chain to delegate its fork-choice to an underlying L1 and root itself around its bridge there?

Despite common beliefs that the L1 wars are over, Ethereum has won, and all Ethereum competitors want to become an L2 now - Ethereum L2s are not the best solution for all chains.

Ethereum L2s are often viewed as the most secure and scalable way to build a chain. However, these security properties are often deeply misunderstood as discussed in my last report. Just posting a proof to Ethereum and delegating your fork-choice rule there doesn't magically make your chain super secure.

The argument that all chains must deploy as Ethereum L2s for their own security is most often incorrect. Rather, the primary benefit of L2s has been the ability to tap into Ethereum's network effects (users, liquidity, developers, tooling, etc.). It's a go-to-market strategy.

Fighting for attention makes sense given that attention is the only scarce resource in crypto. L2s naturally get to be front-and-center with the developers, users, media, etc. that matter the most. Being an L2 used to be enough to get that attention.

However, the attention received from being an L2 is waning. The list of live and upcoming Ethereum L2s is now far too long for any individual to keep track of. Chains pivoting to L2s don't get the attention boost that the early movers did (e.g., Optimism and Arbitrum). Even the flood of long-awaited zkEVMs are struggling to attract users, applications, and value.

So, being an L2 alone doesn't guarantee everyone's attention anymore. However, it can still offer a product advantage vs. standalone chains if

you can attract attention in some other manner. For example, turning a pyramid scheme into a square can attract ~$700mm into a multisig with no L2. Alternatively, you could build Ethereum's first SVM L2.

Assuming you have a product with attention, let's now consider how being an L2 may help a chain tap into Ethereum's user base and offer a better product experience. It would do so primarily by leveraging Ethereum-native assets (e.g., ETH) in a favorable manner (e.g., bridges with attractive security and/or UX).

The value of this is broadly dependent on two core assumptions:

1. That existing Ethereum assets are important to the given use case (e.g., DeFi reliant on ETH)

If your application is heavily reliant on Ethereum-ecosystem assets, then an L2 architecture may be valuable. If you don't care about Ethereum assets at all, then being an Ethereum L2 is not particularly valuable. Ethereum-based assets are clearly the most important ones in crypto today, so there's a big market to be served here today.

Looking ahead at an industry level, the core question here is what will crypto's new and valuable state creation be in the future?

- If this future state is increasingly untethered from the current Ethereum-native state (e.g., unique new state, RWAs, etc.), then the appeal of L2s may be diminished.
- If this future state is heavily reliant on the current Ethereum-native state (e.g., ETH trading), then L2s may play a prominent role.

The former scenario takes the view that we've only seen a drop in the bucket of what crypto will be, and you shouldn't over-index on what's here today. The latter scenario takes the view that crypto development and applications will be incredibly path-dependent, so the current state will influence the outcome.

Both are true to an extent, but I think any optimistic view of the industry's long-term prospects leans to the former. There will be a lot of new and unique state which we can't even reason about that's untethered from today's state. The current state of crypto is a drop in the bucket compared to the expected future state.

For example, the commonly cited "settlement assurances" of Ethereum mean little for real-world assets (RWAs) such as stablecoins (e.g., USDC) or tokenized treasury bills. They're "settled" when the issuer (e.g., Circle) deems them to be.

In this scenario, the draw of being an Ethereum L2 may diminish as a share of applications. A new USDC-based payment application doesn't inherently care whether it's an Ethereum L2 or not. They just want the cheapest, fastest, most reliable infrastructure that allows them to offer users the best product experience.

New state creation has historically been a hurdle for Solana, though we clearly see the winds changing here. Many well-known DeFi and infrastructure projects on Solana are now launching tokens, with more to come. This is kickstarting the Solana flywheel.

Honestly surprised so many people are claiming airdrops on Solana. I can only assume people do not realize that Solana transactions don’t settle back to Ethereum

— Gwart (@GwartyGwart) December 7, 2023

1. That Ethereum ←→ L2 bridges are preferable to Ethereum ←→ L1 bridges (e.g., due to security and/or UX reasons)

Let's assume that the first assumption was indeed met for a given use case (i.e., Ethereum-native are quite valuable to your application). Then, we need to ask if an L2 can expose these assets in a more favorable manner vs. a separate L1. Let's say a user has some ETH, and they want to trade it for USDC. Where do they go?

While bridge security is often cited as a motivation here, this argument seems tenuous based on available information. Many of the largest rollup bridges don't even have proofs, have whitelisted proofs, multisig-controlled upgrades, or they literally don't even have an L2.

This compares to classic consensus-verifier bridging (e.g., IBC). In practice, there have been no major validator quorum failures in such scenarios. Bridge failures have generally occurred due to hacks and/or compromised bridge multisigs (which L2s are equally susceptible to).

While security improvements are less convincing to me here, convenient access to Ethereum users and assets is

a major benefit of L2 bridging today in my view. Rollups such as Base, Optimism, and Arbitrum feel

more like extensions of Ethereum. Users keep the same wallets and addresses, the native gas token is a single canonical version of ETH, ETH dominates DeFi such as all trading pairs, social apps price NFTs in ETH and pay creators in ETH (e.g., friend.tech), deposits into the L2 are instant (because they would reorg together), etc.

Users can't be expected to reason through which bridge to use, analyze the varying security assumptions, get one of several available wrapped tokens, acquire the chain's native token for gas, etc. They just want to bridge their ETH, get ETH on the other side, and keep using the L2 as they would use Ethereum or any other L2. This is why Eclipse will just use ETH as their native token used for gas. Forcing in a new gas token is detrimental to UX.

So, why can't Solana just provide the same advantages as an Ethereum L2? This actually becomes more of an engineering question than anything fundamental, and it's going to get easier over time. This is true for gas tokens as well as other UX challenges related to simply not using the EVM (which is not inherent to L1 vs. L2):

- Gas token
- Gas payments can be abstracted away from users, allowing users to pay in whatever gas token they choose.

- Bridging
- Bridging is likely to harden and standardize more over time, leading to less confusion from users and liquidity fragmentation.
- Wallets
- MetaMask's recently unveiled Snaps extend MetaMask support to non-EVM chains via third party integrations built on top, such as via Drift's or Solflare's MetaMask Snaps.
- Developer Experience
- Language barriers will break down. Projects like Solang and Neon may help Solidity developers build on Solana, and projects like Stylus can help Rust developers build on Arbitrum.

In the future, it would even be possible for ETH to play a role in Solana DeFi if users expressed strong preferences for ETH with Solana's speed and scale. In practice, it's far more likely that these users with Ethereum-native assets will continue to use them within the Ethereum L2 ecosystem for the reasons we've discussed, assuming they have access to comparably scalable L2s.

App-specific vs. General-purpose

Regardless of whether a chain is an L1 or an L2, there's a clear need to increase single-chain throughput by scaling execution. Rollups should not imply fragmentation. Uniting many homogenous chains under one stateful shared sequencer just ends up looking like one parallelized chain from a scaling perspective with more challenging UX.

Agreed, vertical scaling > horizontal scaling.

Although all these superchain approaches seem to converge on this as well now, by functionally constructing one big chain out of man small ones. Trades off less VM complexity for more rigid state space separation.

— ansgar.eth (@ansgar_eth) December 6th, 2023

We often see "dedicated blockspace" cited as the reason to deploy an app-specific rollup However, this misconception has arisen largely due to the unnecessary limitations of the single-threaded EVM with global fee markets. A parallelized SVM with local fee markets greatly reduces the need for app-chains. Hosting more apps on shared infrastructure meaningfully reduces developer and user complexity. Cross-chain UX and developer complexity in a many-chain world is an underappreciated existential risk.

This isn't to say there will be literally one chain at the end of the day. I broadly see four arguments to deploy your own chain:

1. Scalability & Dedicated Blockspace
2. As mentioned above, this one is usually unconvincing. One NFT mint shouldn't effectively shut down the rest of the chain, but the answer generally isn't to go make another chain. This is alleviated by a parallelized VM with local fee markets. To the extent that the bandwidth of the entire network is saturated though, local fee markets will not help (i.e., fees for the shared chain will rise globally). Then you need another chain.
3. Sovereignty
4. Governance in crypto is still quite poor, and having your own chain to fork can be a helpful coordination mechanism My intuition is that this is a very rare circumstance, but it's hard to say with certainty. This is in line with MakerDAO's interest in an app-chain.
5. Customizability
6. Consensus-level customizations can be valuable for certain applications (e.g., dYdX v4), but these cases are empirically few and far between to date. Even dYdX, the shining example of an app-chain, will "likely move more towards generalized execution on the dYdX chain" according to Antonio in his recent Bell Curve episode. The need for full-stack customizability which can't be solved on a shared chain has generally been lacking for most of crypto. Nearly all new rollups continue to be just vanilla EVM forks with new tokens.
7. Value Capture
8. This is arguably a subset of customizability, but it's pretty important so we'll split it out. It can be more challenging to internalize value on shared infrastructure which isn't built with only your application in mind. App-chains can make it easier to allocate value to the responsible applications. However, this is an engineering effort more than a fundamental one, and there's ongoing research in Solana around how to do exactly this Additionally, note that there is a serious overhead cost in deploying your own chain vs. amortizing costs across shared infrastructure.

The primary motivation for launching an app-chain today is often the perceived narrative boost and/or token utility for a

struggling project. The bear market downturn and corresponding lack of application growth incentivized development and funding of an overly complicated architecture which resulted in new projects needed to solve self-inflicted complexities.

Launching your own chain today brings painful and unnecessary tradeoffs (complexity, cost, worse UX, fragmented liquidity, etc.) which most apps can't justify for the incremental benefits. The infrastructure needed to make this UX competitive appears distant. This isn't to say that there's no reason for app-chains to ever exist (there certainly are). Rather, we've just massively over-indexed in this direction on narrative as an industry, and so the current trend towards re-bundling is clearly beneficial given the current state.

Conclusion

Solana has rightfully gained a lot of momentum in recent months. This sharp correction has largely been a recognition of the current state of multi-chain UX - it’s fragmented and painful. The UX of using Solana applications is simply incredible. Smooth and fast.

Rollups and L2s have gotten a bad rep for UX, but the real problem is fragmentation

. We associate rollups and L2s with fragmented horizontal scaling because in practice most of them have forked the EVM as-is and use constrained DA bandwidth. They end up expensive and clunky to use.

However, this isn’t fundamental. Vertically scaling with a powerful VM on a scalable DA layer addresses these UX and cost problems. Some level of re-bundling the stack for both L1s and L2s is likely to occur. L2s and rollups should improve UX if used properly

. That should be their real selling point.

Both approaches have their merits. We just need to do a better job of first asking ourselves "what market is this product trying to address?" and "how can this architecture solve for what I need?" before we go building the next L1, L2, L3...

The author of this report has material personal investments in SOL, JitoSOL, ETH, stETH, and TIA. DBA is an investor in Eclipse Laboratories, Inc. and SOL.